Universidad de Valladolid

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO DE INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN, MENCIÓN EN SISTEMAS ELECTRÓNICOS

Multímetro Digital USB

Autor:

Antonio Barón García

Tutor:

Jesús Manuel Hernández Mangas

Valladolid, 18 de Septiembre de 2020

Título: **Multímetro Digital USB**
Autor: **Antonio Barón García**
Tutor: **Jesús Manuel Hernández Mangas**
Departamento: **Departamento de Electrónica**

TRIBUNAL

Presidente: **Ruth Pinacho**
Vocal: **Jesús Arias**
Secretario: **Jesús M. Hernández**
Suplente: **Iván Santos**
Suplente: **Pedro López**

FECHA:

CALIFICACIÓN:

Resumen

Vivimos en una sociedad marcada por la tecnología con un sinfín de posibilidades. Los dispositivos electrónicos hacen la vida más fácil a sus usuarios, convirtiéndose en elementos indispensables para su día a día. Estos, en su gran mayoría, cuentan con conectores USB como medio de interconexión entre dispositivos o para proveerlos de alimentación eléctrica. Pero como todo, y sin excepción, llega un momento en que sus componentes se degradan, o vienen con defectos perdiendo sus prestaciones.

Por ello, para poder diagnosticar el funcionamiento correcto, el objetivo de este proyecto es diseñar e implementar un dispositivo, que mediante conexión USB sea capaz de realizar medidas de tensión, corriente, resistencia de carga y acumulación de capacidad y energía con gran precisión. De la misma forma, para tener un control de la disipación térmica se instalará un sensor de temperatura próximo a las líneas de alimentación de los conectores USB. Una vez procesada la información por software, se podrá visualizar en una pantalla que irá acoplada al dispositivo.

Como una de las características principales, se podrá analizar en mayor profundidad las variaciones de tensión y corriente respecto al tiempo. Mediante una representación gráfica podremos movernos en diferentes intervalos de tiempo, que podrán ser ajustados según nuestras necesidades.

Abstract

We live in a society marked by technology with endless possibilities. Electronic devices make life easier for their users, becoming indispensable elements in their daily lives. The vast majority of these devices have USB connectors as a means of interconnection between devices or to provide them with electrical power. But like everything, and without exception, there comes a time when their components degrade, or come with defects losing their performance.

Therefore, in order to diagnose the correct functioning, the objective of this project is to design and implement a device, which by means of a USB connection is capable of carrying out measurements of voltage, current, charging resistance and accumulation of capacity and energy. In the same way, in order to have a control of the thermal dissipation, a temperature sensor will be installed next to the supply lines of the USB connectors. Once the information has been processed by software, it can be viewed on a screen that will be attached to the device.

As one of the main characteristics, it will be possible to analyze in greater depth the variations of voltage and current with respect to time. By means of a graphic representation we will be able to move in different time intervals, which can be adjusted according to our needs.

Índice General

Índice de figuras

1.Introducción

En la actual era tecnológica y con el uso más acentuado de dispositivos electrónicos por parte de la sociedad, estos se han convertido en elementos indispensables a lo largo del día debido a la multitud de utilidades que ofrecen a los consumidores.

Un dispositivo electrónico se caracteriza por ser una combinación de componentes electrónicos interconectados destinados a controlar y aprovechar las señales eléctricas. De esta manera utilizan la electricidad para almacenar, transportar o transformar la información. Cómo ejemplo de estos dispositivos existe el teléfono móvil, altavoces, tabletas, ordenadores, etc. Todos ellos cuentan con conexiones USB (*Universal Serial Bus*) [1] de diferentes tipos, ya sea para conectar, comunicar o proveer de alimentación eléctrica entre ordenadores, periféricos y dispositivos electrónicos.

Como ejemplo tenemos los teléfonos móviles, que cuentan con una autonomía que varía en relación con la capacidad de cada modelo. Todo consumidor desea tener un terminal con una duración conforme a sus características, que le permita hacer un uso prolongado y no tener que estar pendiente de su consumo, sacándole el máximo partido a la vida útil de su dispositivo. Por ello, es necesario realizar un correcto mantenimiento asegurándose de que los cargadores, los cables y el dispositivo que se esté alimentando se encuentre en condiciones óptimas.

Por ello, los multímetros, siendo el medio de medida, proporcionan información precisa para el diagnóstico.

1.1. Objetivo

El objetivo de este proyecto es diseñar e implementar un multímetro digital (DMM) con conexión USB, capaz de realizar medidas de tensión y corriente del bus USB. Para ello, se integran una serie de componentes eléctricos a partir de los cuales se obtendrán los datos, junto con una pantalla para la visualización y análisis.

A la hora de realizar un análisis, podemos encontrar dispositivos alimentados mediante cable, y dispositivos que constan de una batería que los provee de autonomía cuando no están conectados a la red eléctrica. Ponemos especial interés en estos últimos, tomando como ejemplo los teléfonos móviles o bancos de energía (*powerbanks*). En el proceso de carga se producen variaciones, tanto en los niveles de tensión, como en el flujo de corriente, dependiendo de la cantidad de carga almacenada y del estándar que se esté utilizando.

Hay diversos estándares de carga, de entre todos ellos hacemos hincapié en los de carga rápida, que se caracterizan por presentar unas mayores fluctuaciones de tensión y corriente, así como unas potencias más elevadas. Durante estos procesos de carga es fácil verse un incremento de la temperatura debido a la disipación de energía en forma de calor, por ello, es esencial el uso de un sensor térmico para su control.

De esta manera, un usuario puede comprobar si se produce adecuadamente la alimentación de un dispositivo, observando los datos que muestra el multímetro digital.

1.2. Decisiones de diseño

Hoy en día, la conexión más común mediante la cual se conecta, comunica o provee de alimentación eléctrica a los dispositivos electrónicos, es mediante un conector USB tipo A.

Por ello a la hora de diseñar el multímetro digital, se ha optado por elegir dicho conector con el fin de evitar el uso de sondas de medida, de esta manera el usuario solo tiene que conectar el DMM en el conector, y empezar a obtener datos de medida. Respecto al estándar USB, se ha elegido el USB 2.0, puesto que el estándar USB 3.0 no incluye pines de alimentación extra, como vemos en la figura 1, lo que supondría un incremento el coste final obteniendo las mismas prestaciones.

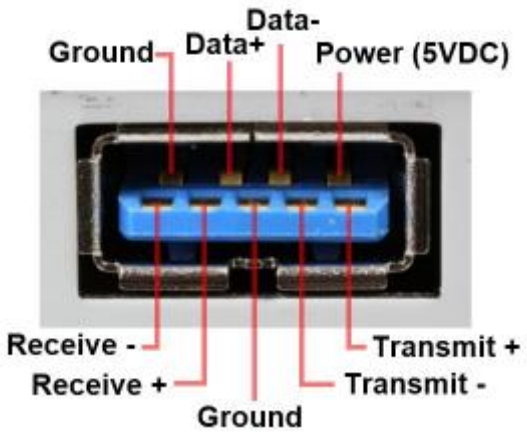


Figura 1: Conector USB tipo A 3.0

A la hora de elegir las especificaciones que debe poseer nuestro DMM, nos hemos fijado en aquellos que se encuentran ya en el mercado, como el caso del UM24C [2] con USB 2.0 tipo A qué se muestra en la figura 2, donde observamos un diseño hardware con formato "*sandwich*", utilizando las placas de circuito impreso como carcasa del dispositivo. Teniendo esto presente, utilizaremos el mismo modo de montaje con la idea de evitar generar una carcasa posterior, consiguiendo reducir el coste final. Este diseño se realiza de manera cuidadosa ya que sería contraproducente si alguna conexión fuese accesible al manejar el dispositivo.

Figura 2: Multímetro digital UM24 con conexión USB 2.0

Puesto que tratamos con un multímetro digital para conectores USB, la idea es que tenga el menor tamaño posible con el fin de acceder a cualquier conexión de cualquier dispositivo, por ello se decide el uso de un módulo TFT de 1.33 pulgadas, un tamaño lo suficientemente pequeño para ocupar poco espacio y que se puedan visualizar los datos correctamente. Este módulo cuenta con una pantalla TFT-LCD IPS de baja resolución, que nos da unos ángulos de visión muy amplios gracias a su tecnología IPS, con colores intensos. También cuenta con un slot para una tarjeta microSD, para en caso de ser necesario, se pueda implementar vía software el almacenamiento de datos en ella.

Por último, en lo que respecta al análisis de datos, se opta por la elección de un microcontrolador con un conversor analógico digital (CAD) de 12 bits, de esta manera se obtiene una resolución adecuada de la señal acondicionada mediante una serie de circuitos. Con esta decisión abaratamos costes, evitando el tener que añadir al diseño un componente extra para la conversión de los datos medidos.

1.3. ¿Qué es la carga rápida (Quick Charge)? [3]

Durante un tiempo, con las baterías de litio de la época, la potencia que se podía transmitir era más que suficiente, pero la cosa cambió cuando estas evolucionaron a las LiPo (Li-Ion Po). Estas baterías aumentaban la densidad energética y el rendimiento, permitiendo mayores corrientes de carga y descarga. Con ello la carga ya no estaba limitada por la batería sino por el sistema de carga donde el problema residía en el cable empleado y la sección de los conductores limitando la corriente máxima.

De esta manera, surgen nuevos métodos con el fin de reducir los tiempos de carga. Entre ellos están los llamados estándares de carga rápida (*Quick Charge*), desarrollados por Qualcomm y siendo el más común del mercado, por lo que nos centraremos en ellos expresamente.

Con respecto a los estándares anteriores de carga, QC optimiza la transferencia de energía en las primeras etapas de la carga, permitiendo un mayor voltaje en el dispositivo y por ende una potencia superior. Así es como nace la carga rápida.

1.3.1 Carga de batería

Partiendo de la que la carga de una batería no es lineal, sino que su proceso de carga se divide en tres fases distintas. En la primera fase, la tensión de la batería va de unos 2V hasta su pico, alrededor de los 4.2V, tensión que varía según la batería exacta. Durante este proceso, la corriente que se suministra es constante, hasta que llega a su punto máximo de tensión.

En la segunda fase, el voltaje de la batería se mantiene constante al máximo valor y la corriente empieza a descender. En la tercera fase, mantenemos el nivel máximo de voltaje, y la corriente empieza a descender más lentamente que en la fase anterior, prolongando la carga para el 20% final de capacidad.

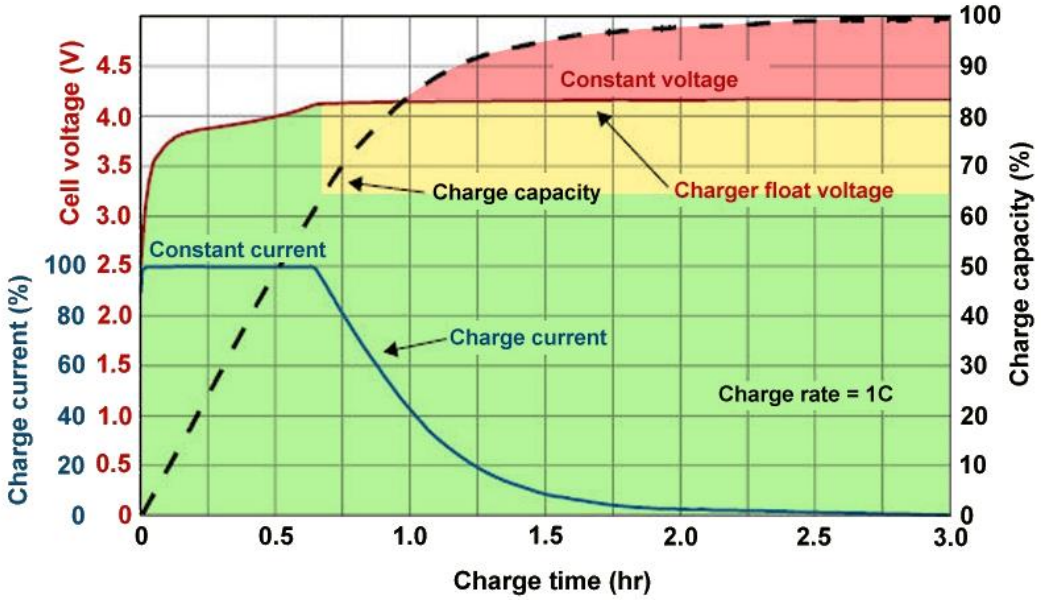


Figura 3: Proceso de carga rápida de una batería de iones de litio [3]

Como podemos observar en la figura 3, en la primera fase es cuando se produce el proceso de carga más rápido, llegando a valores entre el 50% o 60% de capacidad aproximadamente en el menor tiempo posible. En el momento en el que pasamos a la segunda fase, la carga de la batería se realiza más lentamente hasta valores cercanos al 80% de la capacidad de la batería. A partir de aquí, entramos en un proceso aún más lento de carga, durante un tiempo mayor, para un 20% restante. Como norma general, la carga de baterías utiliza su máxima potencia cuando tiene unos niveles muy bajos y va reduciéndose progresivamente a medida que se acerca a la carga completa con el fin de no disminuir la vida útil de la batería.

1.3.2. Carga rápida 2.0 (QC 2.0)

La base de la carga rápida reside en el **aumento de la tensión del cargador** usando la negociación entre el dispositivo y el cargador. De esta manera hay un control más específico sobre cuando hay que aumentarlo o disminuirlo.

Hay dos motivos principales que nos animan a ello, uno de ellos es que las fuentes de alimentación que se utilizan para la carga de los dispositivos móviles son las fuentes conmutadas, puesto que son más eficientes que las lineales y generan mucho menos calor que debe ser disipado, un punto a tener en cuenta para evitar dañar las baterías.

El segundo motivo viene relacionado con la perdida de energía a través de los cables que conectan la fuente con el dispositivo que se alimenta. Los cables ejercen una resistencia al paso de corriente, lo que supone una caída de la tensión según la ley de Ohm(V=I*R) y, por consiguiente, la perdida de potencia (P=V*I).

La pérdida de potencia se convierte en calor por culpa del efecto Joule, que se produce cuando los electrones pierden su energía cinética al chocarse con los átomos del material conductor por el que se propagan.

Por eso, transmitir la misma potencia con un voltaje más alto y una corriente más baja, supone menor disipación térmica, que se traduce en una menor perdida de potencia haciendo la carga más eficiente.

Mientras que con la carga estándar se tiene en el cargador un voltaje constante de 5V a una intensidad de 1A o 2A con una potencia máxima de 10W, en la carga rápida 2.0(QC 2.0) el cargador ofrece voltajes de 5V/2A, 9V/2A y 12V/1.67A en diferentes etapas, con una potencia máxima de 18W.

Tomando como referencia el punto anterior, los estándares de carga rápida son más efectivas cuando la batería se encuentra con una carga inferior al 50%. Por ello, el dispositivo solicitara al cargador la etapa de tensión más elevada (12V/1.67A ó 9V/2A) que pueda soportar, conforme con sus características, cuando la cantidad de carga en la batería sea inferior al 50%. Esto disminuye considerablemente el tiempo de carga en este intervalo hasta casi un 50%, y es cuando se eleva más la temperatura en toda la etapa de carga del dispositivo, al ofrecer una potencia mayor.

A partir del 50% de carga de la batería, se aplicará la tensión estándar de 5V/2A y se ralentizará la carga con el fin de prevenir sobrecalentamientos o sobrecargas.

1.3.3. Carga rápida 3.0 (QC 3.0)

Pronto se vio que la carga podía ser aún más eficiente si el voltaje de entrada se iba adaptando según las necesidades de cada momento. Así surgió QC 3.0, que manteniendo la máxima potencia a 18W en el intervalo de carga del 0% al 50%, permitía variaciones de tensión de 0.2V entre 3.6V y 12V.

Según el estándar se conseguían los siguientes valores ofrecidos por el cargador: 3.6-6.5V / 3A, 6.5-9V / 2A, 9-12V / 1.5A.

Tras el QC 3.0, como no podía ser menos, llegó el QC 4, que no es una evolución directa de QC 3.0, pero es capaz de suministrar entre 2.5A y 4.6A con unos niveles de tensión de 3.6V a 20V.

Con este estándar llegó una mejora en las baterías que las permitía almacenar y administrar cada vez más energía, además de la aparición del USB Tipo C con la capacidad de soportar hasta 100W superando cualquier otro tipo de conector. QC 4 se caracteriza por incorporar nuevas tecnologías que mejoran la eficiencia de la carga, la gestión de la energía y el comportamiento térmico.

Junto con estos estándares, hay muchos más de otras marcas que en cierta medida son parecidos, teniendo en común unas variaciones de tensión que ronda entre los 3.6V y 20V y una corriente inferior a 6A. [4]

1.4. La ley de Ohm

La ley de Ohm, postulada por el físico y matemático Alemán Georg Simon Ohm, es una ley básica de circuitos eléctricos. Establece que la diferencia de potencial (V) que aplicamos entre dos extremos de un conductor es directamente proporcional al producto de la intensidad de corriente que circula por dicho conductor y la resistencia eléctrica que opone [5].

$$V = I * R$$

Según el sistema internacional de unidades, esas magnitudes corresponden, respectivamente, con voltios (V), amperios (A) y ohmios (Ω).

Es muy fácil ver a simple vista que, si conocemos dos valores de la formula, es muy fácil determinar el tercero. Y es ese el pilar fundamental del funcionamiento de los multímetros digitales (DMM).

Como ejemplo, a la hora de la obtención de la intensidad de corriente (I), tomando como base que conocemos el valor de la resistencia eléctrica (R), solo necesitamos realizar medidas de la diferencia de potencial (V) que se produce y por procesamiento software determinar la intensidad de corriente (I) para mostrar por pantalla.

1.5. Resolución y precisión

A la hora de utilizar un multímetro digital, es muy importante conocer una serie de conceptos básicos sobre las especificaciones con el fin de comprender en mayor medida los resultados obtenidos.

Lo primero es la **resolución**, que es la precisión con la que el multímetro realiza una medición. Cuando conocemos esto, somos capaces de determinar hasta qué punto se pueden observar cambios en la señal, ya sean mayores o más leves. Por ejemplo, si estamos midiendo una señal de 5V y contamos con una resolución de 1mV, eso significa que podremos observar como mínimo cambio ese valor, pudiendo llegar a observar 5.001V o 4.999V.

Para describir la resolución se utilizan los dígitos. En el caso de tener 4 1/2 dígitos, el multímetro puede mostrar cuatro dígitos completos con un valor de 0 a 9 y un "medio" digito que solo puede mostrar un valor de 0 o 1. Por lo tanto podrá mostrar valores hasta 19.999, lo cual nos dará una resolución de 1mV para valores inferiores a 20V, en caso de que se supere dicho valor, perderíamos resolución, siendo la siguiente de 10mV.

Lo segundo a tener en cuenta es la **precisión** de medición, que es el error permisible más grande que puede ocurrir en condiciones de funcionamiento específicas. Es decir, es la diferencia máxima que hay entre el valor de medición que nos muestra el multímetro digital y el valor real de la señal que se está midiendo. De manera habitual, este valor se expresa como un porcentaje sobre el valor de lectura, lo que significa que, si obtenemos una medida de 5V y tenemos una precisión de medición del 1%, el valor real está comprendido entre 4.95V y 5.05V.

En las especificaciones también se puede incluir un rango de dígitos que se añaden a los valores de precisión básicos, que nos indica cuanto puede variar un valor en la parte más a la derecha de la pantalla. Si tomamos el ejemplo anterior de una medida de 5V y la precisión es del 1% y 4 dígitos de error, esto significa que el valor real está comprendido entre 4.91V y 5.09V.[6]

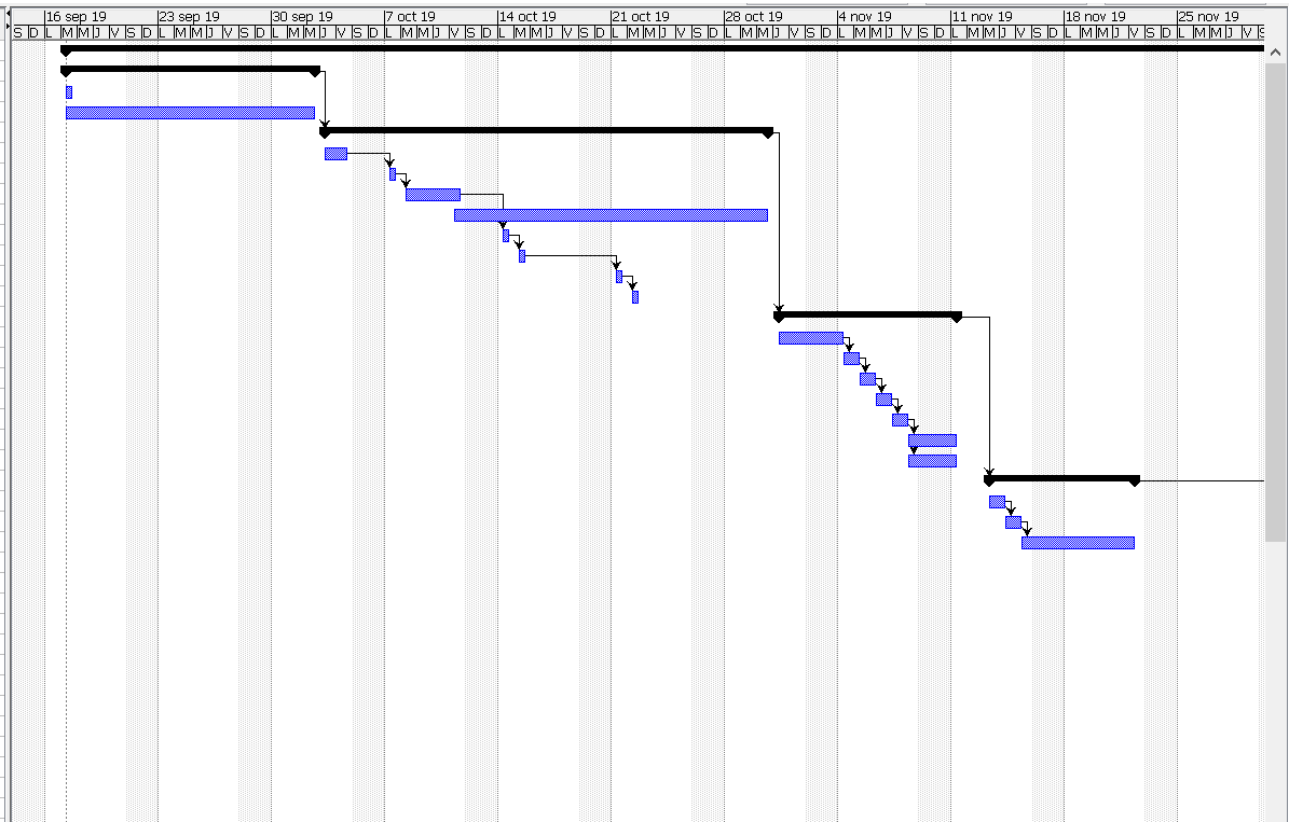
2. Planificación del proyecto

2.1. Estimación inicial de plazos del proyecto

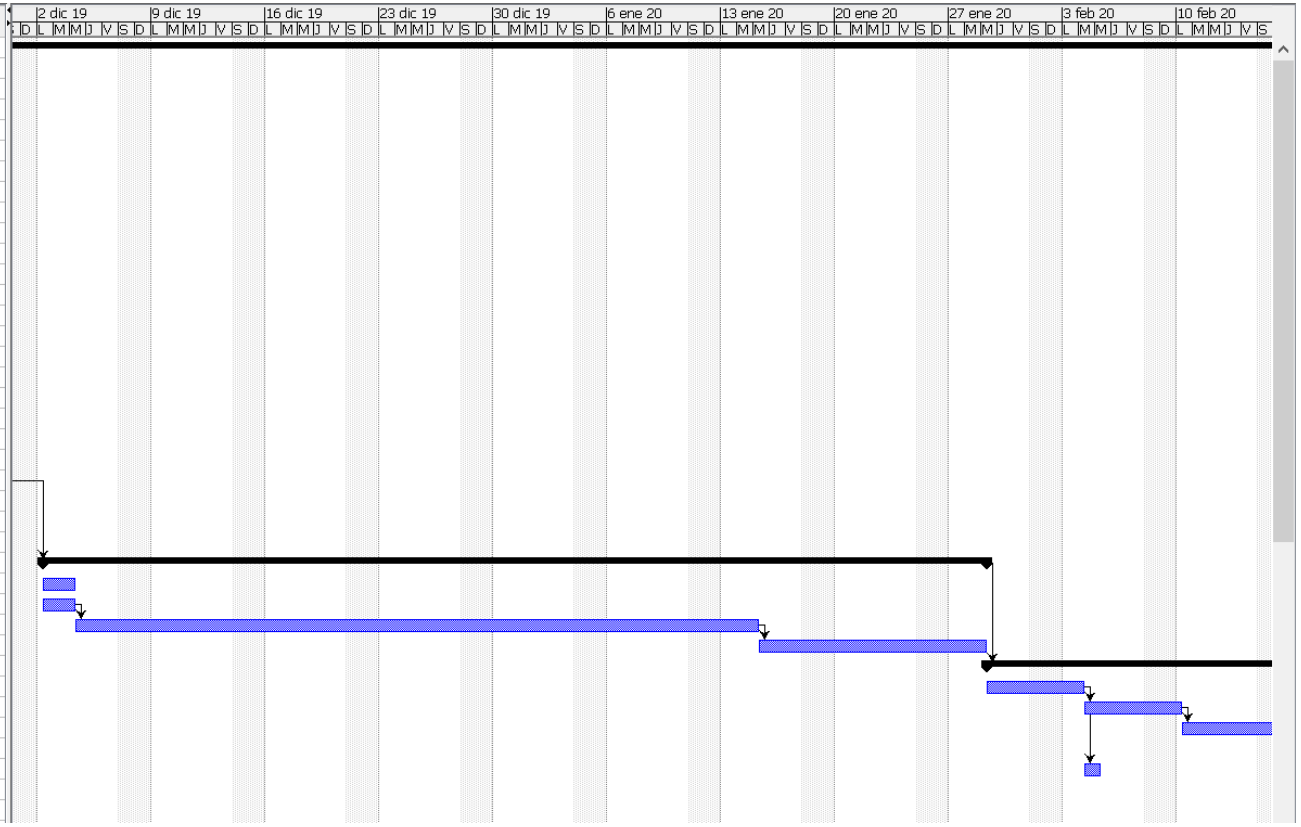
Para desarrollar el proyecto, es importante dividirlo en diferentes etapas, para ir gestionando su progreso de una forma menos compleja. Estableciendo unos tiempos óptimos en los que se desea finalizar.

Con el uso del diagrama de Gantt, nos permite observar el tiempo empleado para cada una de las fases y tareas de cada etapa. Pudiendo ajustar los tiempos en caso de que estos varíen, logrando una retroalimentación para futuros proyectos.

	Nombre	Duración	Inicio	Terminado
1	**Proyecto:Multimetro digital USB**	142,125 ...	**17/09/19 8:00**	**2/04/20 9:00**
2	**Fase 1:Especificación del proyecto**	12 days	**17/09/19 8:00**	**2/10/19 17:00**
3	Deficinición de objetivos temporales	1 day	17/09/19 8:00	17/09/19 17:00
4	Selección de componentes	12 days	17/09/19 8:00	2/10/19 17:00
5	**Fase 2:Captura esquemática**	20 days	**3/10/19 8:00**	**30/10/19 17:00**
6	Diagrama de bloques	2 days	3/10/19 8:00	4/10/19 17:00
7	Posicionado de componentes	1 day	7/10/19 8:00	7/10/19 17:00
8	Creacion de nuevos componentes	4 days	8/10/19 8:00	11/10/19 17:00
9	Documentacion esquemático. Control de cambios	14 days	11/10/19 8:00	30/10/19 17:00
10	Análisis de consumo eléctrico	1 day	14/10/19 8:00	14/10/19 17:00
11	Análisis de disipación de potencia	1 day	15/10/19 8:00	15/10/19 17:00
12	Rutado de conexiones	1 day	21/10/19 8:00	21/10/19 17:00
13	Generación de listado de materiales (BOM)	1 day	22/10/19 8:00	22/10/19 17:00
14	**Fase 3:Diseño de la placa de circuito impreso(PCB)**	7 days	**31/10/19 9:00**	**11/11/19 9:00**
15	Creacion de huellas de nuevos componentes	2 days	31/10/19 9:00	4/11/19 9:00
16	Definicion de borde y zonas de conectores	1 day	4/11/19 9:00	5/11/19 9:00
17	Posicionado de componentes	1 day	5/11/19 9:00	6/11/19 9:00
18	Rutado de conexiones	1 day	6/11/19 9:00	7/11/19 9:00
19	Definición planos de masa y otros	1 day	7/11/19 9:00	8/11/19 9:00
20	Acotaciones geométricas	1 day	8/11/19 9:00	11/11/19 9:00
21	Generación de ficheros para el fabricante (gerber)	1 day	8/11/19 9:00	11/11/19 9:00
22	**Fase 4:Fabricación de la placa de circuito impreso (PCB)**	7 days	**13/11/19 9:00**	**22/11/19 9:00**
23	Envío al fabricante	1 day	13/11/19 9:00	14/11/19 9:00
24	Recepción y revisión	1 day	14/11/19 9:00	15/11/19 9:00
25	Montaje de componentes	5 days	15/11/19 9:00	22/11/19 9:00
26	**Fase 5:Realización Firmware, Simulación y depuración**	42 days	**2/12/19 9:00**	**29/01/20 9:00**
27	Control de elementos hardware: drivers	2 days	2/12/19 9:00	4/12/19 9:00
28	Verificación Hardware	2 days	2/12/19 9:00	4/12/19 9:00
29	Programa principal. Especificaciones	30 days	4/12/19 9:00	15/01/20 9:00
30	Control de cambios Firmware	10 days	15/01/20 9:00	29/01/20 9:00
31	**Fase 6:Verificación Hardware**	20 days	**29/01/20 9:00**	**26/02/20 9:00**
32	Verificación inicial	4 days	29/01/20 9:00	4/02/20 9:00
33	Verificación Firmware	4 days	4/02/20 9:00	10/02/20 9:00
34	Resolución de problemas	10 days	10/02/20 9:00	24/02/20 9:00
35	Análisis de prestaciones	2 days	24/02/20 9:00	26/02/20 9:00
36	Montaje final. Piezas mecánicas	1 day	4/02/20 9:00	5/02/20 9:00
37	**Fase 7:Documentación**	25 days	**27/02/20 9:00**	**2/04/20 9:00**
38	Realización memoria final.	25 days	27/02/20 9:00	2/04/20 9:00



	Nombre	Duración	Inicio	Terminado
1	**Proyecto:Multimetro digital USB**	142,125 ...	**17/09/19 8:00**	**2/04/20 9:00**
2	**Fase 1:Especificación del proyecto**	12 days	**17/09/19 8:00**	**2/10/19 17:00**
3	Deficinición de objetivos temporales	1 day	17/09/19 8:00	17/09/19 17:00
4	Selección de componentes	12 days	17/09/19 8:00	2/10/19 17:00
5	**Fase 2:Captura esquemática**	20 days	**3/10/19 8:00**	**30/10/19 17:00**
6	Diagrama de bloques	2 days	3/10/19 8:00	4/10/19 17:00
7	Posicionado de componentes	1 day	7/10/19 8:00	7/10/19 17:00
8	Creacion de nuevos componentes	4 days	8/10/19 8:00	11/10/19 17:00
9	Documentacion esquemático. Control de cambios	14 days	11/10/19 8:00	30/10/19 17:00
10	Análisis de consumo eléctrico	1 day	14/10/19 8:00	14/10/19 17:00
11	Análisis de disipación de potencia	1 day	15/10/19 8:00	15/10/19 17:00
12	Rutado de conexiones	1 day	21/10/19 8:00	21/10/19 17:00
13	Generación de listado de materiales (BOM)	1 day	22/10/19 8:00	22/10/19 17:00
14	**Fase 3:Diseño de la placa de circuito impreso(PCB)**	7 days	**31/10/19 9:00**	**11/11/19 9:00**
15	Creacion de huellas de nuevos componentes	2 days	31/10/19 9:00	4/11/19 9:00
16	Definicion de borde y zonas de conectores	1 day	4/11/19 9:00	5/11/19 9:00
17	Posicionado de componentes	1 day	5/11/19 9:00	6/11/19 9:00
18	Rutado de conexiones	1 day	6/11/19 9:00	7/11/19 9:00
19	Definición planos de masa y otros	1 day	7/11/19 9:00	8/11/19 9:00
20	Acotaciones geométricas	1 day	8/11/19 9:00	11/11/19 9:00
21	Generación de ficheros para el fabricante (gerber)	1 day	8/11/19 9:00	11/11/19 9:00
22	**Fase 4:Fabricación de la placa de circuito impreso (PCB)**	7 days	**13/11/19 9:00**	**22/11/19 9:00**
23	Envío al fabricante	1 day	13/11/19 9:00	14/11/19 9:00
24	Recepción y revisión	1 day	14/11/19 9:00	15/11/19 9:00
25	Montaje de componentes	5 days	15/11/19 9:00	22/11/19 9:00
26	**Fase 5:Realización Firmware, Simulación y depuración**	42 days	**2/12/19 9:00**	**29/01/20 9:00**
27	Control de elementos hardware: drivers	2 days	2/12/19 9:00	4/12/19 9:00
28	Verificación Hardware	2 days	2/12/19 9:00	4/12/19 9:00
29	Programa principal. Especificaciones	30 days	4/12/19 9:00	15/01/20 9:00
30	Control de cambios Firmware	10 days	15/01/20 9:00	29/01/20 9:00
31	**Fase 6:Verificación Hardware**	20 days	**29/01/20 9:00**	**26/02/20 9:00**
32	Verificación inicial	4 days	29/01/20 9:00	4/02/20 9:00
33	Verificación Firmware	4 days	4/02/20 9:00	10/02/20 9:00
34	Resolución de problemas	10 days	10/02/20 9:00	24/02/20 9:00
35	Análisis de prestaciones	2 days	24/02/20 9:00	26/02/20 9:00
36	Montaje final. Piezas mecánicas	1 day	4/02/20 9:00	5/02/20 9:00
37	**Fase 7:Documentación**	25 days	**27/02/20 9:00**	**2/04/20 9:00**
38	Realización memoria final.	25 days	27/02/20 9:00	2/04/20 9:00



	Nombre	Duración	Inicio	Terminado
1	**Proyecto:Multimetro digital USB**	**142,125 ...**	**17/09/19 8:00**	**2/04/20 9:00**
2	**Fase 1:Especificación del proyecto**	**12 days**	**17/09/19 8:00**	**2/10/19 17:00**
3	Deficinición de objetivos temporales	1 day	17/09/19 8:00	17/09/19 17:00
4	Selección de componentes	12 days	17/09/19 8:00	2/10/19 17:00
5	**Fase 2:Captura esquemática**	**20 days**	**3/10/19 8:00**	**30/10/19 17:00**
6	Diagrama de bloques	2 days	3/10/19 8:00	4/10/19 17:00
7	Posicionado de componentes	1 day	7/10/19 8:00	7/10/19 17:00
8	Creacion de nuevos componentes	4 days	8/10/19 8:00	11/10/19 17:00
9	Documentacion esquemático. Control de cambios	14 days	11/10/19 8:00	30/10/19 17:00
10	Análisis de consumo eléctrico	1 day	14/10/19 8:00	14/10/19 17:00
11	Análisis de disipación de potencia	1 day	15/10/19 8:00	15/10/19 17:00
12	Rutado de conexiones	1 day	21/10/19 8:00	21/10/19 17:00
13	Generación de listado de materiales (BOM)	1 day	22/10/19 8:00	22/10/19 17:00
14	**Fase 3:Diseño de la placa de circuito impreso(PCB)**	**7 days**	**31/10/19 9:00**	**11/11/19 9:00**
15	Creacion de huellas de nuevos componentes	2 days	31/10/19 9:00	4/11/19 9:00
16	Definicion de borde y zonas de conectores	1 day	4/11/19 9:00	5/11/19 9:00
17	Posicionado de componentes	1 day	5/11/19 9:00	6/11/19 9:00
18	Rutado de conexiones	1 day	6/11/19 9:00	7/11/19 9:00
19	Definición planos de masa y otros	1 day	7/11/19 9:00	8/11/19 9:00
20	Acotaciones geométricas	1 day	8/11/19 9:00	11/11/19 9:00
21	Generación de ficheros para el fabricante (gerber)	1 day	8/11/19 9:00	11/11/19 9:00
22	**Fase 4:Fabricación de la placa de circuito impreso (PCB)**	**7 days**	**13/11/19 9:00**	**22/11/19 9:00**
23	Envío al fabricante	1 day	13/11/19 9:00	14/11/19 9:00
24	Recepción y revisión	1 day	14/11/19 9:00	15/11/19 9:00
25	Montaje de componentes	5 days	15/11/19 9:00	22/11/19 9:00
26	**Fase 5:Realización Firmware, Simulación y depuración**	**42 days**	**2/12/19 9:00**	**29/01/20 9:00**
27	Control de elementos hardware: drivers	2 days	2/12/19 9:00	4/12/19 9:00
28	Verificación Hardware	2 days	2/12/19 9:00	4/12/19 9:00
29	Programa principal. Especificaciones	30 days	4/12/19 9:00	15/01/20 9:00
30	Control de cambios Firmware	10 days	15/01/20 9:00	29/01/20 9:00
31	**Fase 6:Verificación Hardware**	**20 days**	**29/01/20 9:00**	**26/02/20 9:00**
32	Verificación inicial	4 days	29/01/20 9:00	4/02/20 9:00
33	Verificación Firmware	4 days	4/02/20 9:00	10/02/20 9:00
34	Resolución de problemas	10 days	10/02/20 9:00	24/02/20 9:00
35	Análisis de prestaciones	2 days	24/02/20 9:00	26/02/20 9:00
36	Montaje final. Piezas mecánicas	1 day	4/02/20 9:00	5/02/20 9:00
37	**Fase 7:Documentación**	**25 days**	**27/02/20 9:00**	**2/04/20 9:00**
38	Realización memoria final.	25 days	27/02/20 9:00	2/04/20 9:00

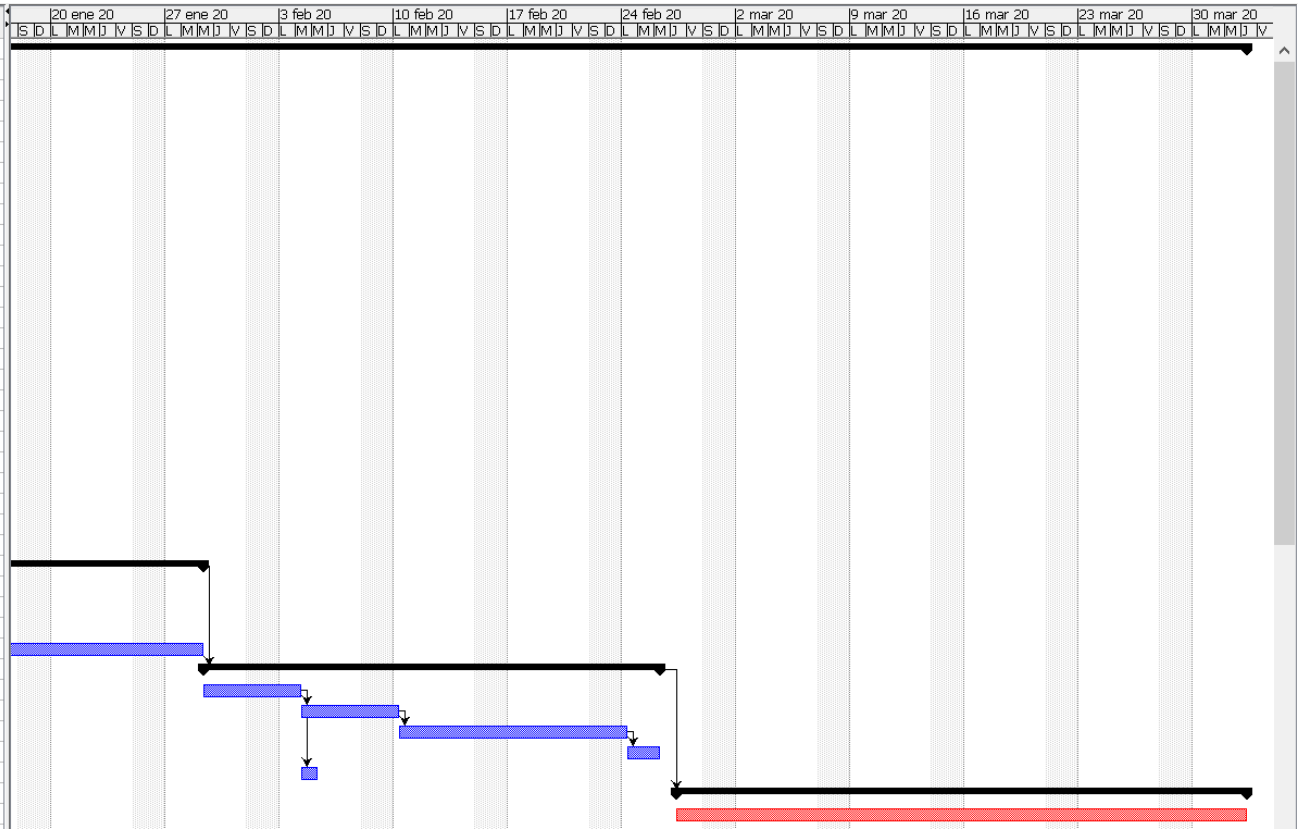


Figura 4: Diagrama de Gantt

2.2. Selección de componentes.

A la hora de diseñar un dispositivo electrónico es muy importante tener en cuenta todo aquello que lo compone. Se analiza cada componente por separado, teniendo en cuenta sus características tanto físicas como técnicas, además de su precio, habiendo intenciones o no de comercialización. De esta manera, la integración de todos ellos para formar un multímetro digital nos dará las siguientes especificaciones técnicas generales:

- Rango de medición / gráficas de voltaje: 3.51-25.30V
- Resolución de medición de voltaje / resolución de gráficos: 0.01V
- Precisión de medición de voltaje: ± 1.68%
- Rango de medición / gráficas de corriente: 0.000-5.926A
- Resolución de medición de corriente / resolución de gráficos: 0.001A
- Precisión de medición de corriente: ± 3.06%
- Rango de acumulación de capacidad: 0-99999mAh
- Rango de acumulación de energía: 0-99999mWh
- Rango de impedancia de carga: 1.5-9999.9Ω
- Rango de potencia: 0-999.99W
- Rango de medición de tiempo: 0-99h59min59s
- Rango de temperatura: -10 ℃ - 100 ℃ / 14 ℉ - 212 ℉
- Resolución de medición de temperatura: 0.1℃/0.2℉
- Error de medición de temperatura: ± 2.2 ℃ / ± 4 ℉
- Frecuencia de actualización: 2Hz

Microcontrolador PIC24FJ256GA702-I/SS

Se trata del microcontrolador que se ha elegido para el desarrollo del multímetro digital. Adquirido a través del distribuidor de componentes electrónicos Farnell y fabricado por Microchip Technology Inc, del cual obtenemos la herramienta Mplab X IDE como entorno de desarrollo para su programación.

Como vemos en la figura 5 se ha optado por un encapsulado SSOP para mayor facilidad de soldadura. Cuenta con 28 pines, de los cuales solo necesitamos un total de 26 pines para llevar a cabo el funcionamiento del dispositivo, ahorrando espacio y costes con relación a los PICs de la misma familia de 44 y 48 pines.

Figura 5: Microcontrolador PIC24FJ256GA702-I/SS

La elección de un PIC24F en vez de cualquier otro PIC o incluso un dsPIC, es debido a que este componente tiene un coste y un uso de potencia inferior en comparación al resto, lo que supone un menor consumo. Existe el inconveniente de que no tenemos tanta velocidad de procesamiento como los dsPIC, consiguiendo mediante la implementación de un cristal de cuarzo una **frecuencia máxima de 32 MHz** que se traduce en **16 MIPS**. Un valor suficiente, ya que no necesitamos una velocidad elevada para el procesamiento y adquisición de datos, así como para la transmisión de datos a la pantalla.

Con la finalidad de mostrar por pantalla gráficas donde se observen las variaciones de tensión y corriente, es necesario almacenar los datos muestreados. Partimos de que los valores que se almacenan tienen un tamaño de 2 bytes. Primero se almacena un valor de tensión y otro de corriente en la memoria RAM por cada segundo que transcurre, siendo un total de 7200 valores en 1 hora, que ocupan un tamaño de 14400 Bytes. Contando que tenemos **16 kB de RAM**, aún tenemos espacio suficiente.

En segundo lugar, tras completar la hora, se toma de cada constante, un valor máximo y mínimo de intervalos de 4 segundos y se almacenan en la memoria Flash siendo 3600 valores, que ocupan 7200 bytes. Si tenemos presente que queremos poder mostrar hasta 18 horas con el fin de poder realizar análisis más prolongados, ocuparía un total de 129600 Bytes. Por este motivo elegimos este microcontrolador que cuenta con una memoria de **256 kB de Flash**, ya que también debemos tener en cuenta el espacio ocupado por el firmware.

La elección de un PIC se centra sobre todo en el gran abanico de posibilidades, pues como sabemos, cuenta con una amplia gama de características hardware incorporadas. Accesibles a través de una gran cantidad de pines de propósito general, de las cuales destacamos aquellas que se han utilizado y su propósito:

- **Temporizador:** Cuya finalidad es el uso periódico de interrupciones temporales de manera precisa. Proporcionando tiempos exactos, con los que medir la cantidad de tiempo que transcurre, así como activar el muestreo de señales, como servir de base para la configuración de una señal PWM.
- **Comparador:** Dentro de las posibilidades que nos ofrece, nos interesa el poder generar una señal **PWM (*Pulse Width Modulation*)** con el fin de controlar el brillo de la pantalla TFT, que se modifica según el nivel de tensión aplicado en el pin LED.
- **Protocolo de comunicación:** La utilización de una pantalla TFT-LCD como un periférico, necesita de la comunicación con el microcontrolador para el intercambio de datos con el fin de representar aquello que se desee. Al constar de una resolución de 240*240 pixeles, necesitamos enviar la información en el menor tiempo posible con el fin de actualizar la pantalla y que no se aprecie el proceso de dibujado.
  Este microcontrolador dispone de varios protocolos de comunicación, de todos ellos elegimos **SPI (*Serial Peripherical Interface*)** por tener una mayor velocidad de comunicación, a pesar de que se necesita un mayor número de pines en comparación, no supone un problema.
- **Conversor analógico digital (CAD):** La finalidad del proyecto es realizar medidas y mostrarlas por pantalla, para ello necesitamos muestrearlas y convertirlas a valores digitales que podamos procesar.
  El CAD tiene la opción de 10 y 12 bits de resolución de conversión, y de acuerdo con las características que solicitamos, necesitamos una resolución mínima de 0.01V y de

0.001A. Teniendo en cuenta los fondos de escala de cada valor deseado, y como veremos más adelante en la captura esquemática, es necesaria una resolución mínima de 12 bits.

Con respecto al resto de características obtenidas del datasheet [7] que nos ofrece el fabricante del producto, podemos destacar:

- Rango de voltaje de alimentación: 2V a 3.6V. Eligiendo un valor de 3.3V, siendo el valor recomendado y un valor típico de operación para el resto de los componentes.
- Temperatura de operación: - 40°C a + 85°C.
- Dimensiones: 10.22 x 7.8 x 1.75mm.
- Consumo máximo de corriente: 7.7 mA a 3.3V.
- Precio unitario bajo: 1.68€.

Sensor de temperatura LMT85DCKT

Para mantener un control de la temperatura, recurrimos al circuito integrado **LMT85DCKT**, un sensor de temperatura tipo CMOS adquirido a través del distribuidor de componentes electrónicos Farnell y fabricado por Texas Instruments con encapsulado SC-70.

Figura 6: Sensor de temperatura LMT85DCKT

Dispone de las siguientes características obtenidas del datasheet [8] que nos ofrece el fabricante del producto y podemos destacar:

- Rango de voltaje de alimentación: 1.8V a 5.5V. Eligiendo un valor de 3.3V, siendo el valor recomendado y un valor típico de operación para el resto de los componentes.
- Salida de tensión analógica inversamente proporcional a la temperatura.
- Sensibilidad promedio: $-8.2\,\frac{mV}{°C}$
- Rango de temperatura: - 50C a + 150°C.
- Temperatura de operación: - 50C a + 150°C.
- Dimensiones pequeñas: 2.15 x 2.40 x 1.10mm.
- Resolución: 0.1°C
- Consumo máximo de corriente: 9 µA a 3.3V.
- Precio unitario bajo: 0.715€.
- Alternativa rentable a los termistores.

El funcionamiento del **LMT85DCKT** se basa en la tecnología CMOS con salida analógica. El elemento sensor de temperatura está compuesto por un transistor con unión emisor-base, polarizado en directa por una fuente de corriente. Esta configuración le hace actuar como un diodo, cuya tensión es inversamente proporcional a la temperatura y varía de forma lineal aproximadamente.

Conectado a un amplificador buffer, acondiciona el valor de tensión del diodo a la salida analógica al pin OUT, proporcionando una baja impedancia en la salida como vemos en el diagrama de bloques (Figura 7).

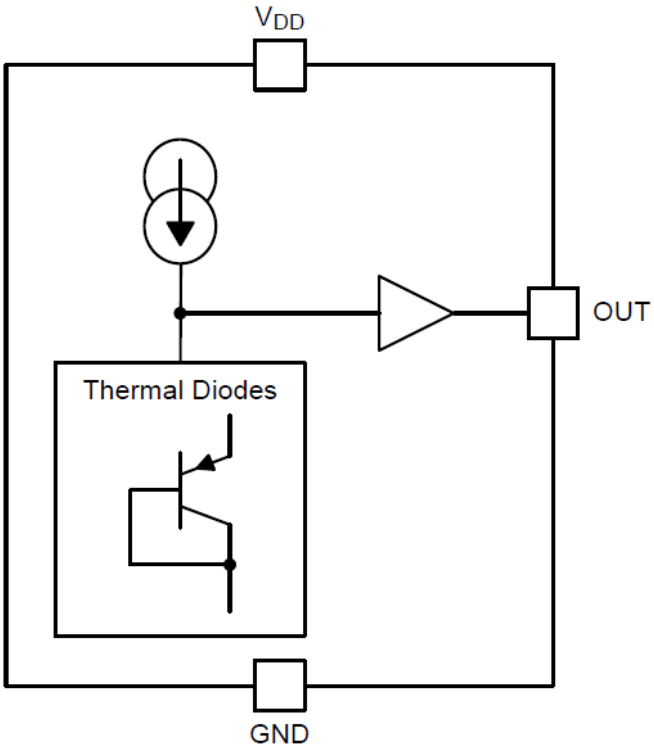


Figura 7: Diagrama de bloques de LMT85DCKT

El fabricante nos proporciona una función de transferencia [8] deducida a partir de una tabla de valores de tensión de salida con respecto a la temperatura, obtenidos en el rango de operación de -50°C a 150°C.

$$T(°C) = \frac{8.194 - \sqrt{(-8.194)^2 + 4 * 0.00262 * (1324 - V_{temp}(mV))}}{2 * 0.00262} + 30 \quad (1)$$

Siendo:

- $V_{temp}(mV)$: Voltaje de salida.
- T(°C): Temperatura de entrada.

Por otro lado, se nos proporciona otra función con menor precisión.

$$V_{temp}(mV) - V_1(mV) = \frac{V_2(mV) - V_1(mV)}{T_2(°C) - T_1(°C)} * (T(°C) - T_1(°C)) \quad (2)$$

Donde:

- $V_{temp}(mV)$: Voltaje de salida.
- T(°C): Temperatura de entrada.
- T_1 y V_1 son las coordenadas de la temperatura más baja.
- T_2 y V_2 son las coordenadas de la temperatura más alta.

Contando con que los valores de temperatura que se lleguen a medir acorde a la finalidad del dispositivo se encuentren entre -10°C y 100°C, el rango de tensión de salida respecto a la temperatura se comprenderá entre 737mV a 100°C y 1648mV a -10°C.

$$T(°\mathrm{C}) = \frac{V_{temp} - 1648mV}{-8.281\frac{mV}{°\mathrm{C}}} - 10°\mathrm{C} \quad (2)$$

De esta manera tenemos la salida de la ecuación 1 en rojo y la salida de la ecuación 2 en azul acotada a ese rango como vemos en la figura 8.

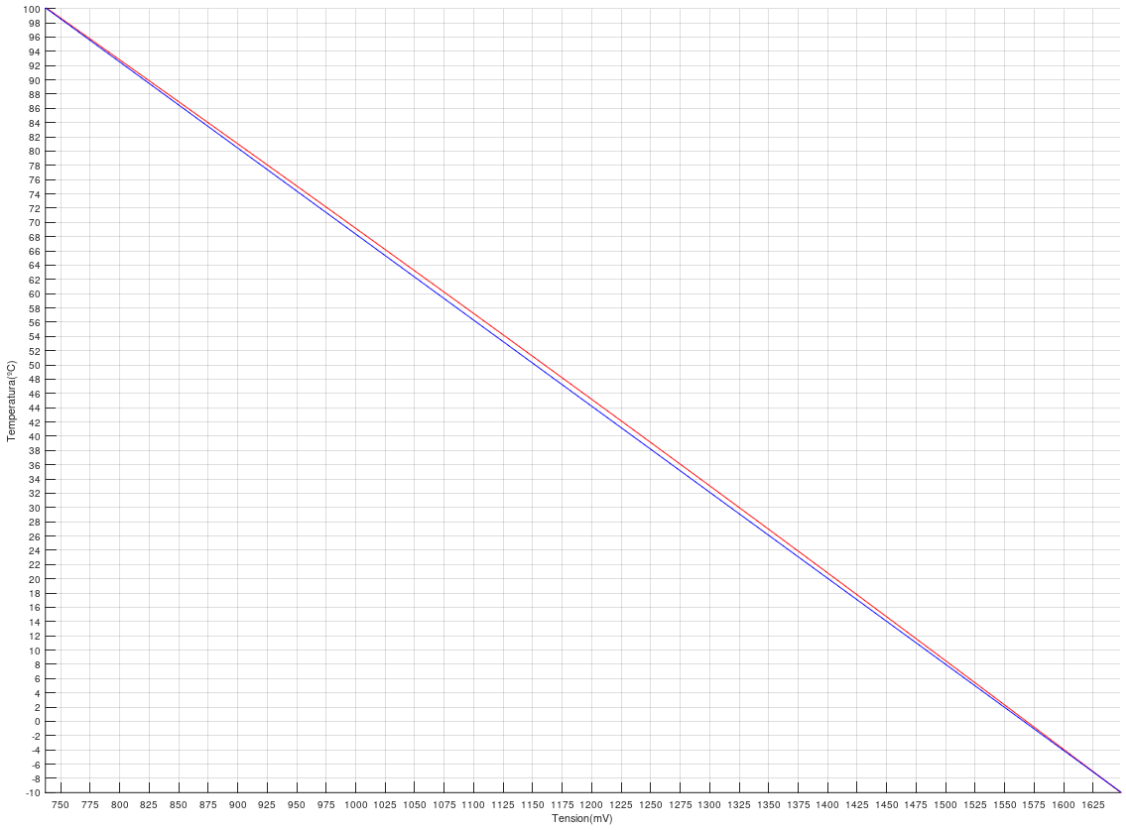


Figura 8: Curvas características de LMT85DCKT

En la salida existe una diferencia con valores próximos a 1°C entre la ecuación 1 y la 2 en los puntos intermedios, que va disminuyendo hacia los extremos debido a la ligera forma parabólica de la ecuación 1. Teniendo esto en como base, se aplicará la ecuación 1 mediante software, ya que como hemos visto, cuenta con la mayor precisión de ambas ecuaciones.

En la figura 8 se puede ver como a medida que la tensión de salida disminuye, la temperatura aumenta, variando de manera inversamente proporcional. Y puesto que utilizaremos un CAD de 12 bits, podremos captar variaciones de 0.805mV. Sabiendo que tenemos una sensibilidad aproximada de $-8.2\,\frac{mV}{°\mathrm{C}}$, calculamos la resolución de temperatura que tenemos en relación con la variación mínima de voltaje que somos capaces de detectar, obteniendo:

Resolución térmica → $\frac{0.805mV}{8.2\frac{mV}{°C}} = 0.09°C$

Puesto que nuestra intención es mostrar un máximo de decimas de grado por pantalla, conseguiremos una resolución de 0.1°C. A pesar de ello, hay que tener en cuenta el error de temperatura que puede haber, relacionado con la fabricación del integrado. Sacando la relación entre la temperatura de entrada y unos valores máximos de error sacados del datasheet [8] y visualizados en la figura 9. Tendremos un error de medición máximo de ±2.2°C centrándonos en nuestro rango de operación.

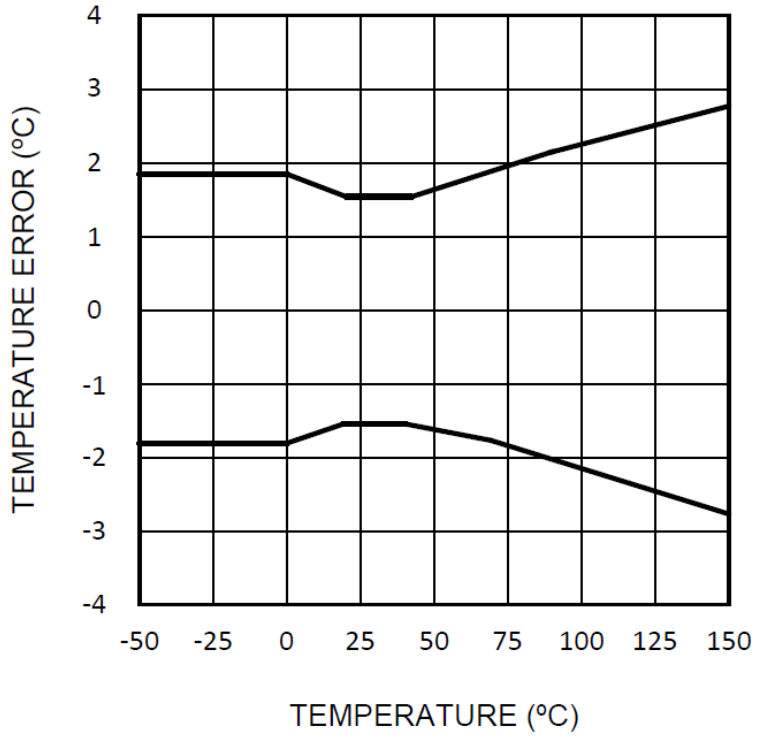


Figura 9: Temperatura vs Error de Temperatura (LMT85DCKT)

Regulador de tensión RT9058-33GV

Puesto que nuestro dispositivo se trata de un multímetro digital, es importante tener en cuenta que se van a producir variaciones de voltaje en la entrada. Por ello, recurrimos al regulador lineal de voltaje (LDO) **RT9058-33GV**, que presenta una baja caída de voltaje y proporciona una salida estable a 3.3V para valores de entrada entre 3.5V y 36V.

Adquirido a un valor de 0.746€, para la elección de este componente también se ha tenido en cuenta su rango de corriente de salida de 100mA, valores suficientes para alimentar todos los componentes del dispositivo como se verá en el **análisis eléctrico**.

Otro punto importante es la elección de la tensión de 3.3V de salida. Un valor en común que se puede aplicar a los componentes elegidos, y un valor estándar recomendable para todos ellos según sus características. Reduciendo así toda la tensión de alimentación a un único componente.

Figura 10: Regulador de tensión RT9058-33GV

Amplificador operacional OPA348AIDBVTG4

A la hora de realizar un control de la corriente que circula a través del dispositivo, es fundamental el circuito de acondicionamiento de las muestras capturadas por el CAD. Como una de las partes principales de ese circuito tenemos el amplificador operacional **OPA348AIDBVTG4** que, mediante una topología de amplificador diferencial, aplica una ganancia en tensión para ajustar la medida dentro del fondo de escala (3.3V) del CAD.

Este componente se caracteriza por las siguientes características obtenidas del datasheet [9] que nos ofrece el fabricante y destacamos:

- Rango de voltaje de alimentación: 2.1V a 5.5V. Eligiendo 3.3V siendo el valor recomendado y el valor en común de todos los componentes.
- Ancho de banda de ganancia: 1MHz. Muy superior al valor necesitado, pues la frecuencia de actualización de la pantalla es de 2Hz.
- Velocidad de respuesta: 0.5V/µs. Valor suficiente para acondicionar la señal según la ganancia configurada, en un tiempo muy inferior al de adquisición de muestras por el CAD.
- Temperatura de operación: - 40°C a +125°C.
- Dimensiones pequeñas: 3.05 x 3.00 x 1.45mm.
- Corriente de reposo máxima (I_Q): 75µA a 3.3V.
- Corriente de polarización máxima (I_B): ±10pA a 3.3V.
- Precio unitario bajo: 0.561€

Figura 11: Amplificador operacional OPA348AIDBVTG4

Modulo TFT de 1.33 pulgadas

Para poder visualizar todos los valores medidos por el dispositivo, recurrimos a un módulo compuesto por una **pantalla TFT-LCD IPS** junto con un slot para una tarjeta microSD, adquirido a través de Aliexpress.

La pantalla TFT-LCD IPS supone una mejora de la pantalla de cristal líquido (LCD) con el uso de la tecnología de transistor de película delgada (TFT) consiguiendo una mejora en la reducción del consumo eléctrico gracias a la matriz activa de transistores, en la que cada píxel está asociado a un transistor. Y la tecnología IPS permite ángulos de visión horizontales y verticales de 178 grados, y colores más intensos siendo visualmente más agradable.

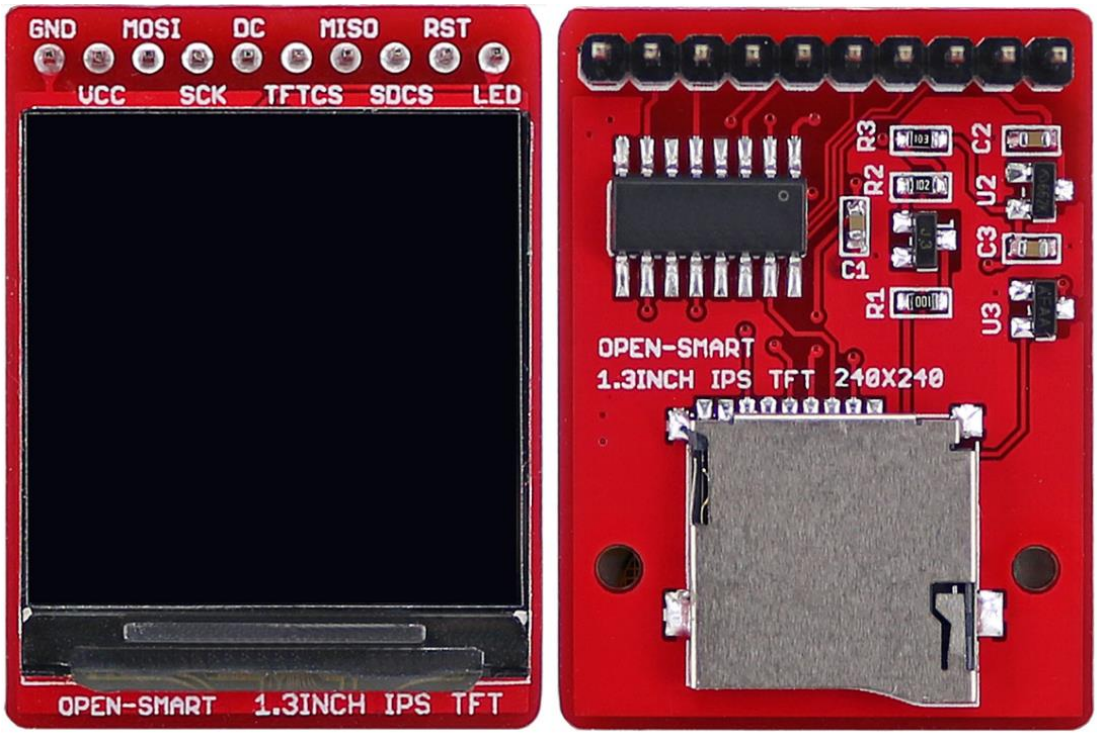


Figura 12: Modulo TFT-LCD IPS 1.33'

Caracterizado por las siguientes características principales obtenidas del fabricante [10]:

- Rango de voltaje de alimentación: 3.3V a 5V. Eligiendo 3.3V siendo el valor en común de todos los componentes.
- Área activa: 23.4 x 23.4mm.
- Resolución: 240*240 pixeles, a un máximo de 16 bits de color. Proporcionando una visualización bien detallada para su tamaño.
- Tamaño de pantalla: 1.33 pulgadas.
- Consumo máximo de corriente: 80mA.
- Protocolo de comunicación: SPI.
- Controlador de la pantalla: ST7789VW.
- Dimensiones del módulo: 37.5x 28 x 0.5mm.
- Precio unitario bajo: 2.88€

Es importante darnos cuenta de que la resolución va íntimamente relacionada con el tiempo necesario de procesamiento para dibujar en la misma. Teniendo en cuenta que solo se

dibujará una primera vez por pantalla, y luego solo se modificaran aquellos valores que cambien, no estaremos dibujando constantemente. Con esto conseguimos reducir notablemente la cantidad de pixeles a modificar por cada refresco, manteniendo una alta resolución sin llegar a notar el dibujado. Además, de entre todas las resoluciones, la elección de un mismo tamaño de alto y ancho nos facilita la representación de las gráficas en diferentes ángulos múltiplos de 90 grados.

La elección de implementar un módulo supone que ya contenga los circuitos que se encarguen de la gestión de la pantalla, así como su controlador. En este caso se trata del **ST7789VW** [11], con el que nos conectamos directamente con nuestro microcontrolador mediante el protocolo SPI.

De la misma forma, como vemos en la figura 12 tenemos los pines TFTCS y SDCS con los que seleccionar la transferencia de datos con la pantalla o la tarjeta de memoria microSD.

Teniendo en cuenta el protocolo SPI, el microcontrolador actúa como maestro, mientras que la pantalla y la tarjeta de memoria actuarán como esclavos.

Como vemos en la figura 13, en este protocolo tenemos un maestro al que se le puede conectar uno o varios esclavos, que enviarán y recibirán información de manera simultánea utilizando diferentes líneas de manera síncrona.

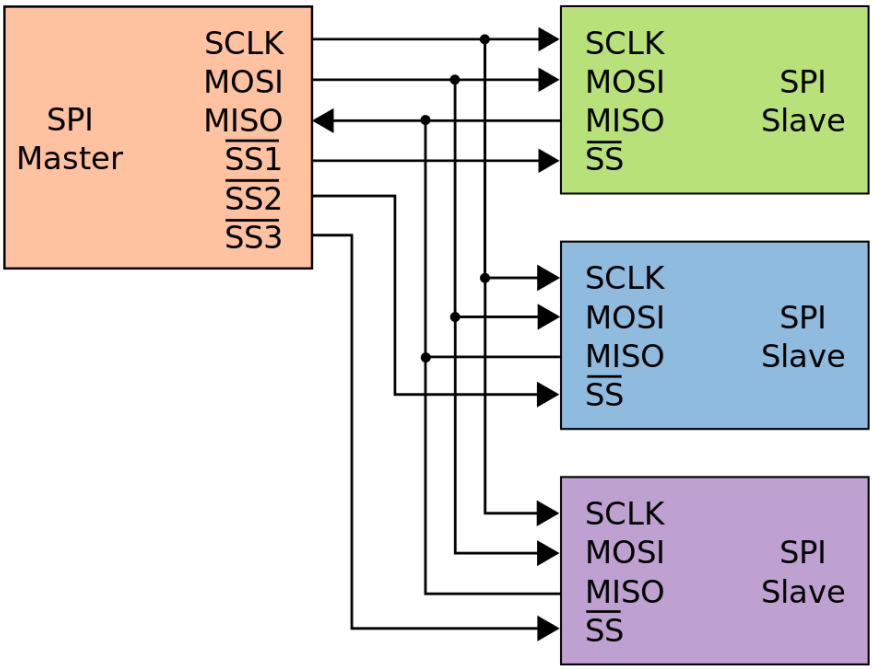


Figura 13: Diagrama de bloques Protocolo SPI

Consta de cuatro líneas lógicas mediante las cuales se realiza todo el proceso:

- MOSI (Master Out Slave In): Línea de salida de datos provenientes del maestro hacia el esclavo.
- MISO (Master In Slave Out): Línea de salida de datos provenientes del esclavo hacia el maestro.
- SCLK o SCK(Clock): Línea en la cual el maestro envía la señal de reloj para sincronizar los dispositivos en la transferencia de datos.
- SS (Slave Select) ó CS (Chip Select): Línea que selecciona y activa un esclavo con el que se va a producir la comunicación con el maestro.

Por último, tenemos un pin denominado LED, que, dependiendo de la tensión aplicada, se podemos controlar el brillo de la pantalla TFT. Para ello utilizamos la señal PWM, caracterizada por ser una señal cuadrada a la que podemos variar su ciclo de trabajo. Teniendo en cuenta que se obtiene un valor de voltaje medio de una señal cuadrada, variante dependiendo del ciclo de trabajo y de su frecuencia, solo es necesario ajustar estos valores para modificar la tensión media y por consiguiente el nivel de brillo.

Es esencial que la frecuencia no sea muy elevada, pues los cambios en el ciclo de trabajo no llegarían a apreciarse, al igual que si es demasiado pequeña, se podrían apreciar variaciones de voltaje, en vez de un valor medio más estable, provocando parpadeos de la luz de fondo de la pantalla muy molestos.

Botón B3F-3150

Contando con varias pantallas de información y opciones que se pueden modificar y ajustar a nuestro gusto, es necesaria la implementación de un par de botones como método para la interacción con el software. Por conseguir una mayor ergonomía en el dispositivo, se eligen los botones **B3F-3150** [12] con un ángulo de 90 grados con respecto al anclaje del componente.

Figura 14: Botón B3F-3150

Conectores USB tipo A 2.0

Como método de entrada y salida del dispositivo, se opta por la conexión USB tipo A 2.0, siendo el estándar más generalizado en el uso medio de las conexiones entre dispositivos de carácter más comercial. Para ello elegimos los conectores con mejor calidad precio, donde encontramos el USB macho **MC32603** [13] como método de entrada y el USB hembra **MC32593** [14] como método de salida. A pesar de contar con una corriente nominal de 1.5A, pueden soportar perfectamente valores superiores sin dañar los componentes.

Figura 15: Conectores USB tipo A

Resistencias y condensadores

La selección de las distintas resistencias y condensadores se han adquirido a través del distribuidor Farnell, con los valores según su finalidad como veremos más adelante. Adquiriéndolos con una tolerancia máxima de un 1% con el fin de introducir el menor error posible en la adquisición y acondicionamiento de los datos, así como el funcionamiento correcto del dispositivo.

Cristal de cuarzo

Con el fin de alcanzar la máxima frecuencia de procesamiento en el microcontrolador, es necesaria la utilización de un cristal de 8MHz. Así como nos proporciona una señal de reloj más estable, permitiéndonos un control del tiempo más preciso. Se elige el cristal **QCL8.00000F18B23B** [15] con patillas de inserción.

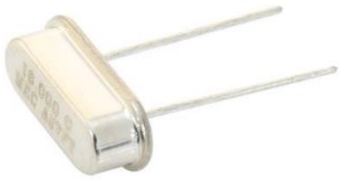


Figura 16: Cristal QCL8.00000F18B23B

3. Hardware

3.1. Captura esquemática

Una vez tenemos toda la elección de componentes, procedemos con el desarrollo hardware del dispositivo. Para ello mediante una serie de borradores iniciales en Proteus 8 Professional, definimos la conexión de los componentes según sus especificaciones y su función, llegando a la captura esquemática final.

Para el desarrollo de esta es necesario que los componentes que vamos a utilizar se encuentren en las librerías de Proteus 8 Professional, o en caso contrario, procederemos a su diseño de la forma esquemática y de su huella correspondiente. En nuestro caso ha sido necesario diseñar todos los componentes del circuito, a excepción de las resistencias y condensadores.

Para ello, primero se procede creando un diseño lo más intuitivo posible para reconocer el componente acorde a sus características físicas, junto con la cantidad de pines de los que está compuesto. El siguiente paso será crear el componente, al que le añadiremos una serie de propiedades que definirán la forma del empaquetado, el precio, la referencia del fabricante, etc. Además de asociarle su huella correspondiente si ya existe, o si la hemos creado nosotros mismos. Una vez finalizamos este último paso, es muy importante poner especial atención al relacionar cada pin con su patilla correspondiente con el fin de evitar errores de diseño en la placa de circuito impreso.

Tras crear todos los componentes, se completa el circuito conectándolos acorde a su configuración. Y es necesario realizar un análisis eléctrico y térmico, que quedará reflejado en la última página del esquemático.

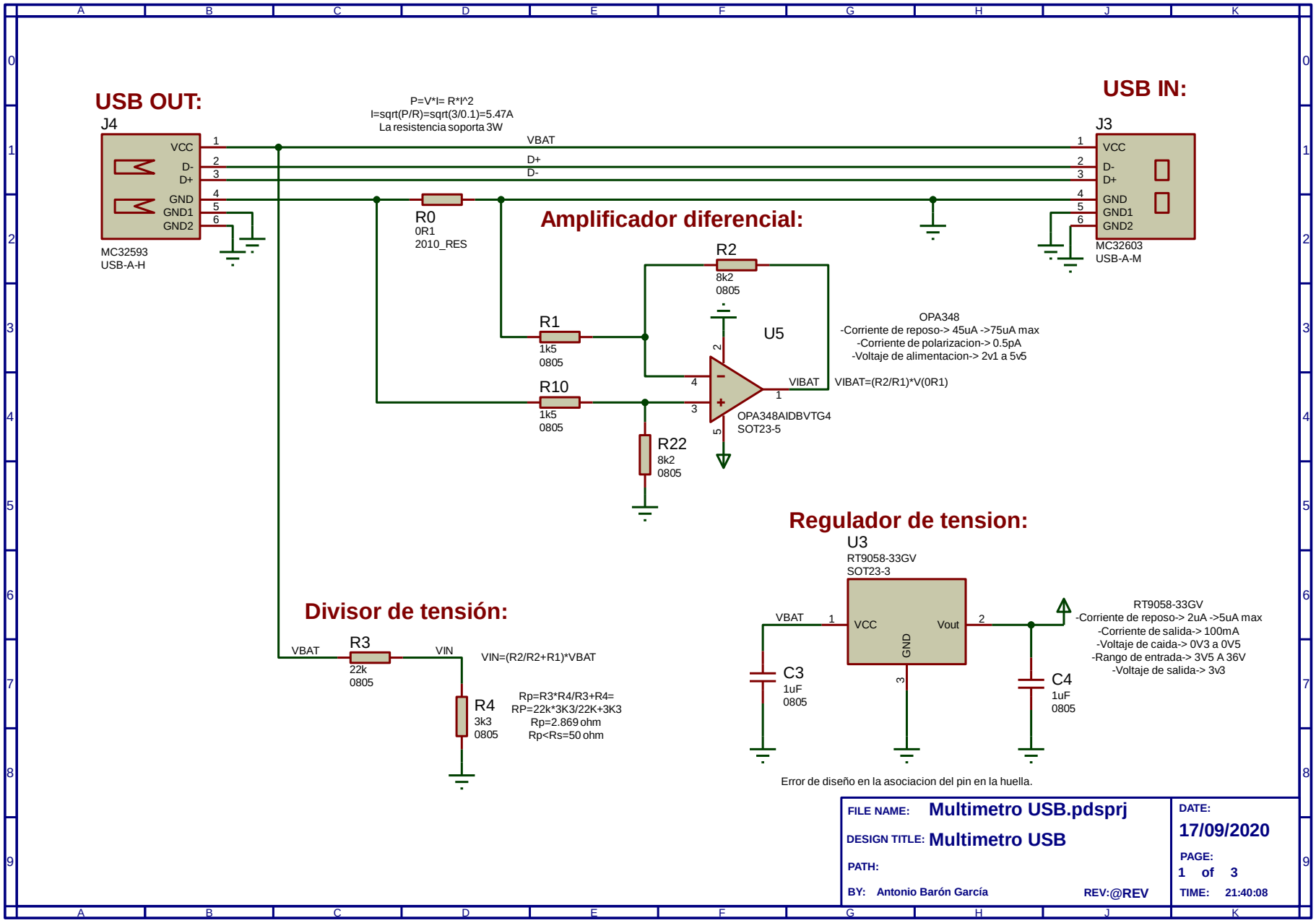


Figura 17: Captura esquemática 1

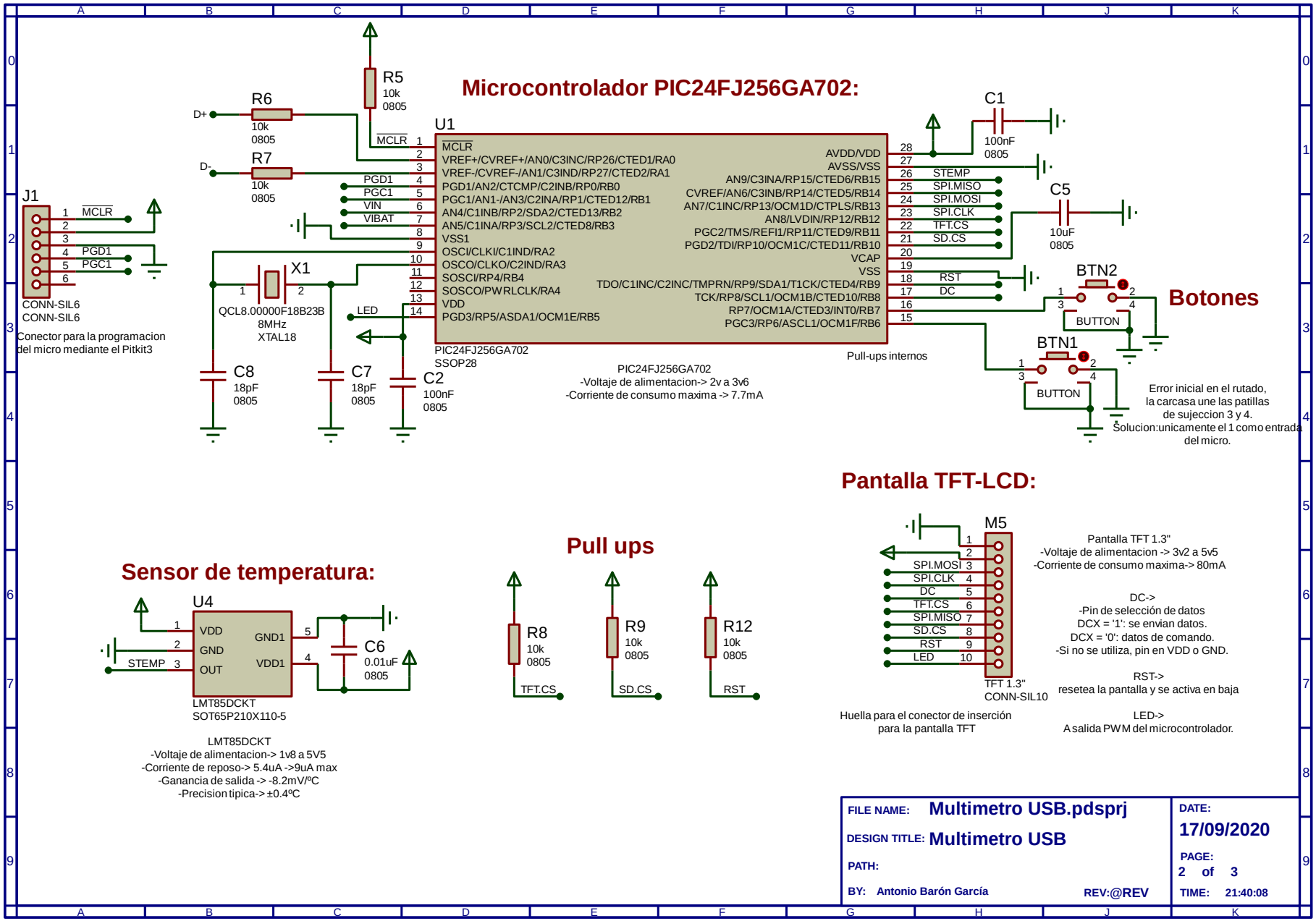


Figura 18: Captura esquemática 2

Analisis Eléctrico y Térmico

Dispositivo	Alimentación:	Consumo de corriente maximo:	Consumo de potencia: P=V*I	Potencia disipada maxima: PD(MAX) = (TJ(MAX) - TA) / ThetaJA
OPA348AIDBVTG4	3V3	75uA + 0.5pA	P=3V3 * (75uA+0.5pA) = 0.247mW	PD(MAX)=(150-25)/(229ºC/W)= 0.545W
PIC24FJ256GA702	3V3	7.7mA	P=3V3 * 7.7mA = 25.41mW	PD(MAX)=(85-25)/(39ºC/W)= 1.538W
LMT85DCKT	3V3	9uA	P=3V3 * 9uA = 0.0297mW	PD(MAX)=(150-25)/(275ºC/W)= 0.45W
TFT 1.3"	3V3	80mA	P=3V3 * 80mA = 0.264W	
RT9058-33GV	3V5 a 24V	5uA =87.789mA	P=(Vin-Vout) * I P=(25.3V - 3V3) * 87.789mA = 1.91W P=(7v98 - 3V3) * 87.789mA = 0.41W P=(3V5 - 3V3) * 87.789mA = 17mW	PD(MAX)=(125-25)/(243.3ºC/W)= 0.411W

De todo esto deducimos que vamos a necesitar disipar el calor que se producirá en el regulador cuando la entrada del conector usb supere los 7v98 y la TFT 1.3" este consumiendo el maximo de 80mA.

FILE NAME: Multimetro USB.pdsprj	DATE: 17/09/2020
DESIGN TITLE: Multimetro USB	PAGE: 3 of 3
PATH:	
BY: Antonio Barón García REV:@REV	TIME: 21:40:08

Figura 19: Captura esquemática 3

En este podemos diferenciar una serie de bloques principales que analizaremos más en detalle, para conocer el motivo de su configuración y su finalidad, dejando aquellos que ya han tenido una explicación suficiente en la selección de componentes y podemos obviar solo con verlos.

Microcontrolador:

Como parte fundamental del dispositivo, se encarga de la gestión de toda la información, así como de los componentes. Una de las labores principales es el uso del convertidor analógico digital (CAD), que, muestreando las salidas provenientes del sensor de temperatura, divisor de tensión y amplificador diferencial, las convierte a un valor digital para su procesamiento. Una vez se analiza, mediante el protocolo SPI se comunica con la pantalla TFT-LCD para la representación de toda la información.

Para conseguir todo esto es necesaria la configuración del microcontrolador que se verá detallada en el código, así como la configuración de sus pines.

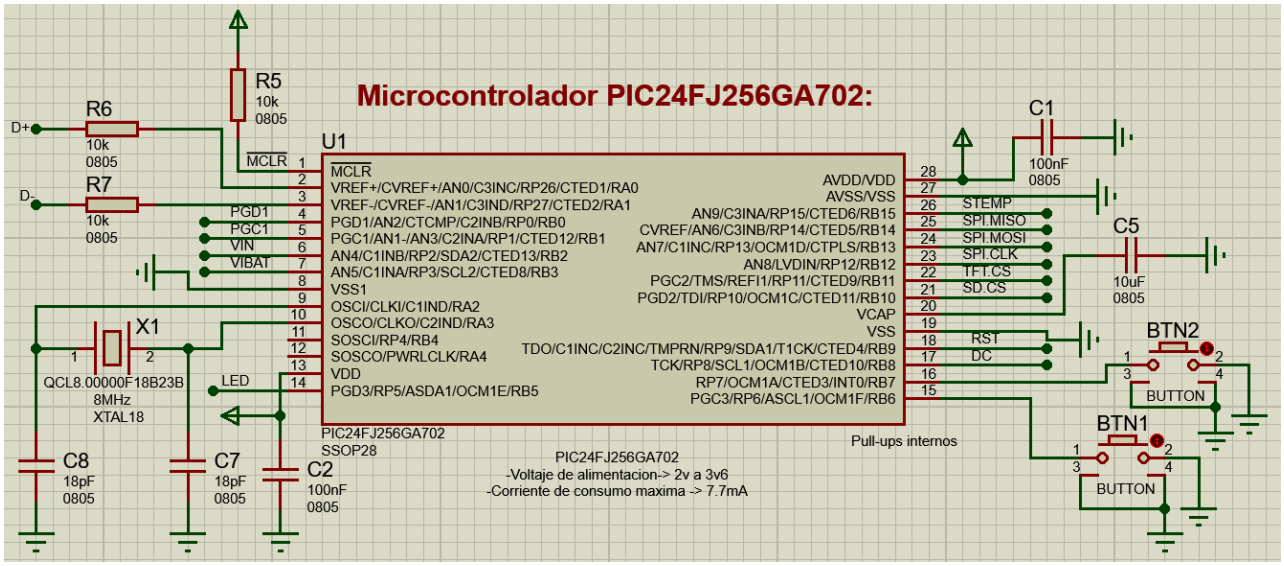


Figura 20: Microcontrolador conexionado

Como vemos en la figura 20 podemos observar la distribución de los pines del microcontrolador, junto con una descripción más detallada de la función de cada pin. Con esto de referencia, los pines se han utilizado de la siguiente manera:

- Los **pines 1, 4 y 5** como método de acceso para su programación mediante el hardware Pickit 3. Configurados como $\overline{MCLR}$, PGD1 Y PGC1 respectivamente. Puesto que $\overline{MCLR}$ se activa en baja, es necesario mantener el pin con una tensión en alta. Para ello se conecta el pin a alimentación a través de una resistencia elevada, manteniendo la tensión alta y limitando el paso de corriente al mínimo para cuando se active el pin en baja, siendo esto conocido como un Pull up.
- El **pin 2 y 3** configurados como entradas analógicas conectados al CAD. Están conectados a la línea de datos D+ y D- del conector USB respectivamente. Se utiliza una resistencia elevada entre ambas conexiones para provocar el menor consumo de corriente durante su muestreo y la menor caída de tensión evitando interferencias en la transmisión de información por ese canal.

- El **pin 6** configurado como entrada analógica conectado al CAD. Analiza la tensión de salida proveniente del pin VDD del USB, acondicionada por el divisor de tensión.
- El **pin 7** configurado como entrada analógica conectado al CAD. Analiza la tensión de caída en la resistencia de precisión R0, acondicionada por el amplificador diferencial para calcular la corriente que circula a través del dispositivo.
- El **pin 8, 19 y 27** conectados a tierra.
- El **pin 9 y 10** configurados como oscilador externo. Conectados al cristal de cuarzo que proporciona la señal de reloj.
- El **pin 13 y 28** conectados a alimentación de 3.3V con condensador de desacoplo.
- El **pin 14** configurado como salida del comparador. Conectado al pin LED de la pantalla TFT-LCD, genera una señal PWM para controlar el nivel de brillo.
- El **pin 15 y 16** configurados como entradas digitales con Pull up interno. Conectado a los botones 1 y 2 respectivamente para el control del dispositivo. Se activan en baja.
- El **pin 17** configurados como salida digital en alta. Conectado al pin DC de la pantalla TFT. Se pone en baja para enviar comandos al controlador de la pantalla. Si se mantiene en alta los bits recibidos se interpretan como datos.
- El **pin 18** configurados como salida digital. Conectado a RST de la pantalla TFT mediante pull up externo. Se activa en baja para realizar un reset mediante hardware.
- El **pin 20** es un regulador de voltaje interno que, al no usarse, se debe conectar a tierra a través de un condensador de 10uF.
- El **pin 21** configurados como salida digital. Conectado a SD.CS de la pantalla TFT mediante pull up externo. Se activa en baja para establecer conexión con la tarjeta microSD.
- El **pin 22** configurados como salida digital. Conectado a TFT.CS de la pantalla TFT mediante pull up externo. Se activa en baja para establecer comunicación con la pantalla.
- El **pin 23** configurados como salida digital y línea lógica CLK del protocolo SPI con la pantalla TFT.
- El **pin 24** configurados como salida digital y línea lógica MOSI del protocolo SPI con la pantalla TFT.
- El **pin 25** configurados como entrada digital y línea lógica MISO del protocolo SPI con la pantalla TFT.
- El **pin 26** configurado como entrada analógica conectado al CAD. Analiza la tensión de salida del sensor de temperatura.

Divisor de tensión:

Para que nuestro dispositivo pueda caracterizarse de multímetro, es evidente la necesidad de medir magnitudes eléctricas. Una de las más importantes es la tensión que se aplica en la entrada. Puesto que la finalidad es captar variaciones que oscilen entre 3.5V y 25V aproximadamente, se diseña una red de resistencias conocida como divisor de tensión.

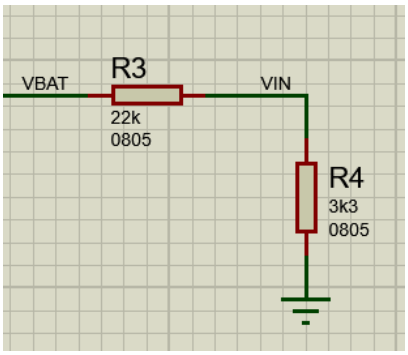


Figura 21: Divisor de tensión

Siendo VBAT la tensión de entrada y VIN la tensión de salida conectada al CAD. Esta configuración aplica una ganancia negativa a la entrada VBAT, que se traduce en VIN. Siendo directamente proporcionales, acorde a la función de transferencia:

$$VIN = \frac{R4}{R4 + R3} * VBAT$$

Lo que pretendemos es acondicionar la entrada máxima al fondo de escala (3.3V) del CAD del microcontrolador para conseguir medir todo el rango de valor y obtener la mayor resolución. Por ello aplicando:

- VBAT = 25V (Valor máximo de tensión de entrada)
- VIN = 3.3V (Fondo de escala del CAD)

Conseguimos una ganancia:

$$\frac{R4}{R4 + R3} = \frac{3.3V}{25V} = 0.132$$

Recurriendo a la utilización del menor número de resistencias con valores estándares, nos proporcionan un valor lo más próximo posible, abaratando costes. De esta manera elegimos R3 = 22KΩ y R4 = 3.3KΩ obteniendo la ganancia de:

$$\frac{3.3\text{K}\Omega}{3.3\text{K}\Omega + \ 22\text{K}\Omega} = 0.130$$

Si ahora lo aplicamos a la función de transferencia, se modifica la tensión máxima de entrada medida por el CAD, variando el rango de medida de 3.5V a 25.3V:

$$VBAT = \frac{VIN}{\frac{R4}{R4 + R3}} = \frac{3.3V}{\frac{3.3\text{K}\Omega}{3.3\text{K}\Omega + \ 22\text{K}\Omega}} = 25.3V$$

Como queremos una resolución de 0.01V, es importante comparar entre el CAD de 10 bits y 12 bits:

Resolución a fondo de escala con 10 bits → $\frac{Fondo\ de\ escala}{2^{número\ de\ bits}-1} = \frac{3.3V}{1024-1} = 0.003225V$

Resolución a fondo de escala con 12 bits → $\frac{Fondo\ de\ escala}{2^{número\ de\ bits}-1} = \frac{3.3V}{4096-1} = 0.000805V$

Ahora aplicamos esta resolución como diferencia en el fondo de escala:

10 bits → $3.3V - 0.003225V = 3.296V$

$$12\ \text{bits} \rightarrow 3.3V - 0.000805V = 3.299V$$

Y lo traducimos en la tensión de entrada, viendo la diferencia entre el valor máximo y el valor inmediatamente anterior, deduciendo la resolución del CAD con esta configuración

$$10\ \text{bits} \rightarrow 25.3V - \frac{3.296V}{\left(\frac{3.3\text{K}\Omega}{3.3\text{K}\Omega + 22\text{K}\Omega}\right)} = 25.3V - 25.269V = 0.031V > 0.01V$$

$$12\ \text{bits} \rightarrow 25.3V - \frac{3.299V}{\left(\frac{3.3\text{K}\Omega}{3.3\text{K}\Omega + 22\text{K}\Omega}\right)} = 25.3V - 25.292V = 0.008V < 0.01V$$

Como podemos ver, para conseguir una resolución de al menos 0.01V es necesaria la utilización del CAD de 12 bits.

Otro factor importante, es que las resistencias que conforman el circuito cuentan con una tolerancia del 1%, lo que puede suponer un posible error en la medida. Aplicado en la función de transferencia con una entrada de ejemplo de VBAT = 5V, se traduce en una salida VIN si los componentes no tuviesen tolerancia:

$$\text{Ideal} \rightarrow VIN = \frac{3.3\text{K}\Omega}{3.3\text{K}\Omega + 22\text{K}\Omega} * 5V = 0.652V$$

Y en una salida si las resistencias tuviesen el error máximo, tenemos una salida comprendida entre 0.663V y 0.641V:

$$\text{Error de 1\%} \rightarrow VIN = \frac{(3.3\text{K}\Omega \pm 1\%)}{(3.3\text{K}\Omega \pm 1\%) + (22\text{K}\Omega \pm 1\%)} * 5V = \frac{(3.3\text{K}\Omega \pm 33\Omega)}{(3.3\text{K}\Omega \pm 33\Omega) + (22\text{K}\Omega \pm 220\Omega)} * 5V =$$

$$\rightarrow 0.663V$$
$$\rightarrow 0.641V$$

Lo que supone un error máximo de medida de:

Error de medida→ 0.663V-0.652V=0.011V →1.68%

Error de medida→ 0.652V-0.641V=0.011V →1.68%

Amplificador diferencial:

Al igual que la tensión de entrada, otra función principal y muy importante, es el cálculo de la cantidad de corriente que circula a través del dispositivo. Para ello lo que hacemos es colocar una resistencia de precisión de 0.1Ω en la línea de tierra del USB de entrada, de manera que será atravesada por la corriente que circule a través del dispositivo más la que consume para funcionar. Esta resistencia está diseñada para de soportar hasta 3W de potencia, lo que supone poder soportar una corriente máxima de 5.47A, como vemos:

$$P = R * I^2 \rightarrow I = \sqrt{\frac{P}{R}} = \sqrt{\frac{3W}{0.1\Omega}} = 5.47A$$

Para calcular esa corriente, lo que haremos será medir la tensión de caída que se aplica en la resistencia y realizar una conversión software mediante la Ley de Ohm. Esta resistencia tiene un valor tan pequeño con la intención de afectar lo menos posible al paso de corriente, de esta manera la diferencia de tensión que se aplicará en sus extremos será muy baja con un valor máximo de 547mV. Por ello, para realizar un análisis con gran resolución, es necesario aplicar una ganancia positiva para ajustar su valor máximo al fondo de escala del CAD.

Esta ganancia se realizará con un amplificador operacional realimentado negativamente, denominado como amplificador diferencial:

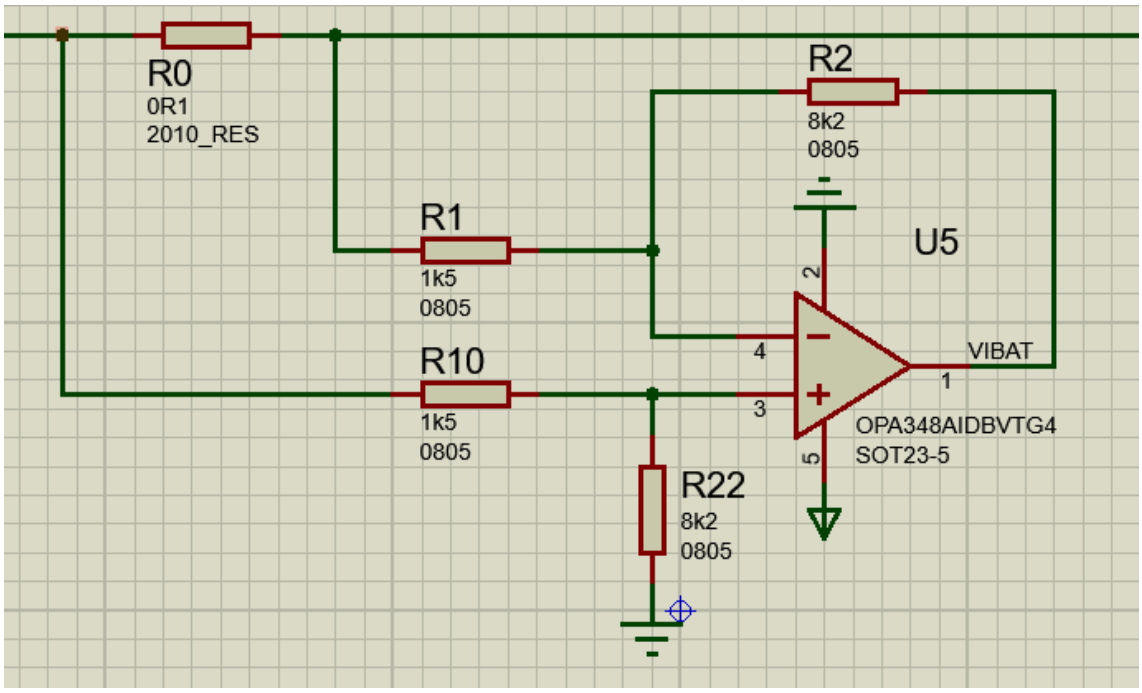


Figura 22: Amplificador diferencial

Siendo VR0 la tensión de entrada en la resistencia de precisión y VIBAT la tensión de salida conectada al CAD, obtenemos la función de transferencia de esta configuración donde R2=R22 y R1=R11:

$$VIBAT = \frac{R2}{R1} * VR0$$

Aunque la corriente máxima soportada por R0 es 5.47A, haremos un cálculo para acondicionar la señal con un máximo de 6A. Recomendando superar los 5.47A durante un corto periodo de tiempo, ya que, gracias al diseño del dispositivo, hay espacio suficiente para que se produzca disipación térmica mediante convección con el aire. Aplicamos los valores:

- VR0 = 0.1Ω*6A = 0.6V (Valor máximo de tensión de entrada)
- VIBAT = 3.3V (Fondo de escala del CAD)

Y obtenemos una ganancia:

$$\frac{R2}{R1} = \frac{3.3V}{0.6V} = 5.5$$

Esta configuración al igual que en el caso anterior, se elegirán unos valores estándar para las resistencias, eligiendo R2 = 8.2KΩ y R1 = 1.5KΩ proporcionando la ganancia más próxima de:

$$\frac{8.2K\Omega}{1.5K\Omega} = 5.466$$

Esto supondrá un cambio en la corriente máxima que podamos medir, modificando el rango de 0A a 6.036A. Para calcular este valor calculamos de nuevo la función de transferencia con las resistencias elegidas:

$$VR0 = \frac{VIBAT}{\frac{R2}{R1}} = \frac{3.3V}{\frac{8.2K\Omega}{1.5K\Omega}} = 0.6036V$$

Que convertido a corriente:

$$I(R0) = \frac{VR0}{R0} = \frac{0.6036V}{0.1\Omega} = 6.036A$$

Con la imposibilidad de realizar pruebas físicas del funcionamiento con los medios de los que dispongo a mi alcance ahora mismo, se realiza un análisis más exhaustivo mediante una simulación en Proteus 8 Professional, para analizar la respuesta del amplificador operacional, aplicando una entrada sinusoidal de corriente a un 1Hz con un offset de 3.018A, y una amplitud de 6.036A.

Realizamos el análisis analógico y obtenemos la siguiente figura 23, donde vemos que la tensión de salida VIBAT(Verde) satura a 3.24V cuando la corriente que circula por RO(Rojo) supera los 5.926A. Por ello, tomando esto como base, fijando el rango de medida de corriente de 0A a 5.926A.

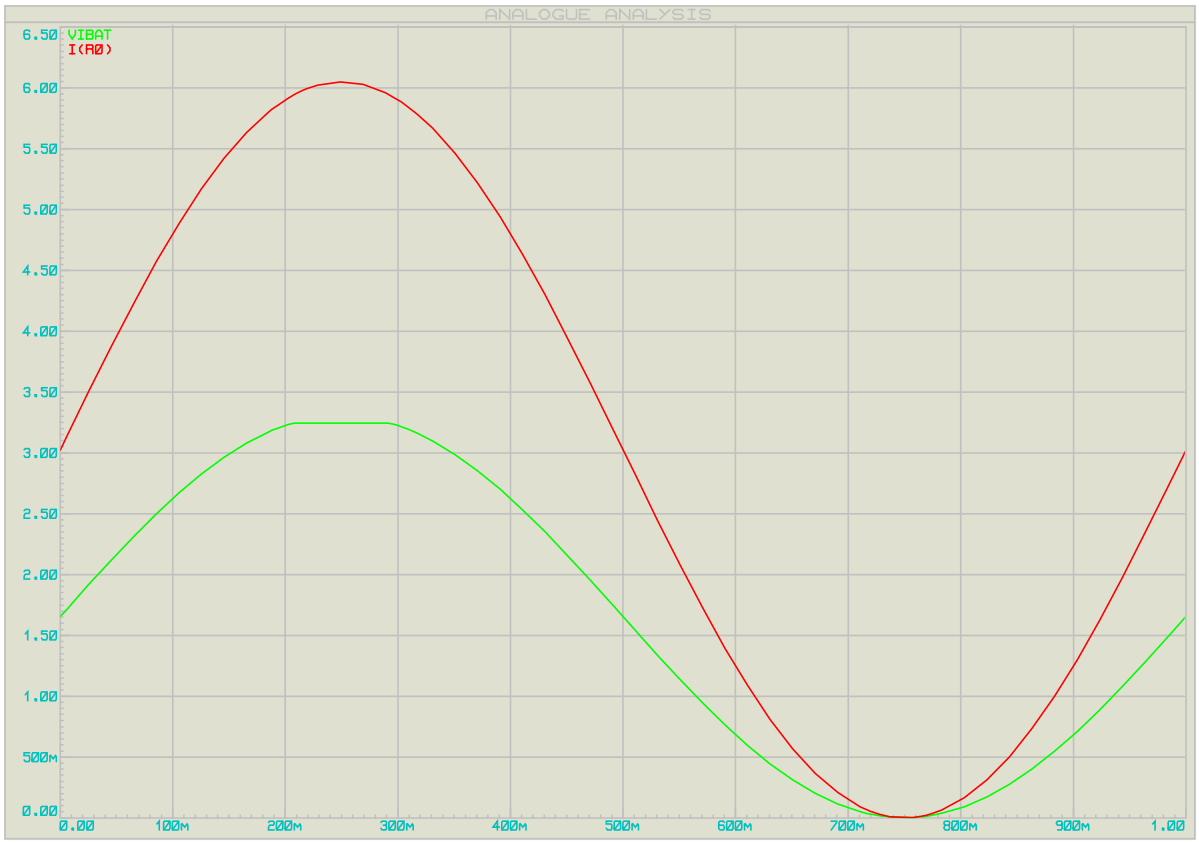


Figura 23: Salida VIBAT vs IR0

En este caso buscamos una resolución de 0.001A, y contando que utilizaremos el CAD de 12bits, puesto que ya tenemos la limitación por el divisor de tensión. Tenemos una resolución del CAD de 0.000805V. Aplicándolo como diferencia al fondo de escala, obtenemos:

$$3.3V - 0.000805V = 3.299V$$

Y lo traducimos en la corriente que fluye a través de R0, viendo la diferencia entre el valor máximo y el valor inmediatamente anterior calculado:

$$6.036A - \frac{\left(\frac{VIBAT}{\frac{R2}{R1}}\right)}{R0} = 6.036A - \frac{\left(\frac{3.299V}{\frac{8.2K\Omega}{1.5K\Omega}}\right)}{0.1\Omega} = 6.036A - 6.035A = 0.001A$$

Consiguiendo una resolución final de la resolución final de 0.001A.

Por último, procedemos a calcular el porcentaje de error de medida máximo que se puede producir debido a las tolerancias de un 1% de las resistencias. Para ello aplicando en la función de transferencia una salida de ejemplo de VIBAT = 2V, con el fin de realizar un mejor análisis, obtendríamos la salida ideal de corriente:

Ideal → $I(RO) = \frac{\left(\frac{VIBAT}{\frac{R2}{R1}}\right)}{R0} = \frac{\left(\frac{2V}{\frac{8.2K\Omega}{1.5K\Omega}}\right)}{0.1\Omega} = 3.658A$

Y si aplicamos el error máximo de 1% de las resistencias a la función, tenemos una salida entre comprendida entre 3.695A y 3.77A:

Error de 1% → $I(RO) = \frac{\left(\frac{2V}{\frac{8.2K\Omega \pm 1\%}{1.5K\Omega \pm 1\%}}\right)}{0.1\Omega \pm 1\%} = \frac{\left(\frac{2V}{\frac{8.2K\Omega \pm 82\Omega}{1.5K\Omega \pm 15\Omega}}\right)}{0.1\Omega \pm 0.001\Omega} = \begin{matrix} \rightarrow 3.55A \\ \rightarrow 3.77A \end{matrix}$

Lo que supone un error de medida máximo del 3.06%:

Error de medida→ 3.658A-3.55A=0.108A →2.95%

Error de medida→ 3.77A-3.658A=0.112A →3.06%

Regulador de tensión:

Es el encargado de proveer a todo el circuito de alimentación a una tensión estable de 3.3V por el pin Vout, para entradas variables por el pin VCC de 3.5V a 36V. Con una configuración base establecida por el fabricante, se colocan condensadores del orden de microfaradios en la entrada y salida del componente. Conocidos como condensadores de desacoplo, eliminan en la medida de lo posible los transitorios que se puedan producir en la alimentación. En lo que respecta al resto de circuitos integrados, se colocaran condensadores de desacoplo cerca de los pines de alimentación con el mismo propósito.

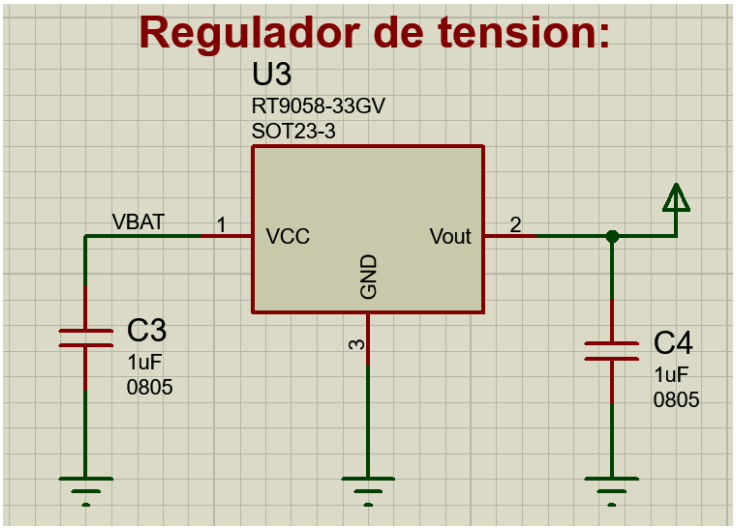


Figura 24: Regulador conexionado

3.2. Lista de materiales

Como parte del diseño es importante tener un control de los materiales utilizados, así como de su precio y otras características. Para ello gracias a Proteus 8 Professional tenemos la posibilidad de dejarlo reflejado en una Lista de Materiales (*Bill of Materials*). Esta se puede configurar para que muestre todas las características que queramos de los componentes. Y en este caso reflejamos la cantidad, referencia, valor, empaquetado y precio, como vemos en la figura 25.

A la hora de diseñar este proyecto, se tuvo muy en cuenta el precio de los componentes con la intención de que el gasto fuese lo menor posible, por si se quisiera optar a la comercialización del dispositivo. En este caso hemos reflejado el precio que supondría comprar una sola unidad de cada componente, con la intención de visualizar el gasto más elevado, ya que, si hacemos un pedido de más unidades, se reduce el coste unitario por cada componente.

Con un precio de 11.17€, tenemos que añadirle el gasto que supone adquirir la placa de circuito impreso, que ronda los 27€ el pedido mínimo de 5 placas. Si a esto le deducimos el valor de una sola placa (5.4€) junto con el resto de los componentes (11.17€), nos sale por valor total de 16.57€.

Si analizamos el mercado podemos observar precios que rondan los 15€ para dispositivos con características parecidas o inferiores y si tenemos en cuenta que son producidos en grandes cantidades, podemos observar la posibilidad de llevar nuestro proyecto al mercado.

1 Modules				
Quantity	References	Value	PCB Package	Precio
1	M5	TFT 1.3"	CONN-SIL10	€2,88
Sub-totals:				€2,88
4 Integrated Circuits				
Quantity	References	Value	PCB Package	Precio
1	U1	PIC24FJ256GA702	SSOP28	€1,68
1	U3	RT9058-33GV	SOT23-3	€0,75
1	U4	LMT85DCKT	SOT65P210X110-5	€0,71
1	U5	OPA348AIDBVTG4	SOT23-5	€0,56
Sub-totals:				€3,70
6 Miscellaneous				
Quantity	References	Value	PCB Package	Precio
2	BTN1-BTN2	B3F-3150	BUTTON	€0,38
1	J1	CONN-SIL6	CONN-SIL6	€0,13
1	J3	MC32603	USB-A-M	€0,47
1	J4	MC32593	USB-A-H	€0,51
1	X1	QCL8.00000F18B23B	XTAL18	€0,18
Sub-totals:				€2,06
13 Resistors				
Quantity	References	Value	PCB Package	Precio
1	R0	0R1	2010_RES	€0,54
2	R1,R10	1k5	0805	€0,01
2	R2,R22	8k2	0805	€0,01
1	R3	22k	0805	€0,01
1	R4	3k3	0805	€0,01
6	R5-R9,R12	10k	0805	€0,01
Sub-totals:				€0,69
8 Capacitors				
Quantity	References	Value	PCB Package	Precio
2	C1-C2	100nF	0805	€0,23
2	C3-C4	1uF	0805	€0,23
1	C5	10uF	0805	€0,23
1	C6	0.01uF	0805	€0,23
2	C7-C8	18pF	0805	€0,23
Sub-totals:				€1,84
Totals:				€11,17

Figura 25: Lista de materiales (BOM)

3.3. Diseño de la placa de circuito impreso (PCB)

A la hora de diseñar la placa de circuito impreso (PCB), lo primero es el diseño de las huellas de nuestros componentes que no se encuentren en las librerías de Proteus 8 Professional. Recurriendo al datasheet de cada componente y según su empaquetado, el fabricante nos da unas dimensiones de los *pads* donde irán soldadas las patillas a la PCB.

Estas huellas se diferencian en dos modos principalmente:

- La tecnología de montaje superficial (SMT - *Surface Mount Technology*), que se caracteriza por la soldadura directa de los pines de los componentes a la superficie de la PCB. Como es el caso de las resistencias soldadas en nuestro dispositivo, como podemos ver en la figura 26, así como de otros componentes como el microcontrolador, sensor de temperatura, etc.

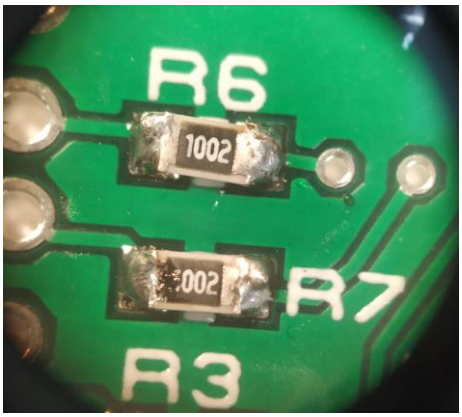


Figura 26: Resistencias R6 y R7 soldadas en PCB

- La tecnología de agujero pasante (THT – *Through Hole Technology*), que se caracteriza por utilizar agujeros creados en la PCB donde se insertan las patillas de los componentes para soldarse posteriormente. Como es el caso del USB macho tipo A, que consta de pines de inserción para un mayor agarre a la PCB. En el lado izquierdo de la figura 27 podemos ver los pines del conector USB insertados en sus agujeros, mientras que en el lado derecho vemos la parte inferior de la PCB por donde sobresale el resto del pin que la atraviesa.

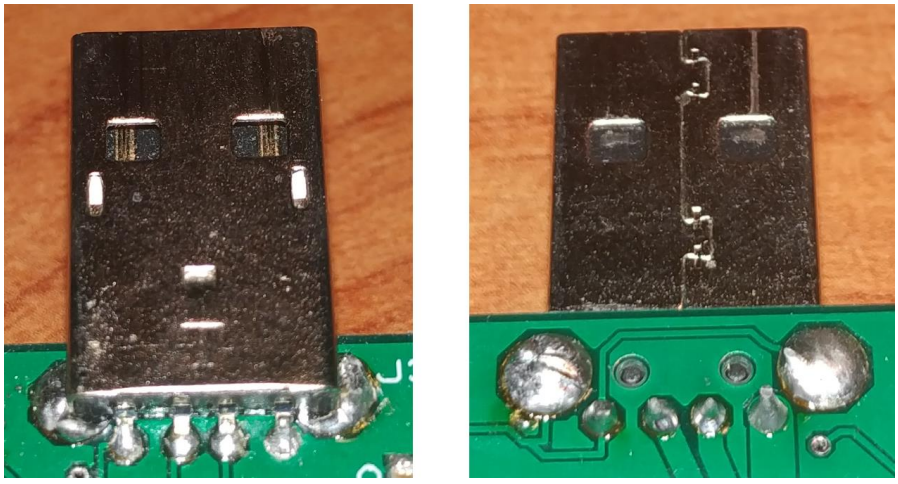


Figura 27: Conector USB tipo A soldado a la PCB

Una vez asignamos las huellas ya existentes, procedemos a la creación de las huellas que restan teniendo en las características mecánicas que nos ofrece el fabricante. Para ello comenzamos creando los *pads* según sus dimensiones y colocándolos a las distancias que corresponden en la capa *Top Copper* y se les asigna su número de pin. Por último, continuamos con la serigrafía del componente en la capa *Top Silk* y se guarda la huella en la librería.

Como resultado final tenemos las huellas que hemos creado:

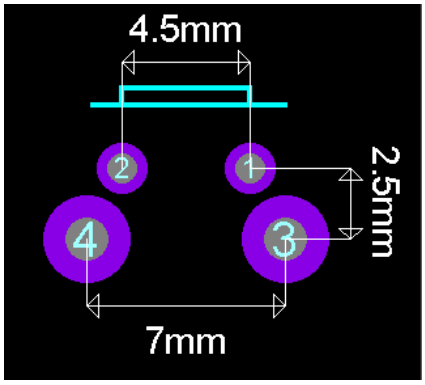


Figura 28: Huella del botón B3F-3150

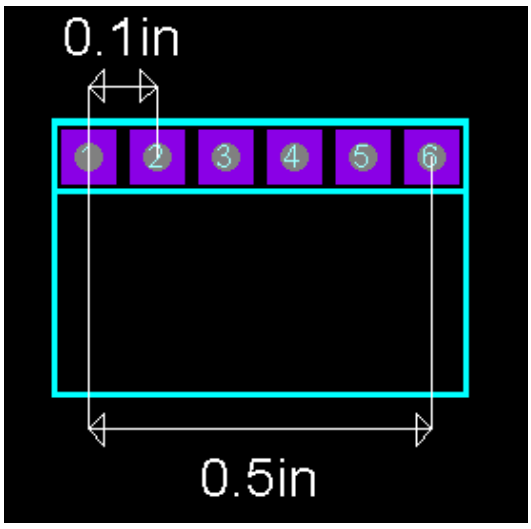


Figura 29: Huella del conector para la programación del microcontrolador

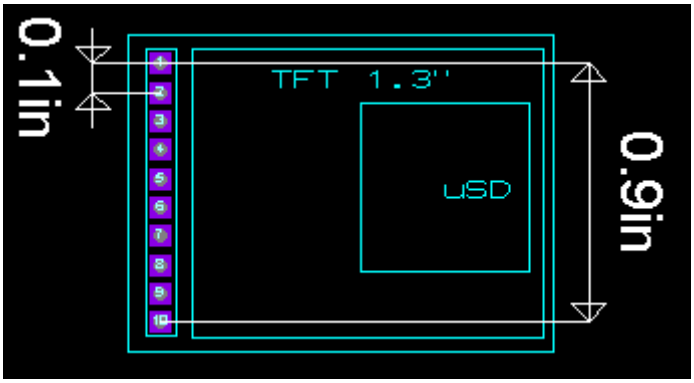


Figura 30: Huella del módulo TFT de 1,33 pulgadas

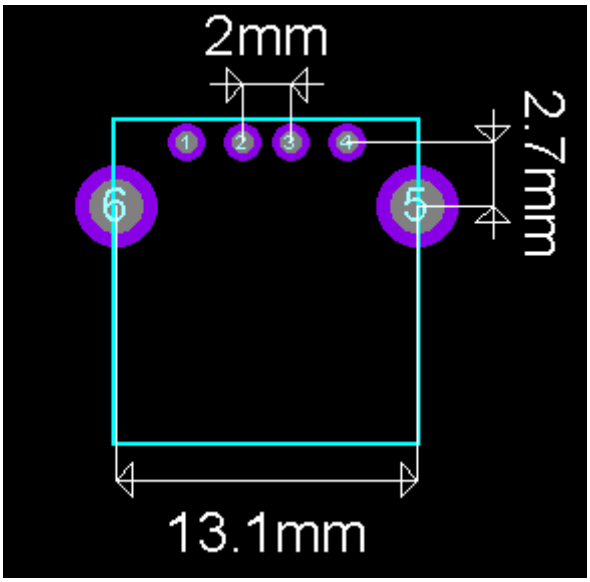


Figura 31: Huella del conector USB tipo A 2.0 hembra

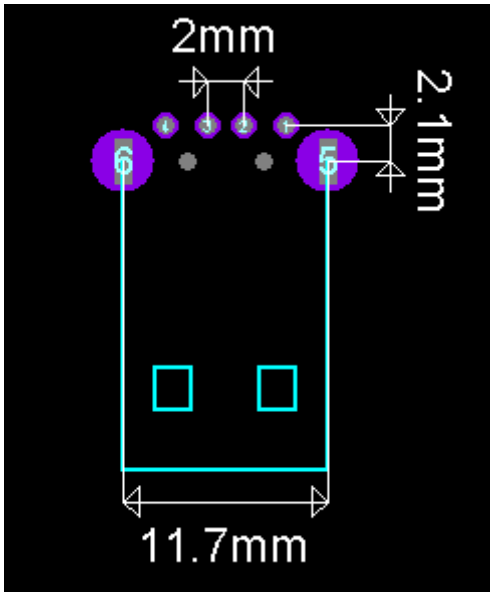


Figura 32: Huella del conector USB tipo A 2.0 macho

Y las huellas que estaban en la librería que hemos aprovechado:

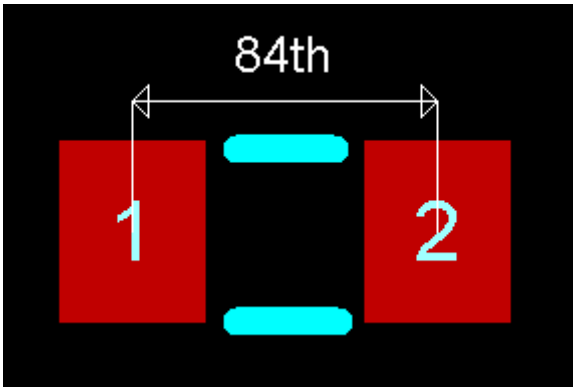


Figura 33: Huella de resistencias y condensadores

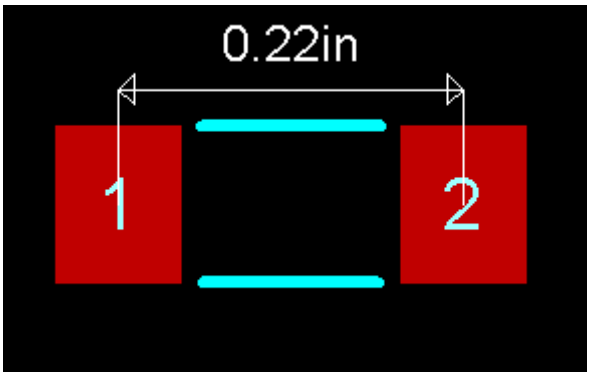


Figura 34: Huella de la resistencia de precisión R0=0.1Ω

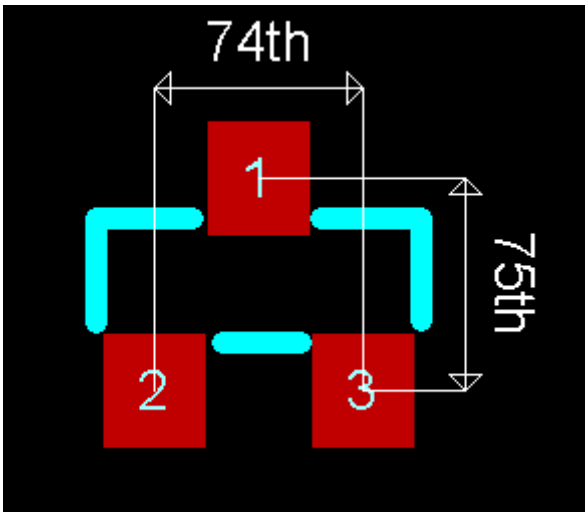


Figura 35: Huella del regulador de tensión RT9058-33GV

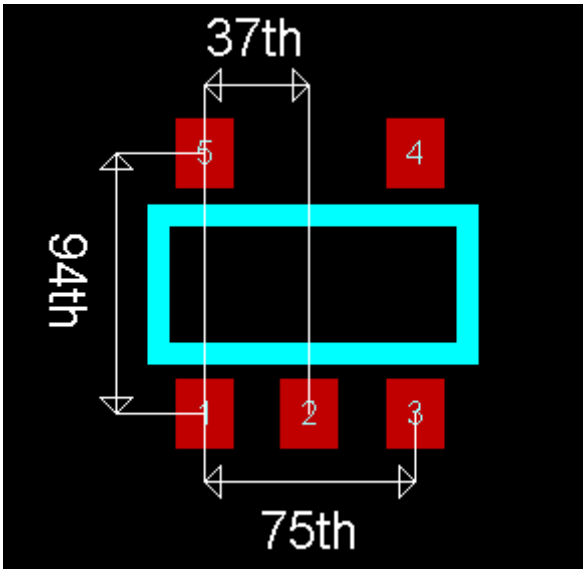


Figura 36: Huella del amplificador operacional OPA348AIDBVTG4

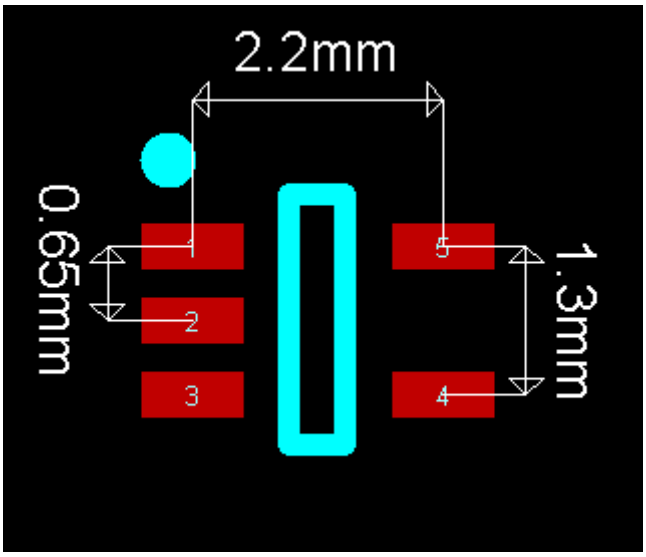


Figura 37: Huella del sensor de temperatura LMT85DCKT

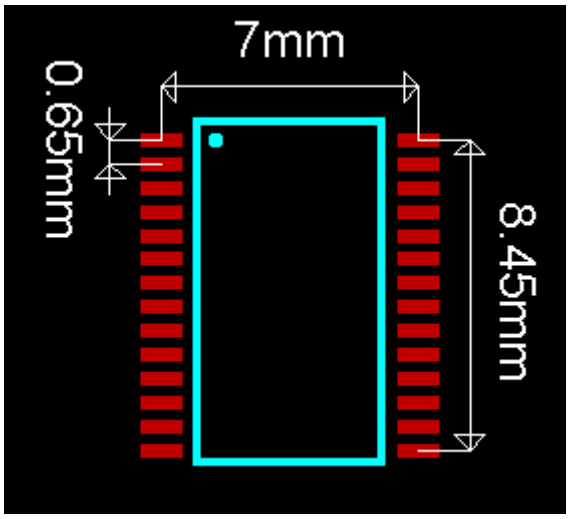


Figura 38: Huella del microcontrolador PIC24FJ256GA702-I/SS

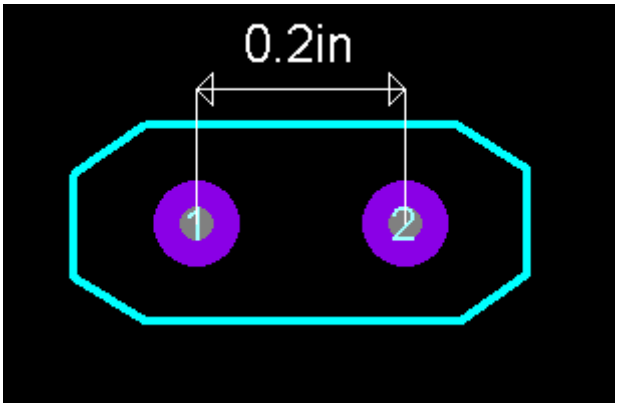


Figura 39: Huella del cristal QCL8.00000F18B23B

Una vez tenemos todas las huellas asignadas a sus respectivos componentes, y realizado todo el conexionado del dispositivo en la captura esquemática. El siguiente punto es el posicionado de los componentes en la placa de circuito impreso (PCB). Para ello, Proteus 8 Professional nos ofrece un entorno de desarrollo en el que lo primero que debemos hacer es

definir las dimensiones del borde de la PCB que contendrá todos los componentes. Se define con la capa *Board Edge* y el tamaño será de 10x10 cm, un valor muy superior al que necesitamos, pero es el que nos ofrece el proveedor PCBWay para que nos salga más barato, que si lo diseñamos con las dimensiones ajustadas.

El siguiente paso será definir las capas *Top Copper* y *Bottom Copper. Top Copper será la capa* la superior donde se colocarán todos los componentes, mientras que *Bottom Copper* será la capa inferior.

Una vez situadas las capas principales, se colocan los componentes en la PCB. A medida que vamos posicionándolos, el programa genera unos vectores que nos muestran las conexiones entre los diferentes componentes de manera idéntica a la configuración de la captura esquemática, como método de ayuda para el rutado de las pistas. Como último paso se realiza el rutado de las pistas y hemos tenido en cuenta:

- La colocación del sensor de temperatura lo más próximo a las pistas de VCC y GND de que atraviesan el dispositivo.
- Pistas VCC y GND que atraviesan el dispositivo con una anchura de pista de 1.8mm con el fin de poder soportar niveles de corriente de hasta 6A. Para el cálculo de esta anchura recurrimos a una calculadora en internet [16], en la que aplicamos los valores de altura de pista, la corriente máxima, la longitud de la línea y la máxima variación que vayamos a soportar de temperatura, dándonos como resultado un tamaño mínimo de 1.05mm como vemos en la figura 40. Le aplicamos un tamaño mayor puesto que tenemos espacio suficiente, con el fin de reducir aún más la resistencia de la pista y reducir la disipación térmica.

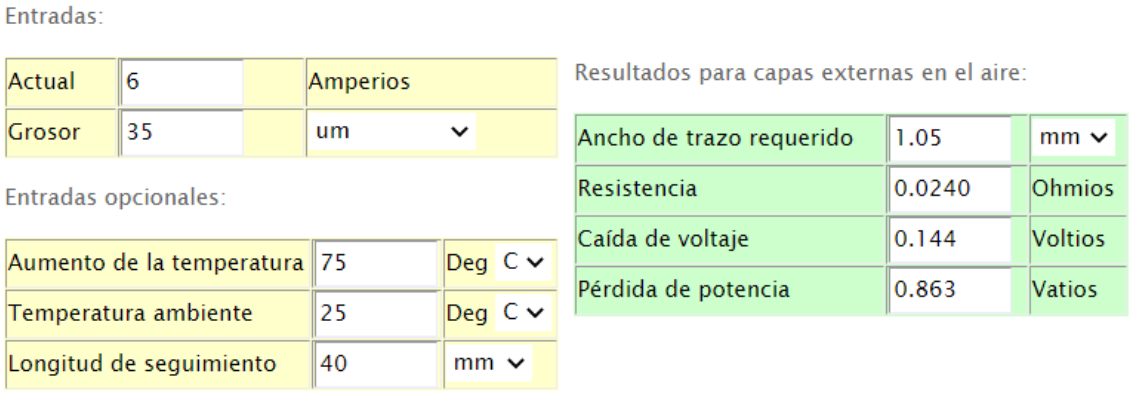


Figura 40: Calculo del grosor de pistas de alimentación VCC y GND del USB

- Pistas con la menor longitud posible para disminuir los efectos parásitos y resistivos en la medida de lo posible.
- Ajuste de la anchura y distancia de pistas de datos D+ y D- calculados en la página web PCBWay[17] para cumplir con los estándares USB [1] de impedancia diferencial de 90Ω y impedancia a tierra de 45Ω.

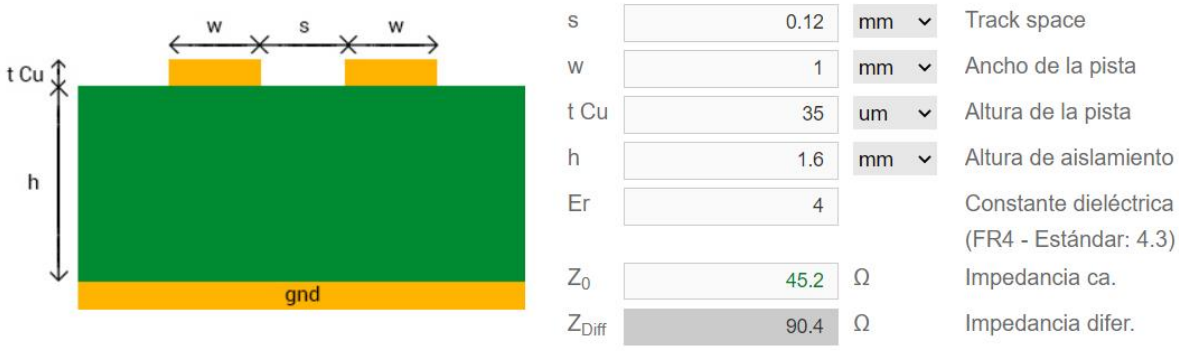


Figura 41: Calculo de grosor y distancia de pistas

- Se ha serigrafiado las zonas donde se aplicarán cortes a la PCB para su montaje, y se han incluido agujeros en los extremos para su sujeción mediante tornillería en su montaje.
- Serigrafia de cada componente, para su rápida ubicación a la hora de montarlos.

Como resultado final, vemos el diseño de la PCB que hemos enviado a fabricar. Para ello es necesario generar los archivos Gerber, que contienen toda la información relevante a la PCB.

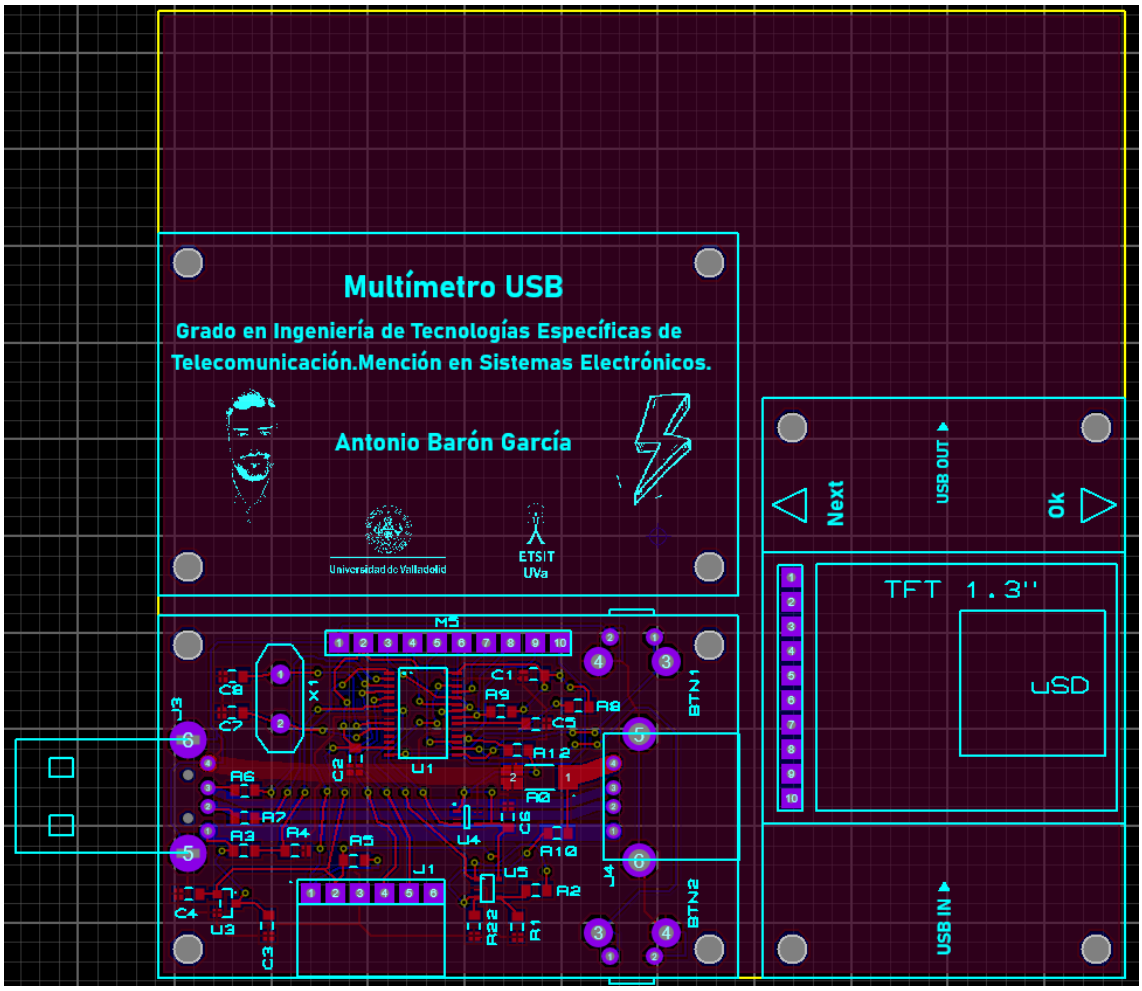


Figura 42: Layout de la PCB

Y como resultado final de la PCB fabricada.

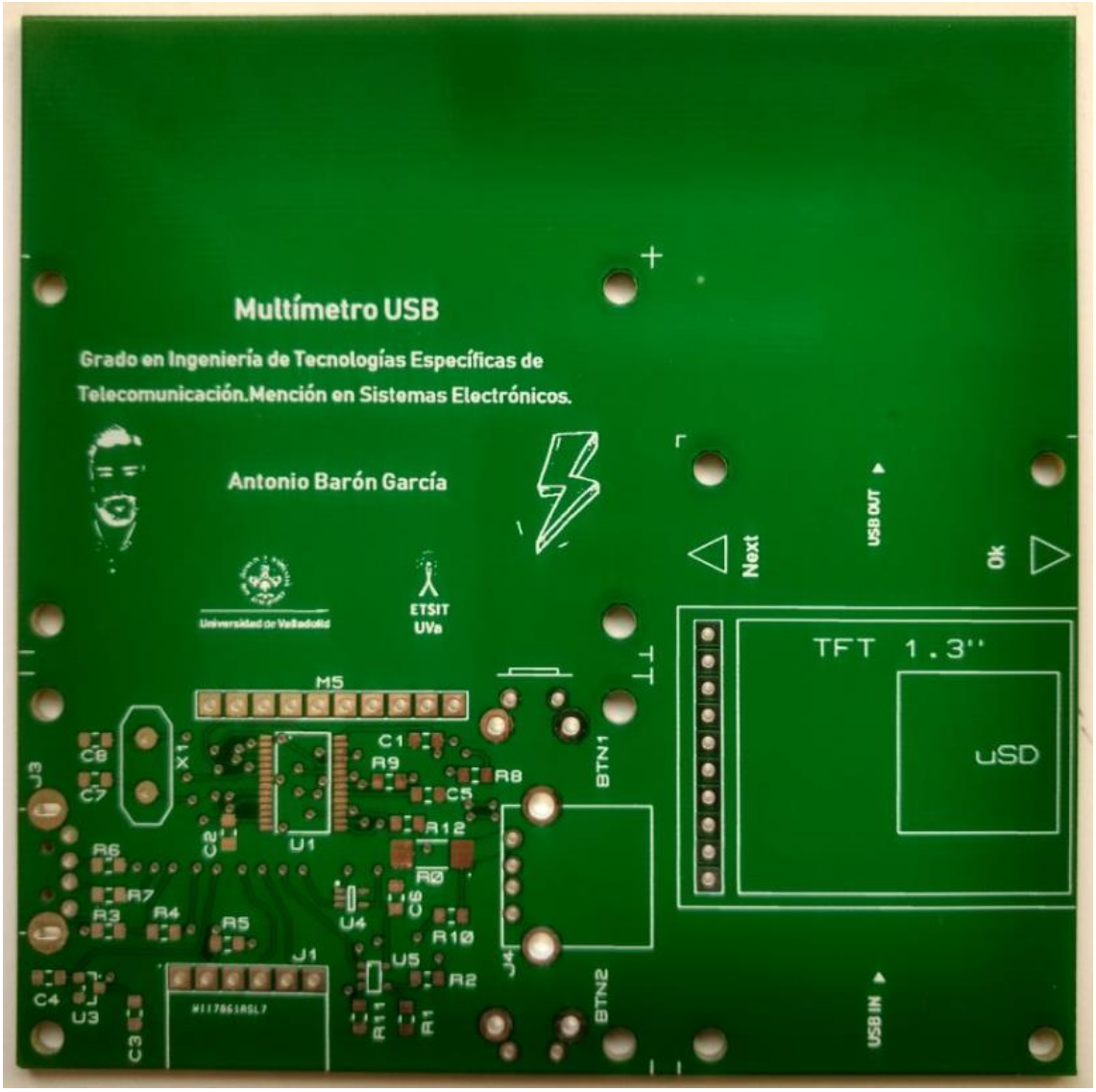


Figura 43: Placa de circuito impreso. Parte frontal

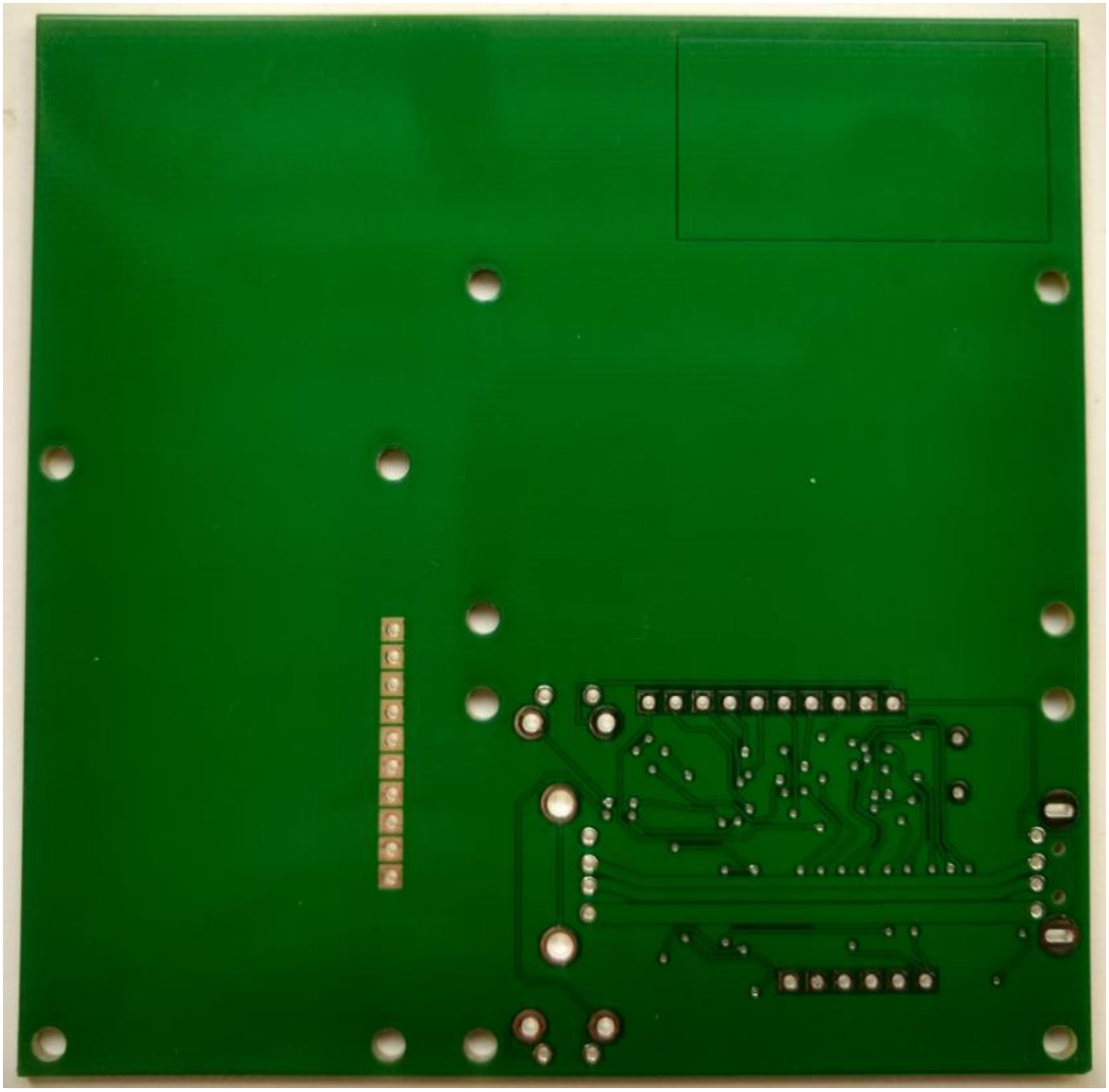


Figura 44: Placa de circuito impreso. Parte trasera

3.3.1. Análisis eléctrico y térmico

Es importante comprobar el consumo eléctrico que va a suponer esta combinación de componentes, para elegir un regulador que sea capaz de proveer al dispositivo de alimentación para su funcionamiento correctamente.

Teniendo en cuenta las corrientes máximas de alimentación de todos los componentes:

- Amplificador operacional OPA348AIDBVTG4 -> 75uA +0.5pA.
- Microcontrolador PIC24FJ256GA702 -> 7.7mA.
- Sensor de temperatura LMT85DCKT -> 9uA.
- Modulo TFT 1.3" -> 80mA.

Tenemos un consumo máximo de 87.784mA, por ello la elección del regulador **RT9058-33GV** en lo que respecta a su capacidad de proporcionar una salida máxima de 100mA, siendo capaz de alimentar el dispositivo de manera eficiente.

Si a los 87.784mA le sumamos el consumo de corriente del regulador, tenemos un consumo máximo total del dispositivo de 87.784mA+5uA=87.789mA. Dato a tener en cuenta, pues no siempre mediremos dispositivos conectados a la red eléctrica, y nuestra intención será causar el menor consumo posible.

Ahora realizamos un análisis térmico para comprobar si es necesario añadir disipación térmica extra a nuestros componentes.

Partiendo del nivel de alimentación de 3.3V y con su consumo máximo de corriente, tenemos una potencia máxima por componente, que si la comparamos con la potencia disipada máxima dentro de los rangos de operación de cada uno. Deducimos que el único componente que necesitaría disipación extra sería el regulador de tensión para valores de entrada superiores a 7.98V. Si tenemos en cuenta el diseño del dispositivo es una carcasa abierta, esto proporciona una disipación térmica por convección con el aire elevada.

Todo esto lo apreciamos en mejor detalle en la última página del esquemático.

Analisis Eléctrico y Térmico

Dispositivo	Alimentación:	Consumo de corriente maximo:	Consumo de potencia: P=V*I	Potencia disipada maxima: PD(MAX) = (TJ(MAX) - TA) / ThetaJA
OPA348AIDBVTG4	3V3	75uA + 0.5pA	P=3V3 * (75uA+0.5pA) = 0.247mW	PD(MAX)=(150-25)/(229ºC/W)= 0.545W
PIC24FJ256GA702	3V3	7.7mA	P=3V3 * 7.7mA = 25.41mW	PD(MAX)=(85-25)/(39ºC/W)= 1.538W
LMT85DCKT	3V3	9uA	P=3V3 * 9uA = 0.0297mW	PD(MAX)=(150-25)/(275ºC/W)= 0.45W
TFT 1.3"	3V3	80mA	P=3V3 * 80mA = 0.264W	
RT9058-33GV	3V5 a 24V	5uA =87.789mA	P=(Vin-Vout) * I P=(25.3V - 3V3) * 87.789mA = 1.91W P=(7v98 - 3V3) * 87.789mA = 0.41W P=(3V5 - 3V3) * 87.789mA = 17mW	PD(MAX)=(125-25)/(243.3ºC/W)= 0.411W

De todo esto deducimos que vamos a necesitar disipar el calor que se producirá en el regulador cuando la entrada del conector usb supere los 7v98 y la TFT 1.3" este consumiendo el maximo de 80mA.

FILE NAME: Multimetro USB.pdsprj	DATE: 17/09/2020
DESIGN TITLE: Multimetro USB	PAGE: 3 of 3
PATH:	
BY: Antonio Barón García REV:@REV	TIME: 21:40:08

Figura 45: Captura esquemática 3. Análisis eléctrico y térmico

3.4. Diseño del dispositivo

Para el diseño del dispositivo, nos hemos decantado por un formato "*sandwich*", que implica la utilización de las placas de circuito impreso como carcasa del dispositivo montadas una encima de otra y sujetas mediante tornillería, evitando costes adicionales.

Estará compuesto por una placa de PCB sin circuitería en la parte inferior que servirá como aislamiento. Justo encima de ella con una separación suficiente para que no haya contacto entre placas, se sitúa la PCB donde se encuentran todos los componentes interconectados, y, por último, una placa que estará cortada en la parte central, que será la posición donde situemos el módulo TFT que ira conectado directamente a la placa intermedia con el resto de circuitería.

Disposición de los conectores USB en los extremos para una mayor comodidad de uso del dispositivo, y reducido el ancho del dispositivo al mínimo, acorde a la longitud del módulo TFT.

Se le añadirá una serigrafia en el dorso como información acerca del proyecto, y serigrafia informativa de uso en la parte frontal. De esta manera llegamos a un primer esquema del diseño final que vamos a obtener.

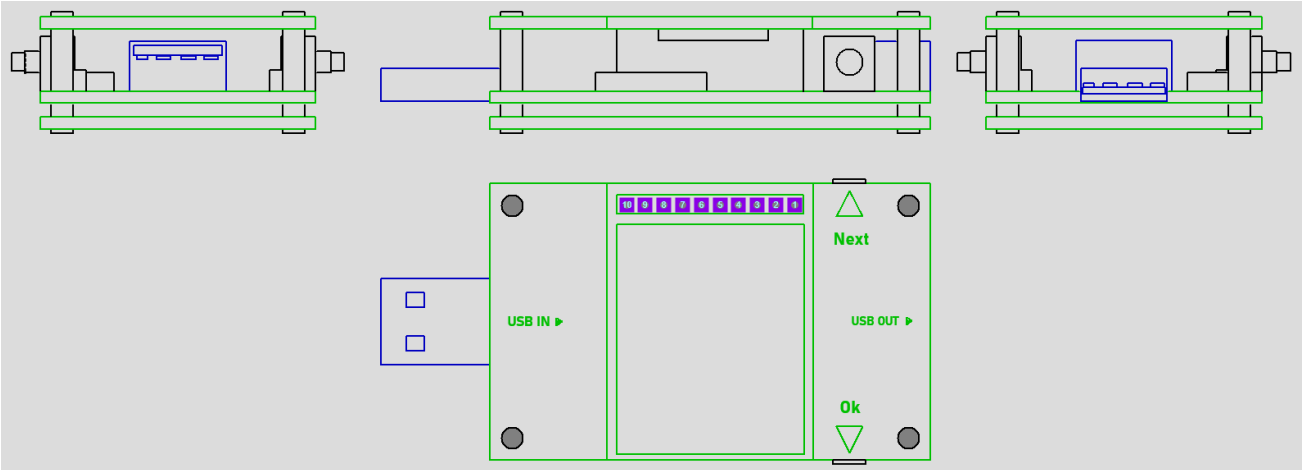


Figura 46: Diseño hardware DMM

3.5. Montaje de componentes

Una vez nos llega la placa de circuito impreso, procedemos con los cortes mecánicos oportunos para separar cada sección con el uso de la guillotina. A partir de ahí teniendo delante la captura esquemática, como la lista de materiales, procedemos a soldar todos sus componentes. Comenzamos con aquellos de montaje superficial, para un proceso más sencillo, ya que en el momento en que sigamos con los de agujero pasante, el apoyo de la placa en una superficie plana será más inestable e incómoda la soldadura.

Para que la soldadura sea lo más limpia y perfecta posible, lo primero que debemos hacer el calentar el *pad* y el pin que se va a soldar, hasta que ambos alcancen una temperatura lo suficientemente elevada. Esto lo conseguimos con un estañador a una temperatura de 400℃. Para la soldadura utilizamos estaño, un material con un punto de fusión muy inferior, 231.9℃. Esto provoca que cuando lo aplicamos en el *pad* caliente, se funda y se propague a lo largo de todo el contacto, incluido el pin, evitando una mala soldadura.

Tras acabar de soldar todos los componentes tenemos como resultado la placa final.

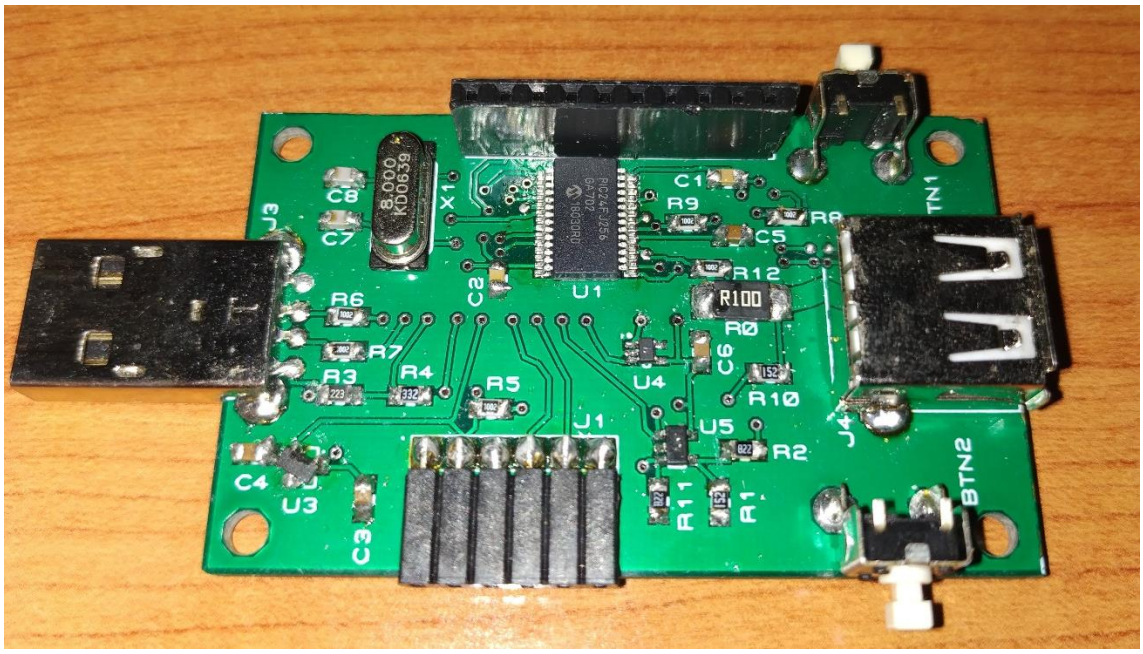


Figura 47: PCB con los componentes soldados

El siguiente paso es montar el dispositivo con su tornillería y añadirle el módulo TFT en sus pines de inserción, consiguiendo el dispositivo con las mismas características mecánicas acordes al diseño inicial.

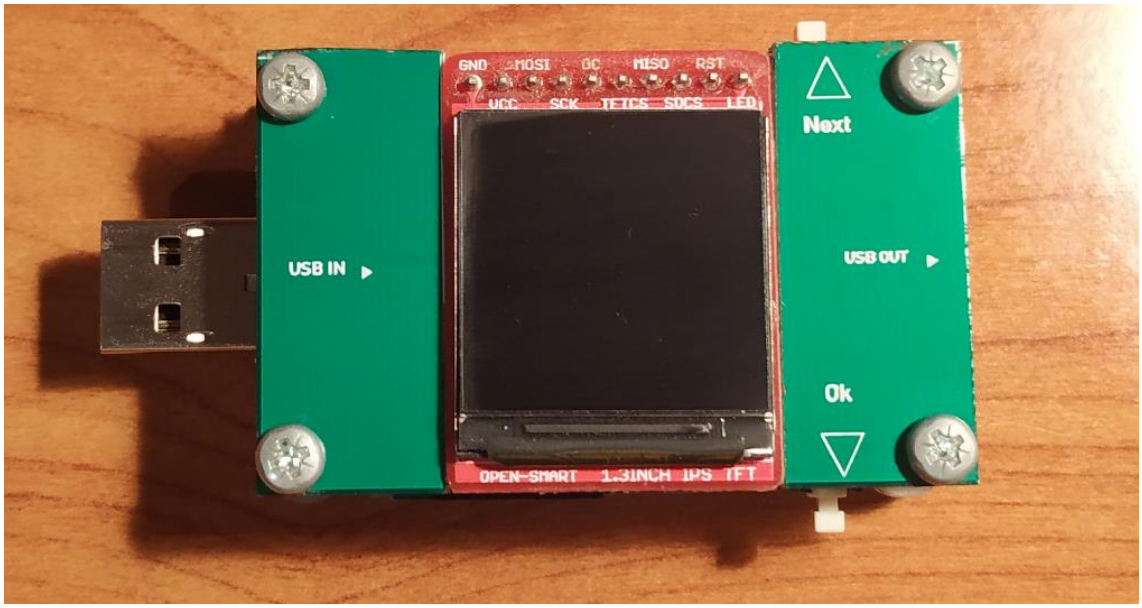


Figura 48: Vista planta DMM

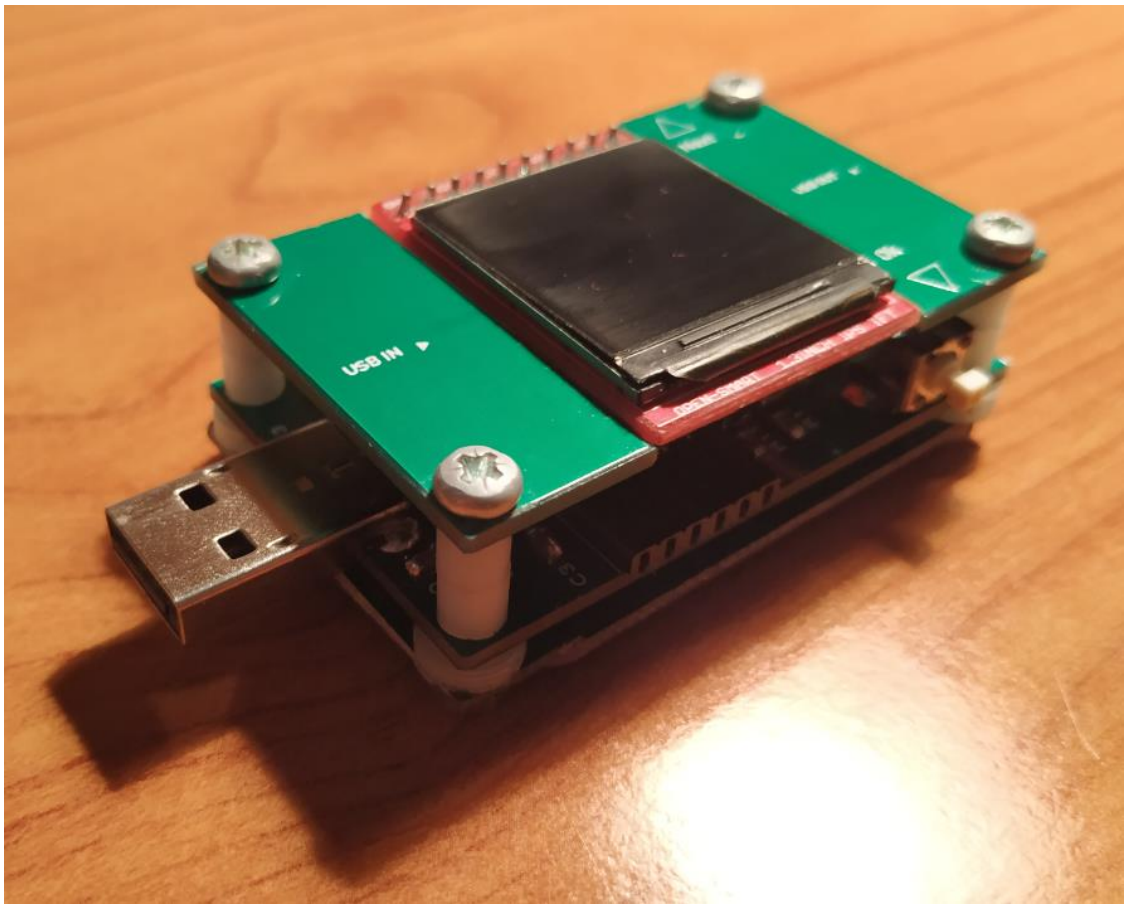


Figura 49: Vista perfil izquierda DMM

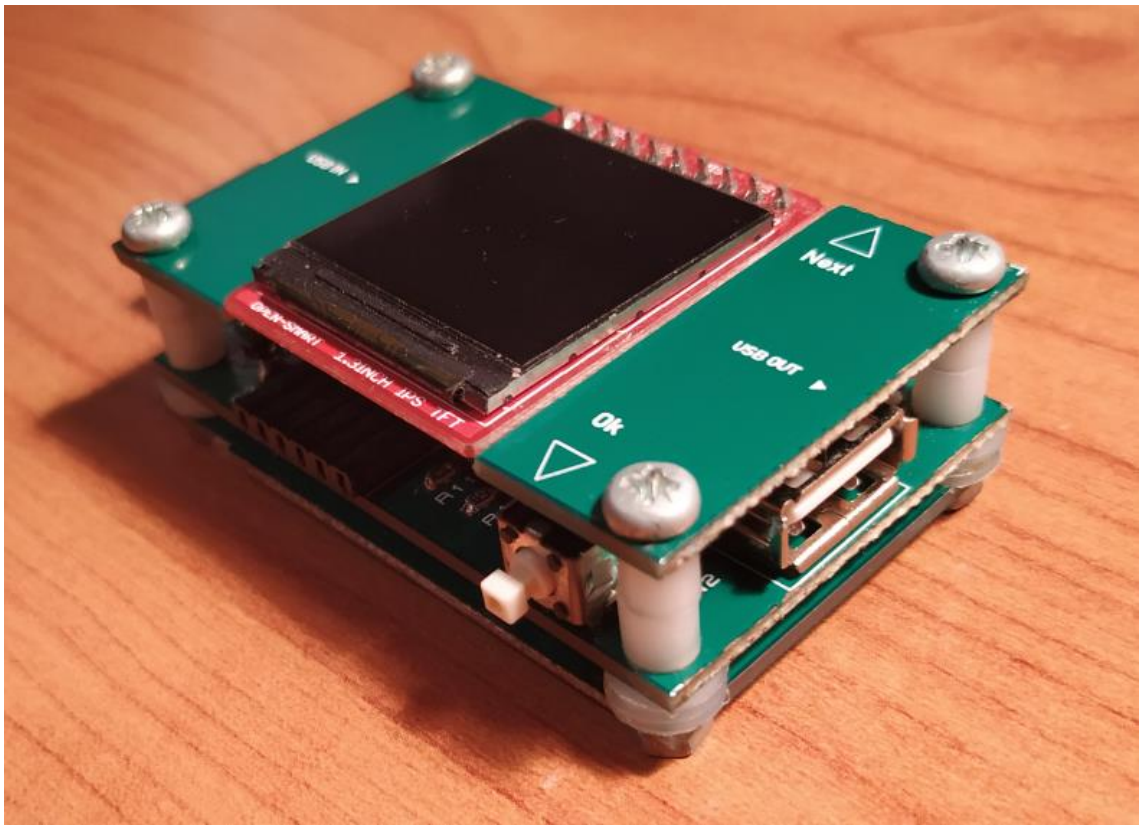


Figura 50: Vista perfil derecha DMM

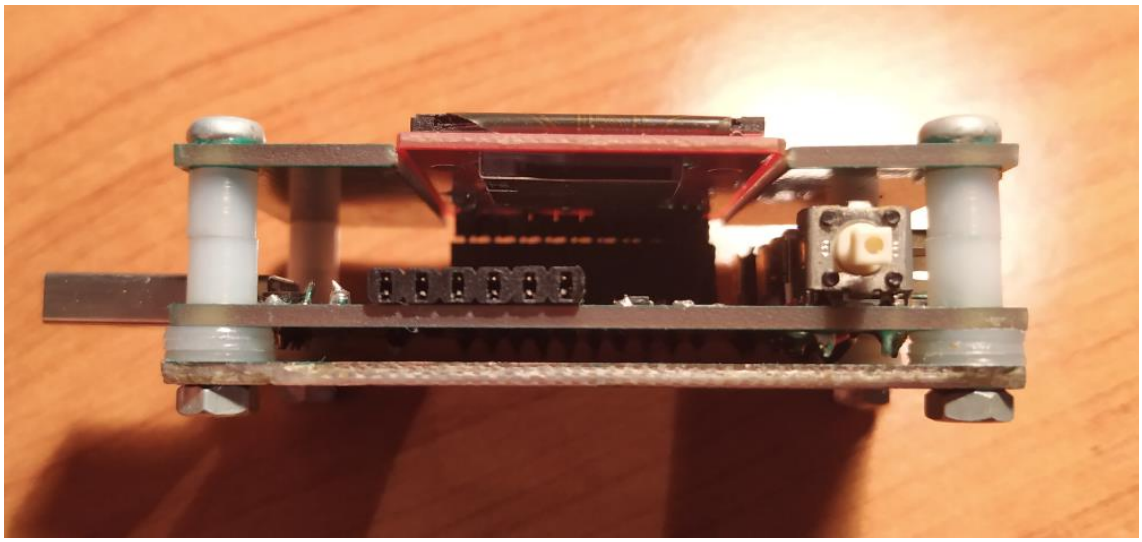


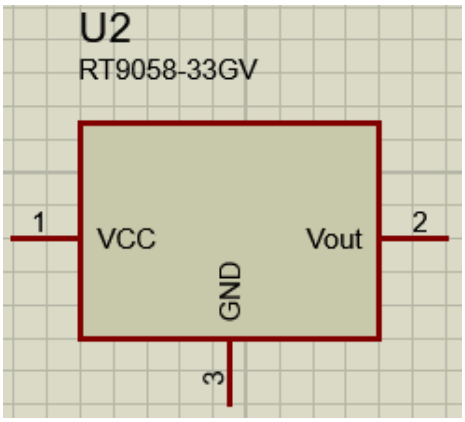
Figura 51: Vista alzado DMM

Figura 52: Vista parte posterior DMM

3.6. Errores de diseño

A medida que se realizaba el montaje, se manifestaron errores en el diseño del proyecto.

El primer error que fuimos capaces de identificar fue que no se había asignado correctamente los pines del regulador de tensión a los pads de su huella. Esto lo deducimos comprobando la alimentación de salida una vez soldado, antes de montar cualquier otro componente para evitar daños innecesarios. Con la captura esquemática correcta, acorde a las especificaciones del datasheet.



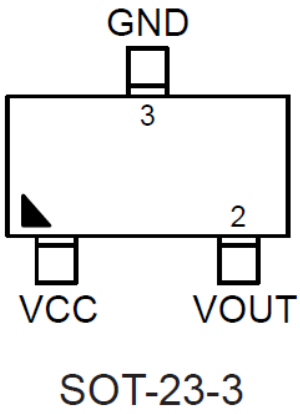


Figura 53: Diseño esquemático RT9058-33GV

Figura 54: Esquema mecánico RT9058-33GV

Se asignaron los pines a la huella de la siguiente manera, confundiendo el pin 1 con el 3, el 2 con el 1 y el 3 con el 2.

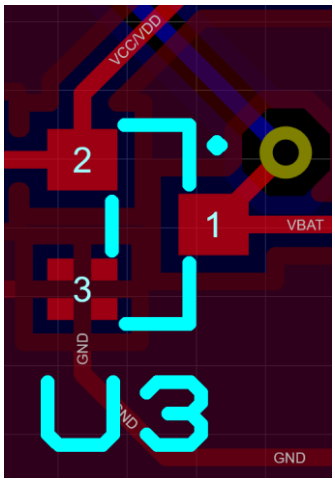


Figura 55: Huella de RT9058-33GV conexionada erróneamente

Para evitar volver a pedir otra PCB para solucionar este error, se giró el componente y soldó acorde a su patillaje correcto.

Figura 56: RT9058-33GV soldado a la PCB

El segundo error de diseño fue cortocircuitar los botones. Estos botones tienen un sencillo diseño, en el que los pines 3 y 4 se utilizan como soldadura de la carcasa a la PCB para añadir estabilidad mecánica, conectados internamente entre ellos, se conectados a tierra. Y desde el otro lado, los pines 1 y 2 suponen la acción de interruptor. En el momento en el que se presiona el botón, se conecta internamente el pin 1 con el 2. Teniendo esto claro, se

mantiene el pin 1 en alta conectado al microcontrolador, y el pin 2 conectado a tierra. En el momento en que se presiona el botón, la tensión que capta el microcontrolador es 0.

Con esto en mente, el problema vino cuando en el diseño se conectaron los pines 1 y 3, lo que significaba tener una tensión nula constantemente vista por el microcontrolador, anulando la función del botón.

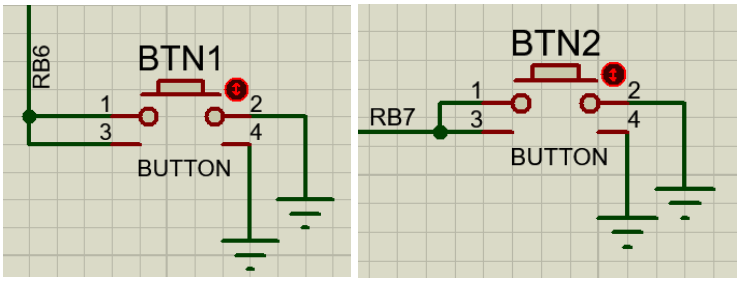


Figura 57: Conexión errónea botones

Siendo la conexión correcta la figura 58.

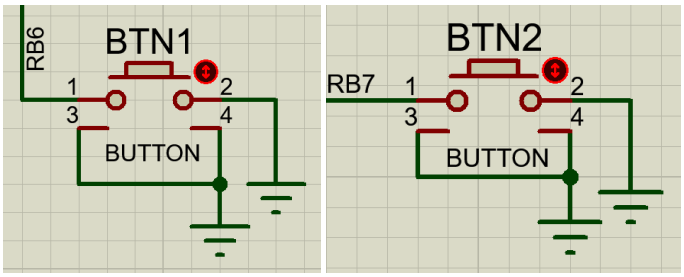


Figura 58: Conexión correcta botones

Para solucionarlo, se produjeron unas incisiones en la PCB con un cúter para cortar la pista del pin 3 al microcontrolador, dejándola únicamente a tierra, y por último con un cable conectamos el pin 1 a su entrada correspondiente en el microcontrolador. Dejando modificada como vemos en la figura 59.

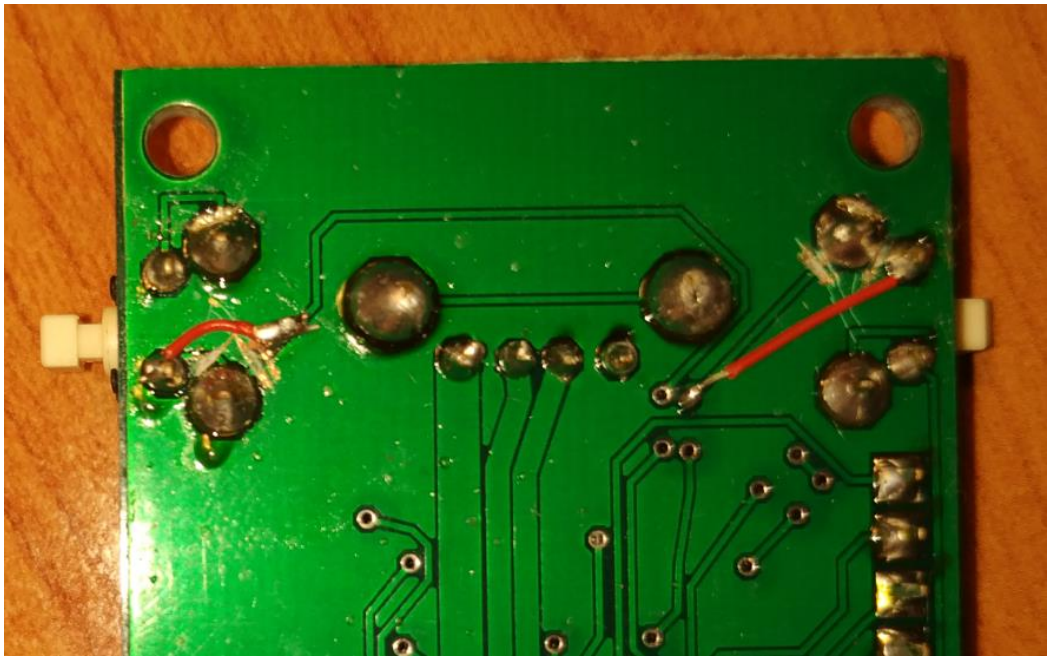


Figura 59: Modificación soldadura de los botones de la PCB

4.Firmware y Software

Tras acabar toda la parte relacionada con el hardware, proseguimos con la programación del firmware. Encargado de establecer el funcionamiento del microcontrolador. De esta manera definirá el propósito de cada pin como hemos visto en el apartado de captura esquemática. De la misma forma, se encargará de configurar el resto de los periféricos que utilizaremos, como el protocolo de comunicación SPI, el conversor analógico digital, el comparador para generar una señal PWM, el temporizador, etc.

Tras la configuración, proseguimos con el desarrollo software, para definir las funciones con las que se representara valores por pantalla, el diseño visual de la pantalla, el procesamiento de datos, cambiar de pantalla al presionar un botón, modificar la configuración del dispositivo, etc.

4.1. MPLAB X IDE

Para llevar a cabo toda esta configuración, hemos utilizado MPLAB X IDE (*Integrated Development Environment*) siendo la herramienta que nos ofrece el fabricante Microchip Technology Inc. Es un software gratuito para programar microcontroladores PIC basado en el lenguaje de programación C y C++.

Tras crear el proyecto, nos ofrece una serie de características que facilitan muy notablemente la programación del Pic24. Seleccionamos el modelo de microcontrolador, incluyendo todas sus librerías. Después seleccionamos el programador a usar, siendo en nuestro caso el Pickit 3. Por último, seleccionamos el compilador XC16. Mostrándonos un breve resumen.

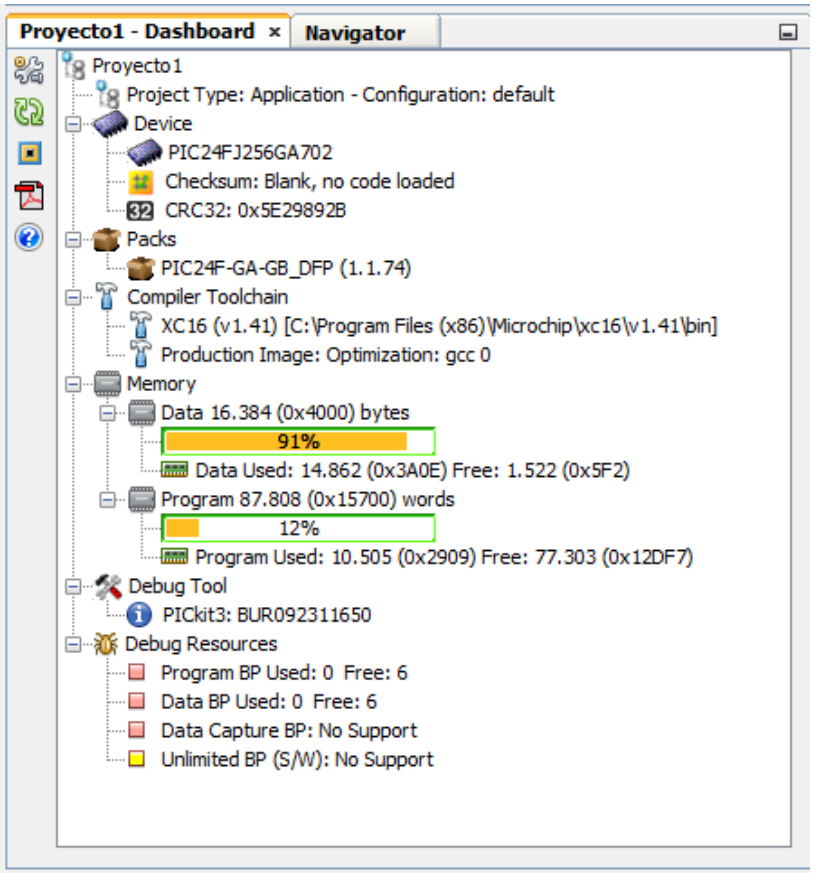


Figura 60: Tablero MPLAB X IDE

Una vez ya creado, podemos programar, pero como característica muy llamativa, descargamos el complemento MCC (MPLAB X IDE Code Configurator), que como vemos en la figura 61, nos da la posibilidad de configurar todas las características de los periféricos, como del sistema de una forma muy gráfica e intuitiva.

De la misma forma, podemos configurar directamente los registros como vemos en la figura 62 que se modifican acorde a la configuración que declaramos.

Y una vez finalizamos, generamos el código que vendrá con una breve descripción. Estableciendo de esta manera el firmware del Pic24. Dentro de MPLAB X IDE, continuaremos con el desarrollo del software en la *main*. Tras realizar toda la configuración deseada y la generación del software, procederemos a quemarlo en el componente, conectando el Pickit 3 mediante unos pines en el conector definido justo debajo de la pantalla. Dejaremos detallado el código empleado en el apéndice del proyecto

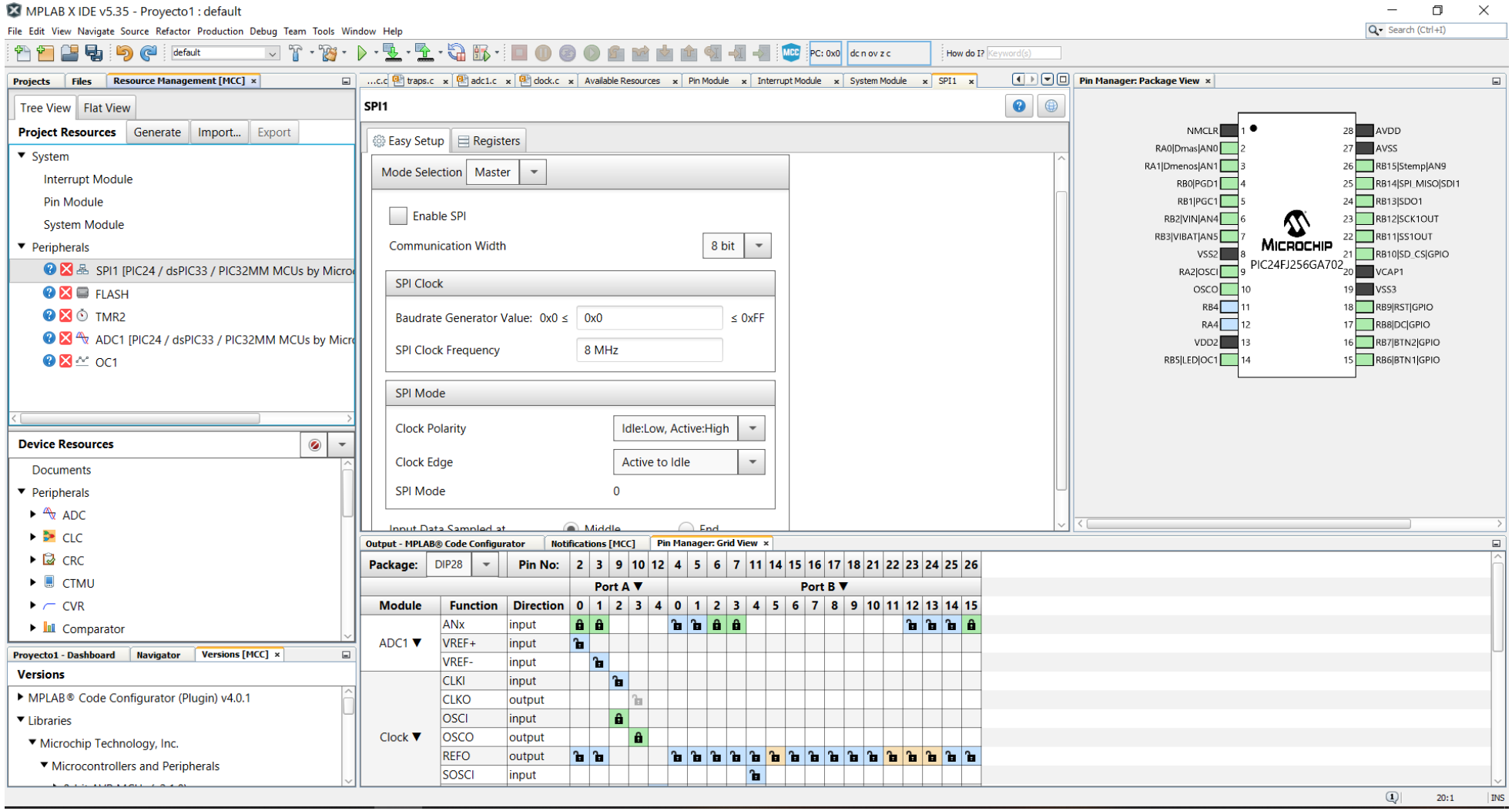


Figura 61: MPLAB X IDE Code Configurator

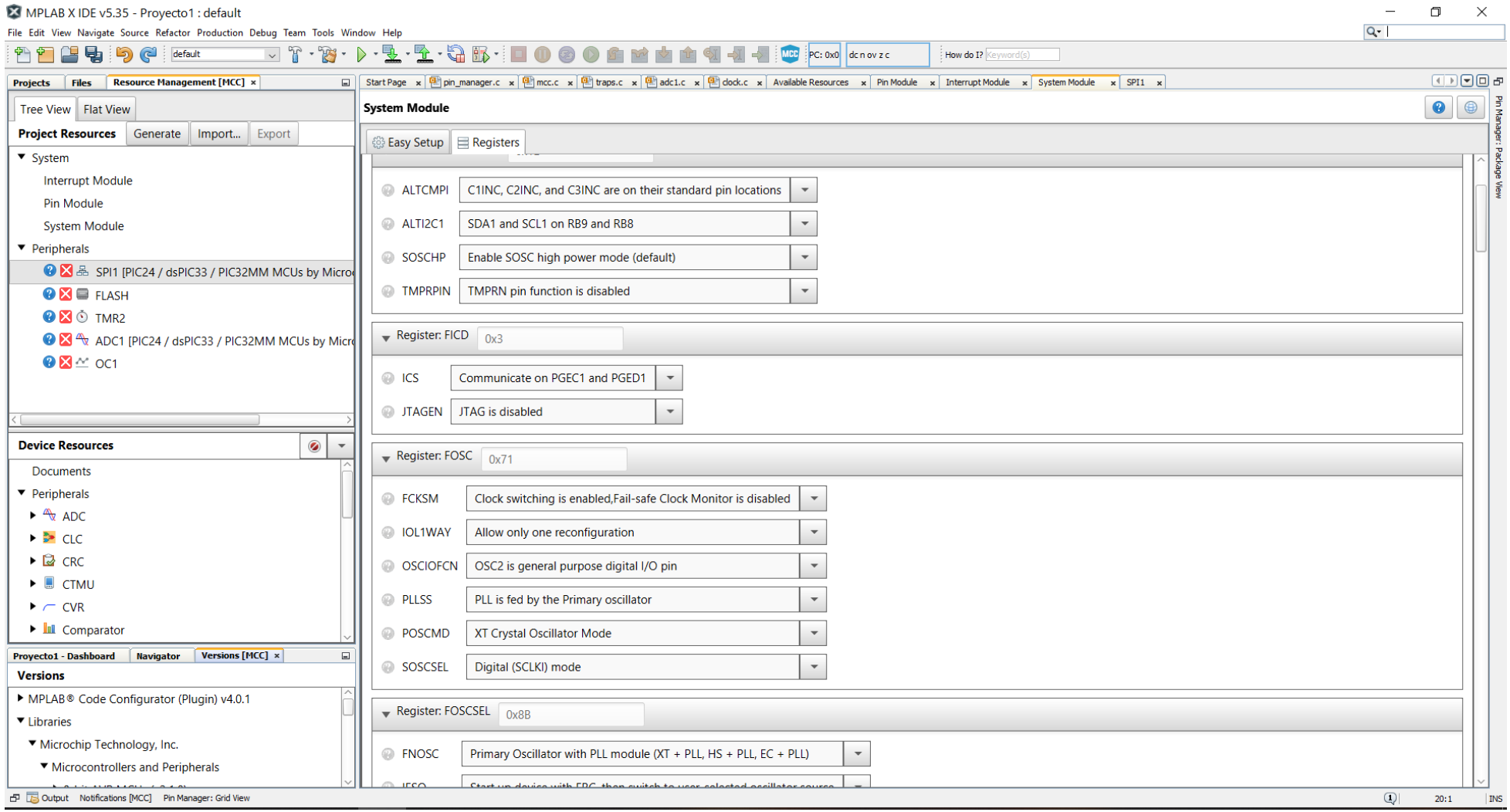


Figura 62: Registros MPLAB X IDE Code Configurator

5.Manual de usuario

Como parte fundamental, es importante conocer el funcionamiento del multímetro digital USB. Donde lo primero que tenemos que hacer es introducir el conector USB tipo A macho en el dispositivo que queramos analizar que conste de un conector USB tipo A hembra. En este momento se enciende el dispositivo, si donde está conectado consta de corriente. También podemos conectar en el otro extremo del multímetro USB, cualquier otro dispositivo, ya sea un cable para alimentar un móvil, un ratón, un teclado, una memoria USB, etc. Una vez conectado, se enciende el multímetro y empieza a capturar valores, mostrándonos la pantalla principal.

Figura 63: Pantalla 1 DMM. Valores globales

En este caso, visualizamos unos valores globales:

- La tensión aplicada al conector en voltios (V).
- La corriente que circula a través del dispositivo en amperios(A).
- La temperatura en grados centígrados o Fahrenheit (℃ /℉).
- La acumulación de capacidad en miliamperios hora(mAh).
- La acumulación de energía en miliwatios hora (mWh).
- La resistencia de carga en ohmios(Ω).
- La potencia en vatios (W).

En este punto, tenemos dos opciones que se visualizan en la pantalla, y están señalizadas en el hardware. Una de ellas es NEXT, que, asociado a un botón, tendrá la finalidad de cambiar de pantalla cuando se presiona. Por otro lado, tenemos el botón OK, que una vez presionado servirá para cambiar la rotación de la pantalla 90 grados como vemos en la figura 64.

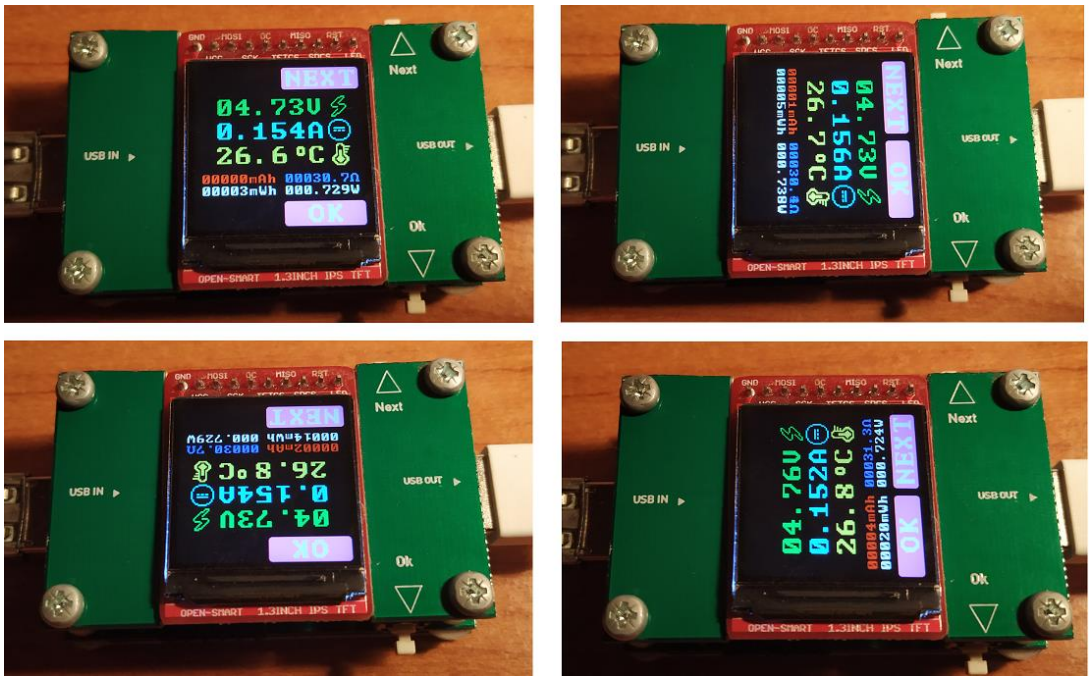


Figura 64: Rotación de pantalla

Esta función de uso de los botones, se establecerán de manera fija en las 3 primeras pantallas de visualización, constando de un total de 6.

En la segunda pantalla visualizamos los tres primeros valores en común, junto con la tensión aplicada en las líneas de datos D+ y D-:

Figura 65: Pantalla 2 DMM. Tensión línea de datos

La tercera pantalla está dedicada al análisis de la acumulación de capacidad y energía, junto con la cantidad de tiempo transcurrido y la corriente que circula en ese momento.

Figura 66: Pantalla 3 DMM. Acumulación energética

Seguimos con la pantalla número 4, que muestra una gráfica de las variaciones de tensión en el tiempo definiendo en los bordes de la pantalla la hora de análisis que se está mostrando, situado abajo a la derecha, así como el eje de tiempo abajo a la izquierda. Estas graficas se ajustan automáticamente a los valores mostrados, para una mayor precisión, mostrando los máximos y mínimos de cada instante, con la representación de una vela.

Figura 67: Pantalla 4 DMM. Gráfica de tensión

En este caso, si pulsamos el botón NEXT cambiamos de pantalla, pero si pulsamos el botón OK, entramos en modo análisis de la gráfica y cambia la función del botón NEXT.

En este modo, primero seleccionamos la hora a analizar con OK, mostrando la gráfica completa de esa hora, si ya ha transcurrido, o si está en proceso actualizándose.

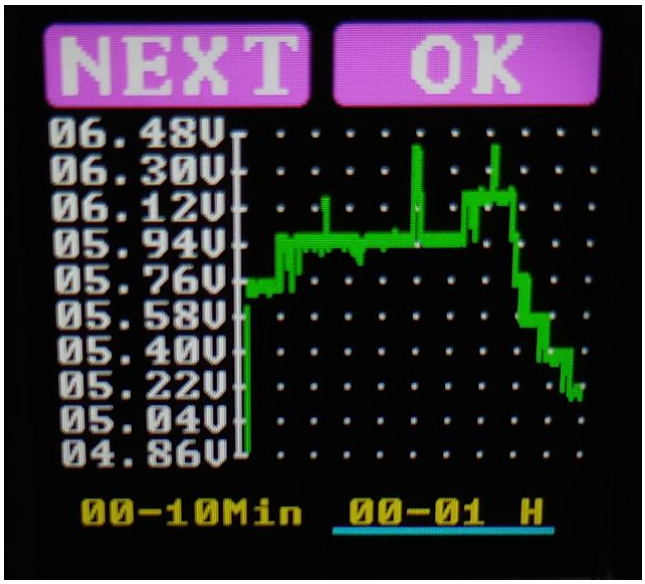


Figura 68: Pantalla 4 DMM. Modo análisis gráfica de tensión. Selección de hora

Una vez elegimos la hora, se presiona NEXT para fijar la gráfica. A partir de ahí con NEXT pasamos a elegir el intervalo de tiempo que queremos visualizar, en tramos de 10, 20 o 30 minutos, que podremos modificar en las opciones.

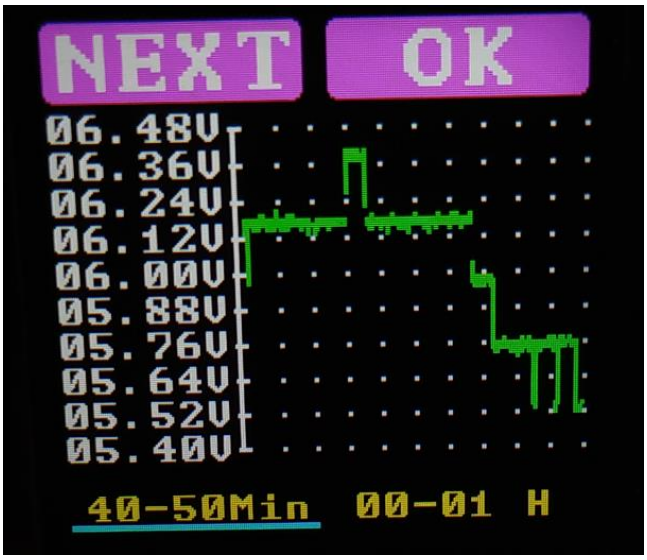


Figura 69: Pantalla 4 DMM. Modo análisis gráfica de tensión. Selección intervalo de tiempo

Para salir de aquí, solo tendremos que presionar OK para salir de modo análisis, mostrándose por pantalla, el ultimo intervalo de tiempo que se está analizando. Y si queremos volver a entrar en modo análisis, solo tenemos que volver a presionar el botón OK y repetir el mismo proceso.

Una vez fuera del modo análisis, presionamos NEXT y pasamos a la pantalla número 5, que nos muestra una gráfica de corriente y donde los botones OK y NEXT se comportan de la misma manera que en la pantalla número 4.

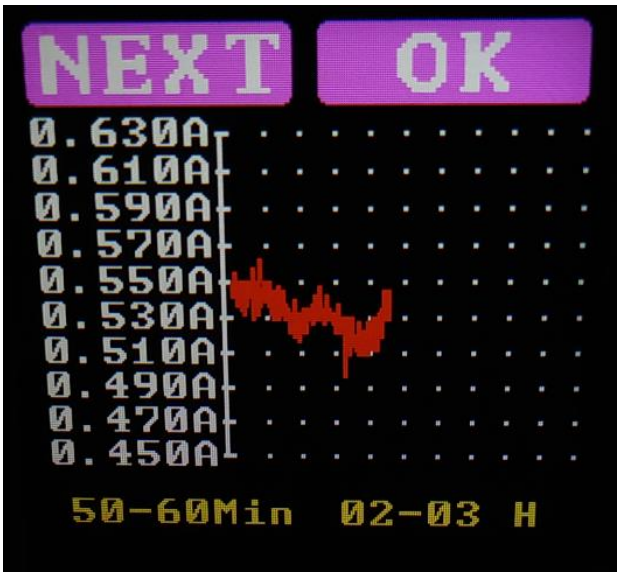


Figura 70: Pantalla 5 DMM. Gráfica de corriente

Por último, pasamos a la pantalla numero 6 con NEXT, donde visualizamos una serie de opciones que podemos modificar.

- Intervalo de tiempo a mostrar en las gráficas: Tiempo real, 10 minutos, 20 minutos, 30 minutos y 1 hora.
  En el intervalo de tiempo real, podemos visualizar en la gráfica 1 punto por segundo.
- Nivel de brillo: 0,1 ,2 ,3 ,4 y 5.
- Grados: Centígrado o Fahrenheit.

Figura 71: Pantalla 6 DMM. Ajustes del dispositivo

Para entrar en modo ajustes solo tendremos que presionar el botón OK, seleccionando la primera opción, que se modificará pulsando OK de nuevo hasta que elijamos, y cambiaremos de opción pulsando NEXT. De esta manera, una vez entramos en modo ajustes, se recorrerán las 3 opciones pulsando NEXT y tras pulsar NEXT en la última opción, saldremos del modo ajustes. En este caso podemos volver a entrar con OK, o pasar con NEXT a la primera pantalla.

Para una comprensión más clara del modo de funcionamiento del dispositivo, a continuación mostramos su diagrama de flujo.

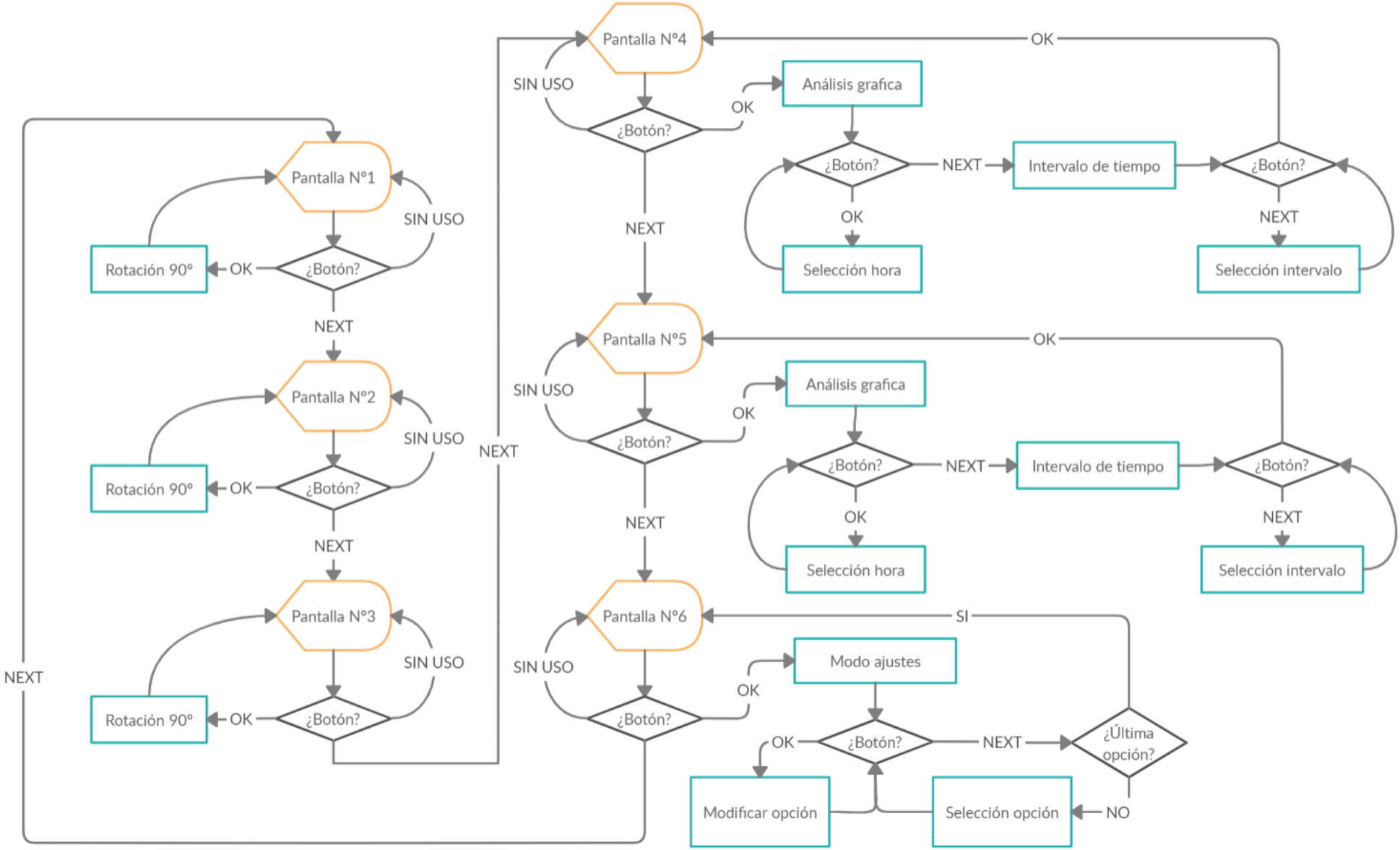


Figura 72: Diagrama de flujo del funcionamiento del dispositivo

6. Caso de uso

Procedemos a utilizar el DMM para analizar los valores de tensión y corriente que se producen en el proceso de carga de un teléfono móvil. Alteraremos este proceso en ciertos momentos, con el fin de observar su captura en las gráficas, como veremos a continuación.

En este caso se cuenta con un cargador que según sus especificaciones tiene una salida de 5V a 2A. como vemos en la figura 73. Y un cable de carga de 1 metro de longitud.

Figura 73: Cargador USB

Tomando como referencia que el dispositivo móvil consta de una batería de 3050mAh. Reducimos su carga hasta el 0% aproximadamente, ya que una batería nunca llega a descargarse completamente.

Una vez descargada, conectamos el cargador a alimentación y a este le acoplamos el DMM. Podemos visualizar como el cargador proporciona una salida de 5.09V, diferente al de sus especificaciones, demostrando la falta de precisión del cargador o en la descripción de sus características.

También podemos ver un consumo de corriente de 17mA, que será el consumo asociado al DMM y una temperatura de 24.4°C como vemos en la figura 74.

Figura 74: Valores sin alimentación de salida conectada

En el momento que conectamos el dispositivo descargado, la tensión se sitúa a 5.20V y la corriente se incrementa hasta 948mA. Transcurridos 35 min, vemos como la temperatura se

ha incrementado por el efecto Joule hasta 34.8°C, y la corriente se ha reducido a 850mA. Durante este transcurso de tiempo, se aprecia una acumulación de corriente de 440mAh.

Figura 75: Valores con alimentación de salida

Una vez cargada la batería al 100% en el transcurso de 2 horas, recurrimos a las gráficas de tensión y corriente para analizar el proceso de carga. Como vemos en la gráfica de tensión de la primera hora (Figura 76), partimos de una tensión de 5.09V hasta que se estabiliza entre 5.19V y 5.22V aproximadamente, como podemos observar en las velas de máximos y mínimos. Antes de los 30 minutos, desconectamos el teléfono móvil y volvimos a conectarlo durante una fracción de segundo, con el fin de verlo reflejado en la gráfica, observando la caída a 5.11V y la recuperación al reconectarlo.

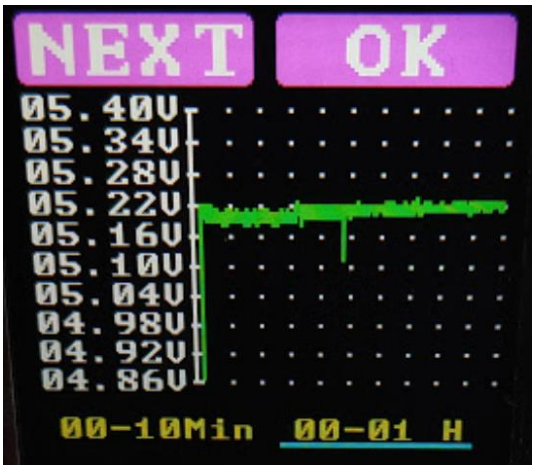


Figura 76: Gráfica de tensión de salida. Primera hora

En la segunda hora (Figura 77), vemos la caída de tensión de forma regular, asociado al fin de la carga de la batería, y observando máximos asociados al encendido de la pantalla del teléfono móvil, solicitando más alimentación. Volviendo a su caída regular una vez esta se apaga.

Figura 77: Gráfica de tensión de salida. Segunda hora

De la misma forma, podemos visualizar la gráfica de corriente de la primera hora (Figura 78) observando la subida a **940mA d**e corriente una vez se conecta el dispositivo, oscilando acorde a la corriente solicitada y disminuyendo lentamente. Podemos ver el resultado de la rápida desconexión y conexión de alimentación del teléfono móvil en el mismo instante que en la gráfica de tensión (Figura 76).

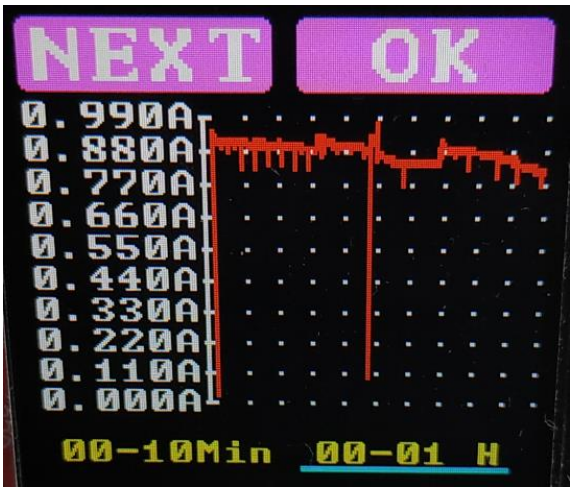


Figura 78: Gráfica de corriente de salida. Primera hora

En la segunda hora (Figura 79) tenemos el fin de la carga total del dispositivo, con una mayor pendiente en la disminución de la corriente, observando los mismos patrones que en la gráfica de tensión de esta misma hora (Figura 77), con incrementos de corriente en el momento que se encendía la pantalla del teléfono móvil.

Figura 79: Gráfica de corriente de salida. Segunda hora

Con un valor final de 1400mAh transferidos al dispositivo, deducimos una deficiencia en la capacidad de almacenamiento de la batería del teléfono móvil, o por consiguiente un problema hardware del DMM.

7. Conclusiones

Tras el desarrollo completo del dispositivo, en sus aspectos hardware y software de manera satisfactoria, podemos sacar en claro que se ha cumplido con los objetivos finales, logrando tomar medidas y representarlas por pantalla, destacando el almacenamiento de los valores de tensión y corriente para su posterior representación gráfica respecto al tiempo. Siendo de gran ayuda, para realizar diagnósticos de cargas prolongadas en el tiempo.

Debido a la imposibilidad de poder realizar pruebas con mayor precisión en el laboratorio por la situación actual que estamos viviendo, no se han podido demostrar con precisión el funcionamiento óptimo del multímetro digital USB en lo que respecta únicamente a la captura de medidas. Pudiendo comprobar que el software funciona correctamente, y en caso de haber algún error, se hallaría en el hardware del DMM.

Con este proyecto, afianzamos los conocimientos adquiridos a lo largo de la carrera en el desarrollo hardware de un proyecto, como las bases de programación firmware de un microcontrolador PIC24, y la utilización de sus periféricos para el análisis de la información capturada, como el control del módulo TFT y su representación de valores.

En líneas futuras podría optarse por realizar cambios en el diseño hardware, con la implementación de una pantalla TFT-LCD soldada directamente a la PCB, lo que supondría disminuir el grosor de la carcasa y su precio.

De la misma forma se podría implementar una conexión bluetooth para conectar a un teléfono móvil y con una aplicación Android poder interactuar más ampliamente con la información de las gráficas, y darnos la posibilidad de almacenar informes en nuestro dispositivo.

Por último, se podría modificar la conexión USB tipo A por el USB tipo C. Aunque lleva poco tiempo, empieza a tomar notoriedad gracias a sus prestaciones de alta velocidad de transmisión de datos y su capacidad para soportar corrientes elevadas.

8. Apéndice

8.1. Código main.c

```
#include "mcc_generated_files/system.h"
#include "xc.h"
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include <math.h>
#define    FCY    16000000UL // Frecuencia del ciclo de instruccion
#include <libpic30.h>
////////////////////////////////////////////////////////////////////////
//////////
#define SS1OUT_SetLow()           _LATB11 = 0 //TFT.CS en baja para
iniciar la transimision de informacion con el controlador de la TFT
#define SS1OUT_SetHigh()          _LATB11 = 1 //TFT.CS en alta supone
que no hay transimision de informacion con el controlador de la TFT
#define DC_SetLow()               _LATB8 = 0  //bit de control de DC
en baja, lo que supone que la informacion que se envia al controlador
de la TFT son Comandos
#define DC_SetHigh()              _LATB8 = 1  //bit de control de DC
en alta,lo que supone que la informacion que se envia al controlador
de la TFT son Datos
#define RST_SetLow()              _LATB9 = 0  //bit de control de
Reset en baja, inicializa reset de la TFT mediante hardware
#define RST_SetHigh()             _LATB9 = 1  //bit de control de
Reset en alta, finaliza reset de la TFT mediante hardware
#define BTN1_GetValue()           _RB6        //valor del boton 1
#define BTN2_GetValue()           _RB7        //valor del boton 2

/*Definicion de colores para los pixeles de la TFT*/
#define RGB565(r,g,b) (((r&0b11111)<<11)|(g&0b111111)<<5|(b&0b11111))
#define BLACK   RGB565(  0,  0,  0)
#define BLUE    RGB565(  0,  0,255)
#define RED     RGB565(255,  0,  0)
#define GREEN   RGB565(  0,255,  0)
#define CYAN    RGB565(  0,255,255)
#define MAGENTA RGB565(255,  0,255)
#define YELLOW  RGB565(255,255,  0)
#define WHITE   RGB565(255,255,255)
#define BROWN   RGB565(112, 66,  0)
int FG_Color = GREEN;//Color frontal
int BG_Color = BLACK;//Color de fondo

/*Comando del controlador de la TFT*/
#define ST_CMD_DELAY      0x80
#define ST77XX_NOP        0x00
#define ST77XX_SWRESET    0x01
#define ST77XX_RDDID      0x04
#define ST77XX_RDDST      0x09
#define ST77XX_SLPIN      0x10
#define ST77XX_SLPOUT     0x11
#define ST77XX_PTLON      0x12
#define ST77XX_NORON      0x13
#define ST77XX_INVOFF     0x20
#define ST77XX_INVON      0x21
#define ST77XX_DISPOFF    0x28
#define ST77XX_DISPON     0x29
#define ST77XX_CASET      0x2A
```

```
#define ST77XX_RASET      0x2B
#define ST77XX_RAMWR      0x2C
#define ST77XX_RAMRD      0x2E
#define ST77XX_PTLAR      0x30
#define ST77XX_COLMOD     0x3A
#define ST77XX_MADCTL     0x36
#define ST77XX_MADCTL_MY  0x80
#define ST77XX_MADCTL_MX  0x40
#define ST77XX_MADCTL_MV  0x20
#define ST77XX_MADCTL_ML  0x10
#define ST77XX_MADCTL_RGB 0x00
#define ST77XX_RDID1      0xDA
#define ST77XX_RDID2      0xDB
#define ST77XX_RDID3      0xDC
#define ST77XX_RDID4      0xDD
#define ST77XX_WRDISBV    0x51

// BigFont.c (C)2010 by Henning Karlsen
// Font Size      : 16x16
// Memory usage   : 3044 bytes
// # characters   : 95+9(Simbolos añadidos)
const unsigned char BigFont[] ={
0x10,0x10,
13,
0x20,0x5F,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //  <Space>
0x00,0x00,0x00,0x00,0x07,0x00,0x0F,0x80,0x0F,0x80,0x0F,0x80,0x0F,0x80,0x0F,0x80,0x07,0x0
0,0x07,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x00,0x00, // !
0x00,0x00,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x06,0x30,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // "
0x00,0x00,0x0C,0x30,0x0C,0x30,0x0C,0x30,0x7F,0xFE,0x7F,0xFE,0x0C,0x30,0x0C,0x30,0x0C,0x3
0,0x0C,0x30,0x7F,0xFE,0x7F,0xFE,0x0C,0x30,0x0C,0x30,0x0C,0x30,0x00,0x00, // #
0x00,0x00,0x02,0x40,0x02,0x40,0x0F,0xF8,0x1F,0xF8,0x1A,0x40,0x1A,0x40,0x1F,0xF0,0x0F,0xF
8,0x02,0x58,0x02,0x58,0x1F,0xF8,0x1F,0xF0,0x02,0x40,0x02,0x40,0x00,0x00,0x00, // $
0x00,0x00,0x00,0x00,0x00,0x00,0x0E,0x10,0x0E,0x30,0x0E,0x70,0x00,0xE0,0x01,0xC0,0x03,0x8
0,0x07,0x00,0x0E,0x70,0x0C,0x70,0x08,0x70,0x00,0x00,0x00,0x00,0x00,0x00, // %
0x00,0x00,0x00,0x00,0x0F,0x00,0x19,0x80,0x19,0x80,0x19,0x80,0x0F,0x00,0x0F,0x08,0x0F,0x9
8,0x19,0xF8,0x18,0xF0,0x18,0xE0,0x19,0xF0,0x0F,0x98,0x00,0x00,0x00,0x00, // &
0x00,0x00,0x00,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x0E,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // '
0x00,0x00,0x00,0x00,0x00,0xF0,0x01,0xC0,0x03,0x80,0x07,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x0
0,0x0E,0x00,0x07,0x00,0x03,0x80,0x01,0xC0,0x00,0xF0,0x00,0x00,0x00,0x00,0x00, // (
0x00,0x00,0x00,0x00,0x0F,0x00,0x03,0x80,0x01,0xC0,0x00,0xE0,0x00,0x70,0x00,0x70,0x00,0x7
0,0x00,0x70,0x00,0xE0,0x01,0xC0,0x03,0x80,0x0F,0x00,0x00,0x00,0x00,0x00, // )
0x00,0x00,0x00,0x00,0x01,0x80,0x11,0x88,0x09,0x90,0x07,0xE0,0x07,0xE0,0x3F,0xFC,0x3F,0xF
C,0x07,0xE0,0x07,0xE0,0x09,0x90,0x11,0x88,0x01,0x80,0x00,0x00,0x00,0x00, // *
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x80,0x01,0x80,0x01,0x80,0x0F,0xF0,0x0F,0xF
0,0x01,0x80,0x01,0x80,0x01,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // +
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,0x00,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x0E,0x00,0x00,0x00,0x00, // ,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0xF8,0x1F,0xF
8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // -
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,0x00,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00, // ,
0x00,0x00,0x00,0x00,0x00,0x02,0x00,0x06,0x00,0x0E,0x00,0x1C,0x00,0x38,0x00,0x70,0x00,0xE
0,0x01,0xC0,0x03,0x80,0x07,0x00,0x0E,0x00,0x1C,0x00,0x00,0x00,0x00,0x00, // /

0x00,0x00,0x00,0x00,0x0F,0xF0,0x1C,0x38,0x1C,0x78,0x1C,0xF8,0x1C,0xF8,0x1D,0xB8,0x1D,0xB
8,0x1F,0x38,0x1F,0x38,0x1E,0x38,0x1C,0x38,0x0F,0xF0,0x00,0x00,0x00,0x00, // 0
0x00,0x00,0x00,0x00,0x01,0x80,0x01,0x80,0x03,0x80,0x1F,0x80,0x1F,0x80,0x03,0x80,0x03,0x8
0,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x1F,0xF0,0x00,0x00,0x00,0x00, // 1
0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x70,0x1C,0x38,0x00,0x38,0x00,0x70,0x00,0xE0,0x01,0xC
0,0x03,0x80,0x07,0x00,0x0E,0x38,0x1C,0x38,0x1F,0xF8,0x00,0x00,0x00,0x00, // 2
0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x70,0x1C,0x38,0x00,0x38,0x00,0x70,0x03,0xC0,0x03,0xC
0,0x00,0x70,0x00,0x38,0x1C,0x38,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // 3
0x00,0x00,0x00,0x00,0x00,0xE0,0x01,0xE0,0x03,0xE0,0x06,0xE0,0x0C,0xE0,0x18,0xE0,0x1F,0xF
8,0x1F,0xF8,0x00,0xE0,0x00,0xE0,0x00,0xE0,0x03,0xF8,0x00,0x00,0x00,0x00, // 4
0x00,0x00,0x00,0x00,0x1F,0xF8,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1F,0xE0,0x1F,0xF
0,0x00,0x78,0x00,0x38,0x1C,0x38,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // 5
0x00,0x00,0x00,0x00,0x03,0xE0,0x07,0x00,0x0E,0x00,0x1C,0x00,0x1C,0x00,0x1F,0xF0,0x1F,0xF
8,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x0F,0xF0,0x00,0x00,0x00,0x00, // 6
```

```
0x00,0x00,0x00,0x00,0x1F,0xFC,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x00,0x1C,0x00,0x38,0x00,0x7
0,0x00,0xE0,0x01,0xC0,0x03,0x80,0x03,0x80,0x03,0x80,0x00,0x00,0x00,0x00, // 7
0x00,0x00,0x00,0x00,0x0F,0xF0,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1F,0x38,0x07,0xE0,0x07,0xE
0,0x1C,0xF8,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x0F,0xF0,0x00,0x00,0x00,0x00, // 8
0x00,0x00,0x00,0x00,0x0F,0xF0,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1F,0xF8,0x0F,0xF
8,0x00,0x38,0x00,0x38,0x00,0x70,0x00,0xE0,0x07,0xC0,0x00,0x00,0x00,0x00, // 9
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x80,0x03,0x80,0x03,0x80,0x00,0x00,0x00,0x0
0,0x03,0x80,0x03,0x80,0x03,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // :
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x80,0x03,0x80,0x03,0x80,0x00,0x00,0x00,0x0
0,0x03,0x80,0x03,0x80,0x03,0x80,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // ;
0x00,0x00,0x00,0x70,0x00,0xE0,0x01,0xC0,0x03,0x80,0x07,0x00,0x0E,0x00,0x1C,0x00,0x1C,0x0
0,0x0E,0x00,0x07,0x00,0x03,0x80,0x01,0xC0,0x00,0xE0,0x00,0x70,0x00,0x00, // <
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3F,0xFC,0x3F,0xFC,0x00,0x00,0x00,0x0
0,0x3F,0xFC,0x3F,0xFC,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // =
0x00,0x00,0x1C,0x00,0x0E,0x00,0x07,0x00,0x03,0x80,0x01,0xC0,0x00,0xE0,0x00,0x70,0x00,0x7
0,0x00,0xE0,0x01,0xC0,0x03,0x80,0x07,0x00,0x0E,0x00,0x1C,0x00,0x00,0x00, // >
0x00,0x00,0x03,0xC0,0x0F,0xF0,0x1E,0x78,0x18,0x38,0x00,0x38,0x00,0x70,0x00,0xE0,0x01,0xC
0,0x01,0xC0,0x00,0x00,0x00,0x00,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x00,0x00, // ?

0x00,0x00,0x0F,0xF8,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0xFC,0x1C,0xFC,0x1C,0xF
C,0x1C,0xFC,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1F,0xF0,0x07,0xF8,0x00,0x00, // @
0x00,0x00,0x00,0x00,0x03,0xC0,0x07,0xE0,0x0E,0x70,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x3
8,0x1F,0xF8,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x00,0x00,0x00,0x00, // A
0x00,0x00,0x00,0x00,0x1F,0xF0,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0F,0xF0,0x0F,0xF
0,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x1F,0xF0,0x00,0x00,0x00,0x00, // B
0x00,0x00,0x00,0x00,0x07,0xF0,0x0E,0x38,0x1C,0x38,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1C,0x0
0,0x1C,0x00,0x1C,0x00,0x1C,0x38,0x0E,0x38,0x07,0xF0,0x00,0x00,0x00,0x00, // C
0x00,0x00,0x00,0x00,0x1F,0xE0,0x0E,0x70,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x3
8,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x70,0x1F,0xE0,0x00,0x00,0x00,0x00, // D
0x00,0x00,0x00,0x00,0x1F,0xF8,0x0E,0x18,0x0E,0x08,0x0E,0x00,0x0E,0x30,0x0F,0xF0,0x0F,0xF
0,0x0E,0x30,0x0E,0x00,0x0E,0x08,0x0E,0x18,0x1F,0xF8,0x00,0x00,0x00,0x00, // E
0x00,0x00,0x00,0x00,0x1F,0xF8,0x0E,0x18,0x0E,0x08,0x0E,0x00,0x0E,0x30,0x0F,0xF0,0x0F,0xF
0,0x0E,0x30,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x1F,0x00,0x00,0x00,0x00,0x00, // F
0x00,0x00,0x00,0x00,0x07,0xF0,0x0E,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x00,0x1C,0x00,0x1C,0x0
0,0x1C,0xF8,0x1C,0x38,0x1C,0x38,0x0E,0x38,0x07,0xF8,0x00,0x00,0x00,0x00, // G
0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1F,0xF0,0x1F,0xF
0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x00,0x00,0x00,0x00, // H
0x00,0x00,0x00,0x00,0x0F,0xE0,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x8
0,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x0F,0xE0,0x00,0x00,0x00,0x00, // I
0x00,0x00,0x00,0x00,0x01,0xFC,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x7
0,0x38,0x70,0x38,0x70,0x38,0x70,0x38,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // J
0x00,0x00,0x00,0x00,0x1E,0x38,0x0E,0x38,0x0E,0x70,0x0E,0xE0,0x0F,0xC0,0x0F,0x80,0x0F,0x8
0,0x0F,0xC0,0x0E,0xE0,0x0E,0x70,0x0E,0x38,0x1E,0x38,0x00,0x00,0x00,0x00, // K
0x00,0x00,0x00,0x00,0x1F,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x0
0,0x0E,0x00,0x0E,0x08,0x0E,0x18,0x0E,0x38,0x1F,0xF8,0x00,0x00,0x00,0x00, // L
0x00,0x00,0x00,0x00,0x1C,0x1C,0x1E,0x3C,0x1F,0x7C,0x1F,0xFC,0x1F,0xFC,0x1D,0xDC,0x1C,0x9
C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x00,0x00,0x00,0x00, // M
0x00,0x00,0x00,0x00,0x1C,0x1C,0x1C,0x1C,0x1E,0x1C,0x1F,0x1C,0x1F,0x9C,0x1D,0xDC,0x1C,0xF
C,0x1C,0x7C,0x1C,0x3C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x00,0x00,0x00,0x00, // N
0x00,0x00,0x00,0x00,0x03,0xE0,0x07,0xF0,0x0E,0x38,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1
C,0x1C,0x1C,0x1C,0x1C,0x0E,0x38,0x07,0xF0,0x03,0xE0,0x00,0x00,0x00,0x00, // O

0x00,0x00,0x00,0x00,0x1F,0xF0,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0F,0xF0,0x0F,0xF
0,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x1F,0x00,0x00,0x00,0x00,0x00, // P
0x00,0x00,0x00,0x00,0x03,0xE0,0x0F,0x78,0x0E,0x38,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1
C,0x1C,0x7C,0x1C,0xFC,0x0F,0xF8,0x0F,0xF8,0x00,0x38,0x00,0xFC,0x00,0x00, // Q
0x00,0x00,0x00,0x00,0x1F,0xF0,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0F,0xF0,0x0F,0xF
0,0x0E,0x70,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x1E,0x38,0x00,0x00,0x00,0x00, // R
0x00,0x00,0x00,0x00,0x0F,0xF0,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x00,0x0F,0xE0,0x07,0xF
0,0x00,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x0F,0xF0,0x00,0x00,0x00,0x00, // S
0x00,0x00,0x00,0x00,0x1F,0xFC,0x19,0xCC,0x11,0xC4,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC
0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x07,0xF0,0x00,0x00,0x00,0x00, // T
0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x7
0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // U
0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x7
0,0x1C,0x70,0x1C,0x70,0x0E,0xE0,0x07,0xC0,0x03,0x80,0x00,0x00,0x00,0x00, // V
0x00,0x00,0x00,0x00,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x9C,0x1C,0x9
C,0x1C,0x9C,0x0F,0xF8,0x0F,0xF8,0x07,0x70,0x07,0x70,0x00,0x00,0x00,0x00, // W
0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0E,0xE0,0x07,0xC0,0x03,0x80,0x03,0x8
0,0x07,0xC0,0x0E,0xE0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x00,0x00,0x00,0x00, // X
0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0E,0xE0,0x07,0xC
0,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x0F,0xE0,0x00,0x00,0x00,0x00, // Y
0x00,0x00,0x00,0x00,0x1F,0xF8,0x1C,0x38,0x18,0x38,0x10,0x70,0x00,0xE0,0x01,0xC0,0x03,0x8
0,0x07,0x00,0x0E,0x08,0x1C,0x18,0x1C,0x38,0x1F,0xF8,0x00,0x00,0x00,0x00, // Z
0x00,0x00,0x00,0x00,0x07,0xF0,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x0
0,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0xF0,0x00,0x00,0x00,0x00, // [
0x00,0x00,0x00,0x00,0x10,0x00,0x18,0x00,0x1C,0x00,0x0E,0x00,0x07,0x00,0x03,0x80,0x01,0xC
0,0x00,0xE0,0x00,0x70,0x00,0x38,0x00,0x1C,0x00,0x07,0x00,0x00,0x00,0x00, // <Backslash>
```

```
0x00,0x00,0x00,0x00,0x07,0xF0,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x7
0,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x07,0xF0,0x00,0x00,0x00,0x00, // ]
0x00,0x00,0x01,0x80,0x03,0xC0,0x07,0xE0,0x0E,0x70,0x1C,0x38,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // ^
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0xFF,0x7F,0xFF, // _

0x00,0x00,0x00,0x00,0x1C,0x00,0x1C,0x00,0x07,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // '
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xE0,0x00,0x70,0x00,0x7
0,0x0F,0xF0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0F,0xD8,0x00,0x00,0x00,0x00, // a
0x00,0x00,0x00,0x00,0x1E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0F,0xF0,0x0E,0x38,0x0E,0x3
8,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x1B,0xF0,0x00,0x00,0x00,0x00, // b
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x70,0x1C,0x7
0,0x1C,0x00,0x1C,0x00,0x1C,0x70,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // c
0x00,0x00,0x00,0x00,0x00,0xF8,0x00,0x70,0x00,0x70,0x00,0x70,0x0F,0xF0,0x1C,0x70,0x1C,0x7
0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0F,0xD8,0x00,0x00,0x00,0x00, // d
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x70,0x1C,0x7
0,0x1F,0xF0,0x1C,0x00,0x1C,0x70,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // e
0x00,0x00,0x00,0x00,0x03,0xE0,0x07,0x70,0x07,0x70,0x07,0x00,0x07,0x00,0x1F,0xE0,0x1F,0xE
0,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x1F,0xC0,0x00,0x00,0x00,0x00, // f
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xD8,0x1C,0x70,0x1C,0x7
0,0x1C,0x70,0x1C,0x70,0x0F,0xF0,0x07,0xF0,0x00,0x70,0x1c,0x70,0x0F,0xE0, // g
0x00,0x00,0x00,0x00,0x1E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0xF0,0x0F,0x38,0x0F,0x3
8,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x1E,0x38,0x00,0x00,0x00,0x00, // h
0x00,0x00,0x00,0x00,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x00,0x00,0x0F,0xC0,0x01,0xC0,0x01,0xC
0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x0F,0xF8,0x00,0x00,0x00,0x00, // i
0x00,0x00,0x00,0x00,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x00,0x03,0xF0,0x00,0x70,0x00,0x7
0,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x1C,0x70,0x0C,0xF0,0x07,0xE0, // j
0x00,0x00,0x00,0x00,0x1E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x38,0x0E,0x70,0x0E,0xE
0,0x0F,0xC0,0x0E,0xE0,0x0E,0x70,0x0E,0x38,0x1E,0x38,0x00,0x00,0x00,0x00, // k
0x00,0x00,0x00,0x00,0x0F,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC
0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x0F,0xF8,0x00,0x00,0x00,0x00, // l
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0xF8,0x1C,0x9C,0x1C,0x9
C,0x1C,0x9C,0x1C,0x9C,0x1C,0x9C,0x1C,0x9C,0x1C,0x9C,0x00,0x00,0x00,0x00, // m
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0xE0,0x1C,0x70,0x1C,0x7
0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x00,0x00,0x00,0x00, // n
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x70,0x1C,0x7
0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // o

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1B,0xF0,0x0E,0x38,0x0E,0x3
8,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0F,0xF0,0x0E,0x00,0x0E,0x00,0x1F,0x00, // p
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0xB0,0x38,0xE0,0x38,0xE
0,0x38,0xE0,0x38,0xE0,0x38,0xE0,0x1F,0xE0,0x00,0xE0,0x00,0xE0,0x01,0xF0, // q
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1E,0xF0,0x0F,0xF8,0x0F,0x3
8,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x1F,0x00,0x00,0x00,0x00,0x00, // r
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x30,0x1C,0x3
0,0x0F,0x80,0x03,0xE0,0x18,0x70,0x18,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // s
0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x03,0x00,0x07,0x00,0x1F,0xF0,0x07,0x00,0x07,0x0
0,0x07,0x00,0x07,0x00,0x07,0x70,0x07,0x70,0x03,0xE0,0x00,0x00,0x00,0x00, // t
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x7
0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0F,0xD8,0x00,0x00,0x00,0x00, // u
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x7
0,0x1C,0x70,0x1C,0x70,0x0E,0xE0,0x07,0xC0,0x03,0x80,0x00,0x00,0x00,0x00, // v
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x1C,0x1C,0x1C,0x1C,0x1
C,0x1C,0x9C,0x1C,0x9C,0x0F,0xF8,0x07,0x70,0x07,0x70,0x00,0x00,0x00,0x00, // w
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0xE0,0x1C,0xE0,0x0F,0xC
0,0x07,0x80,0x07,0x80,0x0F,0xC0,0x1C,0xE0,0x1C,0xE0,0x00,0x00,0x00,0x00, // x
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0E,0x38,0x0E,0x38,0x0E,0x3
8,0x0E,0x38,0x0E,0x38,0x07,0xF0,0x03,0xE0,0x00,0xE0,0x01,0xC0,0x1F,0x80, // y
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0xE0,0x18,0xE0,0x11,0xC
0,0x03,0x80,0x07,0x00,0x0E,0x20,0x1C,0x60,0x1F,0xE0,0x00,0x00,0x00,0x00, // z
0x00,0x00,0x00,0x00,0x01,0xF8,0x03,0x80,0x03,0x80,0x03,0x80,0x07,0x00,0x1C,0x00,0x1C,0x0
0,0x07,0x00,0x03,0x80,0x03,0x80,0x03,0x80,0x01,0xF8,0x00,0x00,0x00,0x00, // {
0x00,0x00,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC
0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x00,0x00,0x00, // |
0x00,0x00,0x00,0x00,0x1F,0x80,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x00,0xE0,0x00,0x38,0x00,0x3
8,0x00,0xE0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x1F,0x80,0x00,0x00,0x00,0x00, // }
0x00,0x00,0x00,0x00,0x1F,0x1C,0x3B,0x9C,0x39,0xDC,0x38,0xF8,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // ~
0x00,0x00,0x00,0x00,0x03,0xE0,0x07,0xF0,0x0E,0x38,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1
C,0x1C,0x1C,0x1C,0x1C,0x0E,0x38,0x3E,0x3E,0x3E,0x3E,0x00,0x00,0x00,0x00, //Simbolo ohmio
127
0x00,0x00,0x00,0x00,0x00,0xF0,0x01,0xF8,0x01,0x98,0x01,0x98,0x01,0x98,0x01,0xF8,0x00,0xF
0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //Simbolo °
GRADO
0x7F,0xFE,0xFF,0xFF,0xC1,0x83,0xC0,0x03,0xC1,0x83,0xC1,0x83,0xC1,0x83,0xE1,0xE7,0xE1,0xE
7,0xC0,0x03,0xC0,0x03,0xC0,0x03,0xC0,0x03,0xC1,0x83,0xFF,0xFF,0x7F,0xFE, //Simbolo reloj
```

```
0x01,0x80,0x61,0x86,0x70,0x0E,0x33,0xCC,0x07,0xE0,0x0F,0xF0,0x1F,0xF8,0xDF,0xFB,0xDF,0xF
B,0x1F,0xF8,0x0F,0xF0,0x07,0xE0,0x33,0xCC,0x70,0x0E,0x61,0x86,0x01,0x80, //Simbolo
brillo
0x03,0xC0,0x06,0x60,0x04,0x3E,0x04,0x20,0x04,0x3C,0x05,0xA0,0x05,0xB8,0x05,0xA0,0x0D,0xB
0,0x19,0x98,0x33,0xCC,0x37,0xEC,0x33,0xCC,0x19,0x98,0x0C,0x30,0x07,0xE0, //Simbolo
termometro
0x00,0x00,0x00,0x00,0x07,0x80,0x04,0x40,0x02,0x20,0x01,0x10,0x7F,0x08,0x40,0x04,0x40,0x0
4,0x7F,0x08,0x01,0x10,0x02,0x20,0x04,0x40,0x07,0x80,0x00,0x00,0x00,0x00, //Simgolo
flecha ->
0x07,0xE0,0x18,0x18,0x20,0x04,0x40,0x02,0x40,0x02,0x80,0x01,0x8F,0xF1,0x80,0x01,0x8D,0xB
1,0x80,0x01,0x80,0x01,0x40,0x02,0x40,0x02,0x20,0x04,0x18,0x18,0x07,0xE0, //Simbolo
corriente continua
0x00,0x38,0x00,0xC8,0x01,0x10,0x02,0x20,0x04,0x40,0x08,0x80,0x10,0x78,0x10,0x04,0x0F,0x0
4,0x00,0x08,0x01,0x10,0x02,0x20,0x04,0x40,0x09,0x80,0x12,0x00,0x1C,0x00, //Simbolo
tension
0xFF,0xFF,0x80,0x01,0x9F,0xF9,0x9F,0xF9,0x80,0x01,0x8F,0xF1,0x8F,0xF1,0x80,0x01,0x80,0x0
1,0x8F,0xF1,0x8F,0xF1,0x80,0x01,0x9F,0xF9,0x9F,0xF9,0x80,0x01,0xFF,0xFF  //Simbolo USB
};

/*Estructura de fuente de los caracteres*/
struct _current_font
{
      unsigned char x_size;//Tamaño en X de la caracter
      unsigned char y_size;//Tamaño en Y de la caracter
      unsigned char x_step;//Distancia de representado entre un
caracter y otro
      unsigned char offset;//offset necesario para restar al caracter
a representar para su correcta seleccion de bits en la fuente
      unsigned char numchars;//numero de caracteres
      unsigned char font[];//fuente para cada caracter
};
struct _current_font *CF = (struct _current_font*) BigFont;
/*Funcion de transmision de datos mediante SPI*/
uint8_t SPI1_Transmision(uint8_t dato)
{
    while( SPI1STATLbits.SPITBF == true ) ;
    SPI1BUFL = dato;
    while ( SPI1STATLbits.SPIRBE == true);
    return SPI1BUFL;
}
/*Funcion Para la transmision de comandos mediante SPI a la TFT
 *cmd=Comando a enviar*/
void writeCommand(uint8_t cmd)
{
    DC_SetLow();//DC en baja para que el controlador de la TFT
identifique los datos que le llegan como comandos
    SPI1_Transmision(cmd);//Transmision del comando
    DC_SetHigh();//DC en alta para la finalizacion de transmision de
comandos
}
/*Comandos de inicializacion de TFT*/
const uint8_t
  cmd_240x240[] =  {     // Init commands for 7789 screens
    9,                   //  9 commands in list:
    ST77XX_SWRESET,   ST_CMD_DELAY, //  1: Software reset, no args,
w/delay
      150,                          //    150 ms delay
    ST77XX_SLPOUT ,   ST_CMD_DELAY, //  2: Out of sleep mode, no args,
w/delay
      255,                          //     255 = 500 ms delay
    ST77XX_COLMOD , 1+ST_CMD_DELAY, //  3: Set color mode, 1 arg +
delay:
      0x55,                         //     16-bit color
      10,                           //     10 ms delay
    ST77XX_MADCTL , 1,   //  4: Mem access ctrl (directions), 1 arg:
```

```
      0x00,               //     Row/col addr, bottom-top refresh
//0x80
    ST77XX_CASET  , 4,    //  5: Column addr set, 4 args, no delay:
      0, 0,               //     XSTART = 0
      0, 240,             //     XEND = 240
    ST77XX_RASET  , 4,    //  6: Row addr set, 4 args, no delay:
      0, 0,               //     YSTART = 0
      0, 240,             //     YEND = 240
    ST77XX_INVON  , 0,    //  7: hack
    ST77XX_NORON  , 0,    //  8: Normal display on, no args, w/delay
    ST77XX_DISPON , 0,    //  9: Main screen turn on, no args, delay
    };
/*Funcion de transmision de comandos a la TFT
 *addr=puntero de direccion del comando*/
void displayInit(const uint8_t *addr)
{
    uint8_t  numCommands, numArgs;
    uint16_t ms;

    numCommands = *addr++;   // Numero de comandos a seguir
    while(numCommands--)     // Se ejecuta hasta que todos los
comandos se envien
    {
      SS1OUT_SetLow();  //
      writeCommand(*addr++); // Transmision de comando a la TFT
      numArgs  = *addr++;    // Numero de argumentos a seguir
      ms       = numArgs & ST_CMD_DELAY;   // Si hay un retardo
habilitado en cmd_240x240[], ms en alta
      numArgs &= ~ST_CMD_DELAY;            // Bit de retardo de
mascara
      while(numArgs--) {                   // Para cada argumento
        SPI1_Transmision(*addr++);   // Transmision de cada argumento
a la TFT
      }
      SS1OUT_SetHigh();
      if(ms) //Si se ha habilitado un retardo
      {
        ms = *addr++; // Tiempo de retardo
        if(ms == 255) ms = 500;     // Si es 255, retardo de 500 ms.
No se almacena en cmd_240x240[] el valor de 500, ya que es una
variable de 8 bits
        __delay_ms(ms);
      }
    }
}
/*Funcion inicializacion de pantalla TFT*/
void tft_init(void)
{
    /*Reset mediante hardware, con un minimo de 1 ms en baja*/
    RST_SetLow();
    __delay_ms(1);
    RST_SetHigh();
    __delay_ms(120);//retardo necesario de 120 ms antes de enviar
cualquier comando despues de un reset por hardware
    displayInit(cmd_240x240);//Comandos de inicializacion
    /*SS1OUT_SetLow();
    writeCommand(ST77XX_MADCTL);
    SPI1_Transmision(ST77XX_MADCTL_RGB);
    SS1OUT_SetHigh();*/
}
int offsety=0,offsetx=0;//Variables necesarias cuando la rotacion
varía
```

```
/*Funcion para variar la rotacion de la pantalla
 *x=orientacion*/
void setRotation(int  x) {
    uint8_t madctl = 0;
     switch (x) {
        case 0:
            /*Posicion de 270°*/
            madctl  = ST77XX_MADCTL_MX | ST77XX_MADCTL_MY |
ST77XX_MADCTL_RGB;
            offsetx=0;
            offsety=80;
          break;
         case 1:
            /*Posicion de 180°*/
            madctl  = ST77XX_MADCTL_MY | ST77XX_MADCTL_MV |
ST77XX_MADCTL_RGB;
            offsetx=80;
            offsety=0;
          break;
       case 2:
            /*Posicion principal 90°*/
            madctl  = ST77XX_MADCTL_RGB;
            offsetx=0;
            offsety=0;
          break;
        case 3:
            /*Posicion de 0°*/
            madctl  = ST77XX_MADCTL_MX | ST77XX_MADCTL_MV |
ST77XX_MADCTL_RGB;
            offsetx=0;
            offsety=0;
          break;
  }
    SS1OUT_SetLow();//Chip select en baja para poder enviar datos al
controlador de la TFT
    writeCommand(ST77XX_MADCTL);//Transmision del comando para cambiar
la rotacion de la pantalla
    SPI1_Transmision(madctl);//Valores de comando que indican donde
empieza el eje X y el eje Y
    SS1OUT_SetHigh();//Chip select en alta, fin de transmision de
datos con el controlador de la TFT
}
/*Funcion para enviar los comandos y las coordenadas X e Y para
representar en pantalla
 *x=posicion coordenada X
 *y=pocision coordenada Y
 *w=ancho
 *h=alto*/
void comandoXY (int x, int y,int w, int h)
{
    SS1OUT_SetLow();
    writeCommand(ST77XX_CASET); //Comando establecimiento direccion de
columna
    SPI1_Transmision(x>>8);//Principio de trama
    SPI1_Transmision(x);//Fin de trama
    SPI1_Transmision(w>>8);//Principio de trama
    SPI1_Transmision(w);//Fin de trama
    SS1OUT_SetHigh();

    SS1OUT_SetLow();
    writeCommand(ST77XX_RASET); //Comando establecimiento direccion de
fila
```

```
    SPI1_Transmision(y>>8);
    SPI1_Transmision(y);
    SPI1_Transmision(h>>8);
    SPI1_Transmision(h);
    SS1OUT_SetHigh();

    SS1OUT_SetLow();
    writeCommand(ST77XX_RAMWR); //Comando de escritura en RAM de la
TFT
}
/*Funcion para representar en pantalla un cuadrado/rectangulo
 *x=posicion coordenada X
 *y=pocision coordenada Y
 *w=ancho
 *h=alto
 *color=color en el que se representa*/
void dimensiones (int x, int y, int w, int h, uint32_t color)
{
    /*Offsets necesarios dependiendo de la rotacion establecida*/
    y+=offsety;
    x+=offsetx;
    comandoXY(x,y,(x+w-1),(y+h-1));//Establecimiento de la zona a
representar en pantalla
    uint16_t py = w*h;//Numero de pixeles que se representan en dicha
zona
    while (py--) { //Envia el color de cada pixel
      SPI1_Transmision(color >> 8);
      SPI1_Transmision(color );
    }
    /*tras llamar a comandoXY() deja chip select en baja, puesto que a
continuacion siempre se envia la informacion correspondiente a cada
pixel*/
    SS1OUT_SetHigh();//Fin de transmision de datos al controlador de
la TFT
}
/*Representacion de caracter pequeño
 *x=posicion coordenada X
 *y=pocision coordenada Y
 *c=caracter*/
void TFT_putch(int x, int y, unsigned char c)
{ // Faster version (now 197 ut, before 2312 ut)
    int j, d, org, i, n;
    /*Offsets necesarios dependiendo de la rotacion establecida*/
    y+=offsety;
    x+=offsetx;
    n = CF->x_size*CF->y_size>>3;//numero de bytes de informacion por
caracter (32)
    org = (c - CF->offset)*n;//direccion de la fuente donde se
encuentran los bytes del caracter
    comandoXY(x,y,(x+(CF->x_size-1)),(y+(CF->y_size-
1)));//Establecimiento de la zona del caracter con un tamaño de 16x16
    for(i=0;i<n;i++)
    {
      d = CF->font[ org++ ];//Cada byte de informacion
      for(j=0;j<8;j++)//Envia la informacion del color de cada uno de
los 8 bits que contine un byte
      {
       if(d&0x80) {//cuando el bit es 1, color frontal
       SPI1_Transmision(FG_Color >> 8);
       SPI1_Transmision(FG_Color );
       }
       else {  //cuando el bit es 0, color fondo
```

```
        SPI1_Transmision(BG_Color >> 8);
        SPI1_Transmision(BG_Color);
        }
        d = (d<<1);//Se mueve un bit a la izquierda para analizar el
siguiente bit menos significativo
      }
    }
    /*Fin de comunicacion SPI con la TFT*/
    SS1OUT_SetHigh();
}
/*Representacion de caracter grande
 *x=posicion coordenada X
 *y=pocision coordenada Y
 *c=caracter*/
void TFT_putchBIG(int x, int y, unsigned char c)
{
    int j, d, org, i, n, z=0,org1;
    /*Offsets necesarios dependiendo de la rotacion establecida*/
    y+=offsety;
    x+=offsetx;
    n = CF->x_size*CF->y_size>>3;//numero de bytes de informacion por
caracter (32)
    org = (c - CF->offset)*n;//direccion de la fuente donde se
encuentran los bytes del caracter
    org1 = org;//direccion necesaria para la repeticion de un bloque
de 2 bytes
    comandoXY(x,y,(x+((CF->x_size*2)-1)),(y+((CF->y_size*2)-
1)));//Establecimiento de la zona del caracter con un tamaño de 32x32
    for(i=0;i<(n*2);i++)//Debido a que el tamaño aumenta al doble, el
bucle se repite el doble de veces 32x2=64
    {
      /*Cada dos bytes de informacion, se vuelven a enviar dichos
bytes, de manera que se duplica la <altura> del caracter, 16x2=32*/
      if(z<2)d = CF->font[ org++ ];//Byte de informacion
      else if (z>=2)d = CF->font[ org1++ ];//Byte de informacion
repetido al anterior
      for(j=0;j<8;j++)
      {
       /*Se duplica la transmision de informacion por bit, de esta
manera se duplica la <anchura> de caracter, 16x2=32*/
       if(d&0x80) {//cuando el bit es 1, color frontal
       SPI1_Transmision(FG_Color >> 8);
       SPI1_Transmision(FG_Color );
       SPI1_Transmision(FG_Color >> 8);
       SPI1_Transmision(FG_Color );
       }
       else {  //cuando el bit es 0, color fondo
       SPI1_Transmision(BG_Color >> 8);
       SPI1_Transmision(BG_Color);
       SPI1_Transmision(BG_Color >> 8);
       SPI1_Transmision(BG_Color);
       }
       d = (d<<1);//Se mueve un bit a la izquierda para analizar el
siguiente bit menos significativo
      }
      z++;//incremento de variable bandera para controlar el envio de
2 bytes
      if(z==4)z=0;//una vez se envian los bytes repetidos, se reinicia
a z=0 para continuar
    }
    /*Fin de comunicacion SPI con la TFT*/
    SS1OUT_SetHigh();
```

```
}
/*Representacion de cadenas de caracteres de tamaño pequeño
 *x=posicion coordenada X
 *y=pocision coordenada Y
 *s=cadena de caracteres*/
void TFT_print(int x, int y, char * s)
{
 int i=0;
 /*bucle de representacion para cada caracter de una cadena con un
salto de CF->x_step*/
 while(s[i]){ TFT_putch(x+i*CF->x_step, y,s[i]); i++; }
}
/*Representacion de cadenas de caracteres de tamaño grande
 *x=posicion coordenada X
 *y=pocision coordenada Y
 *s=cadena de caracteres*/
void TFT_printBIG(int x, int y, char * s)
{
 int i=0;
  /*bucle de representacion para cada caracter de una cadena con un
salto de (CF->x_step*2)*/
 while(s[i]){ TFT_putchBIG(x+i*2*CF->x_step, y,s[i]);i++;}
}

int rotacion=2;//rotacion de la pantalla
int brightness=1;//nivel de brillo de la pantalla
int uno=0,dos=0,tres=1;//variables para actualizar grafica y valores
int count=0,count1=0;//contadores para la interrupcion cada 10 ms
int m=0,zz1=0;//m=Posicion de pantalla/zz1=grados centigrados o
fahrenheit
unsigned int long
stemp,stemp1=0,stemp2=0,vin,vin1=0,vin2,ibat,vibat1=0,vibat2;//sensor
de temperatura/tension aplicada en el USB/corriente de consumo que
circula a traves del hardware
unsigned int long  watts,mwattsh,mamph,res;//vatios/mili vatios
hora/mili amperios hora/resistencia
unsigned int       segs =0, min= 0, hora=0,segs1,min1,hora1;//reloj
char c[6],d[5],f[7],p[8];// variables de representacion de datos
char c1[7],e1[7],f1[8],p1[9],ph1[9],ah1[9],r1[9];//variables globales
para al comparar con su par evitar dibujar si el valor no varía
char v[6];//cadena para representacion del eje de ordenadas

/*Calculo de valores de parametros a analizar por el hardware*/
void valores(void)
{
    float procesamiento;//variable para el calculo de ecuaciones con
valores tipo float

/*_________________________________tension_____________________________
__*///valor maximo 25357 ->25.357V
    /*Puesto que el valor maximo del ADC es 3.3V, y tenemos una
resolucion del 12 bits,
     *esto se traduce en 3.3/((2^12)-1) = 3.3 /4095= 0.000805 V de
resolución.
     *Si vin2*0.000805=(R4/R4+R3)*Vin ->
vin2*0.000805=(3300/3300+22000)*Vin -> Vin = vin2*0.000805 /0.130
     *Y lo multiplicamos por 1000 para su analisis para mostrar por
pantalla*/
    procesamiento=(vin2*0.000805L/0.130)*1000L;
    vin= (unsigned int long) procesamiento;//pasamos el valor entero a
vin
```

```
/*_____________________________corriente_____________________________
___*///valor maximo 6037 -> 6.037A
    /*Acorde al circuito con realimentacion negativa, funcion de
transferencia es
     *vibat2*0.000805=R2/R1*VR0  ->  VR0=(vibat2*0.000805L)/R2/R1 ->
VR0=(vibat2*0.000805)/(8200/1500)
     *como lo que nos interesa es la corriente que circula por R0=0.1
Ohmnio
     *ibat= VR0/0.1 -> ibat=((vibat2*0.000805)/(8200/1500))/0.1 y se
multiplica por 1000 para su analisis
     *y tener precision de milesimas de Amp, siendo:*/
    procesamiento=((vibat2*0.000805L)/(8200L/1500L))*10000L;
    ibat= (unsigned int long) procesamiento;//pasamos el valor entero
a ibat

/*_____________________________potencia_____________________________
___*///valor maximo 153080 -> 153.08W
    /*Funcion de transferencia P=V*I que para su analisis y con
nuestras variables es lo siguiente*/
    watts=(vin*ibat)/1000L;//resultado real multiplicado por 1.000
para su analisis

/*_____________________________milivatioshora_____________________________
___*/
     /*mWh = watts *(0.5s/3600s/1h), que sería
     * mWh = watts *(0.000138)h, para evitar perdida de información
     * haremos lo siguiente:
     * mWh = watts1 *((0.5*10.000.000)s/3600s/1h)
     * mWh = watts1 *(1388)h
     * A la hora de almacenarlo, dividiremos por 10.000
     * mWh = (watts *1388h)/10.000
     * Y almacenaremos los mWh multiplicados por 1.000, pudiendo
      almacenar hasta
     * hasta 2^32/1.000=4294967,296 mWh
     * pero el valor maximo mostrado será de 99999mWh */
    if(mwattsh<99999000L)mwattsh=mwattsh + ((watts
*1388L)/10000L);//resultado real multiplicado por 1.000

/*_____________________________miliamperioshora_____________________________
___*/
    /*Puesto que la muestra es cada 0.5 segundos, y los mAh son:
     * mAh = mA * (0.5s/3600s/1h), que sería
     * mAh = mA * (0.000138)h, para evitar perdida de información
     * haremos lo siguiente:
     * mAh = mA * ((0.5*10.000.000)s/3600s/1h)
     * mAh = mA * (1388)h
     * A la hora de almacenarlo final, dividiremos por 10.000
     * mAh = (mA * (1388)h)/10.000
     * Y almacenaremos los mAh multiplicados por 1.000, pudiendo
      almacenar hasta
     * hasta 2^32/1.000=4294967,296 mAh
     * pero el valor maximo mostrado será de 99.999mAh*/
    if(mamph<99999000L)mamph=mamph + ((ibat*1388L)/10000L);//resultado
real multiplicado por 1.000
}
/*Representa la tension de alimencion del USB con 4 digitos y una
precision de centesimas
 *x=posicion coordenada X
 *y=pocision coordenada Y */
void valortension(int x, int y)//GREEN
{
```

```
    FG_Color = GREEN;//Color
    unsigned int long analisis;
    analisis=vin;
    if(analisis<=25357)
    {
            //se le suma 48 al valor entero para convertirlo en
caracter
            c[0]= (analisis/10000)+48;//obtenemos el primer
digito(Decena)
            analisis= analisis-((c[0]-48)*10000);//Restamos el primer
digito multiplicado por 10000
            c[1]= (analisis/1000)+48; //obtenemos el segundo
digito(Unidad))
            analisis= analisis-((c[1]-48)*1000);//Resta del segundo
digito multiplicado por 1000
            c[2]= '.';//caracter a representar
            c[3]= (analisis/100)+48;//obtenemos el tercer digito
(Decimas)
            analisis= analisis-((c[3]-48)*100);//Restamos el tercer
digito multiplicado por 100
            c[4]= (analisis/10)+48; //obtenemos el cuarto
digito(Centesimas)
            c[5]= 'V';//caracter a representar
            c[6]= 0;//fin de cadena de caracteres
            /*Si ambas cadenas son diferentes, se representa por
pantalla, en caso de ser iguales se evita representar de nuevo*/
            if(strcmp(c,c1)!=0)
            {
               TFT_printBIG(x, y, c);
               strcpy(c1,c); //copiamos la cadena c en c1
            }
    }
        else if(analisis>=25357) //Se ha producido un error
        {
           TFT_printBIG(x, y, "ERROR!");
        }
}
/*Representa la tension de D+ y D- con 3 digitos y una precision de
centesimas
 *x=posicion coordenada X
 *y=pocision coordenada Y
 *z=1 valor de D+ , z=0 valor de D-*/
void tensiondatos(int x, int y, int z)//WHITE
{
    FG_Color = WHITE;
    unsigned int long dvalor;
    /*Calculo de tension de D+ y D- a partir del valor obtenido del
ADC, dependiendo del valor de Z */
    if(z == 1)//D+
    {
        dvalor= (ADC1BUF0* 805L)/1000L;//Valor entero multiplicado por
1000
    }
        else if(z == 0)//D-
        {
           dvalor= (ADC1BUF1* 805L)/1000L;//Valor entero multiplicado
por 1000
        }
    if(dvalor<=3296)
    {
        d[0]= (dvalor/1000)+48;//unidad
        dvalor= dvalor-((d[0]-48)*1000);
```

```
        d[1]= '.';
        d[2]= (dvalor/100)+48;//decima
        dvalor= dvalor-((d[2]-48)*100);
        d[3]= (dvalor/10)+48;//centesima
        d[4]= 'V';
        d[5]=0;
        TFT_print(x, y, d);
    }
         else if(dvalor>3296)//Se ha producido un error
         {
            TFT_print(x, y, "ERROR!");
         }
}
/*Representa la corriente que pasa a traves del dispositivo, junto con
el consumo por parte del hardware,
 *con 4 digitos y una precision de milesimas
 *x=posicion coordenada X
 *y=pocision coordenada Y */
void corriente(int x, int y)//CYAN
{
    FG_Color = CYAN;
    unsigned int long analisis;
    analisis=ibat;
    if(analisis<=6037)
    {
            c[0]= (analisis/1000)+48;//unidades
            analisis= analisis-((c[0]-48)*1000);
            c[1]= '.';
            c[2]= (analisis/100)+48;//decimas
            analisis= analisis-((c[2]-48)*100);
            c[3]= (analisis/10)+48;//centesimas
            analisis= analisis-((c[3]-48)*10);
            c[4]= (analisis/1)+48;//milesimas
            c[5]= 'A';
            c[6]= 0;
            /*Si ambas cadenas son diferentes, se representa por
pantalla, en caso de ser iguales se evita representar de nuevo*/
            if(strcmp(c,e1)!=0)
            {
               TFT_printBIG(x, y, c);
               strcpy(e1,c);
            }
    }
        else if(analisis>6037)//Se ha producido un error
        {
           TFT_printBIG(x, y, "ERROR!");
        }
}
/*Representa la temperatura en grados centigrados o farenheit
 *Rango de -10°C hasta 100°C y rango de 14°F hasta 212°F
 *centigrados= 3 digitos para valores negativos, y 4 digitos para
valores positivos, con una precision de decimas de grado
 *fahrenheits= 4 digitos con una precision de decimas de fahrenheits
 *x=posicion coordenada X
 *y=pocision coordenada Y*/
void temperatura(int x, int y) //YELLOW
{
    FG_Color = YELLOW;
    float procesamiento;//variable para el calculo de funciones con
valores tipo float
    unsigned int long stemp3;//Variable para evitar entrar en otro
condicional tras operar
```

```
    /*Partimos de una tension inversamente proporcional a la
temperatura, que la obtenemos a partir del numero de escalones que nos
proporciona el CAD
     *Esta tension en mV se aplica en la funcion de transferencia
proporcionada por el fabricante
     *Pasamos el valor obtenido del ADC a mV multiplicado por 1000
para evitar perdida de informacion
     *Tª=(8.194-sqrt((8.194^2)+(4*0.00262*(1324-Vtemp(mV)))))/(2*-
0.00262)+30
     *A la cual se multiplica por 10 para al pasarse a enteros, tener
precision decimal de la temperatura: */
    procesamiento=(((8.194-sqrt((8.194*8.194)+(4*0.00262L*(1324-
(stemp2*0.805L))))))/(2*-0.00262L))+30)*10;
    stemp= (unsigned int long) procesamiento;//pasamos el valor entero
de la funcion de transferencia a nuestra variable de analisis sin
signo
    stemp3=stemp;
    if(((stemp2* 805L)<737000L)|((stemp2* 805L)>1648000L))//Si la
temperatura es menor que -10°C o superior a 100°C
    {
        TFT_printBIG(x, y, " DANGER");
        goto fin3;
    }
    /*_________________GRADOS_CENTIGRADOS_________________*/
    if(zz1==0)
    {   ////////////////////////////
_____________________________________________________________________
______
        if(procesamiento<0)//Temperatura negativa
        {
            f[0]= '-';//Signo negativo
            if(stemp==100)//Si es -10.0°C, tiene 3 digitos
            {
               f[1]= (stemp/100)+48;//decena
               stemp= stemp-((f[1]-48)*100);
            }
                else if(stemp!=100)//Si es un valor negativo con 2
digitos, no tiene decenas
                {
                    f[1]= ' ';//para evitar dibujar un 0 en las
decenas
                }
        }
        else if(procesamiento>0)//Temperatura positiva
        {
           if(stemp==1000)//Valor de maximo de 100°C, por ello para
evitar representar un digito mas en la temperatura, representamos
directamente 100°C
            {
               char f[]={' ','1','0','0',' ',' ','C'};
               f[5]= 128;//Simbolo °
               f[7]= 0;
               TFT_printBIG(x, y, f);
               goto fin3;//salimos de la funcion para evitar realizar
las siguientes operaciones
            }
             else if (stemp<1000)//Valor positivo inferior a 100°C
            {
                f[0]= ' ';//Espacio correspondiente para cuando sea
valor negativo, en caso positivo no hace falta añadirle un signo
positivo, se obvia
                if(stemp>100)//Valor comprendido entre 10.0°C y 99.9°C
```

```
                {
                   f[1]= (stemp/100)+48;//decena
                   stemp= stemp-((f[1]-48)*100);
                }
                    else if(stemp<100)//Valor comprendido entre 0.0°C
y 9.9°C
                    {
                        f[1]= ' ';//para evitar dibujar un 0 en las
decenas
                    }
            }
        }
        f[2]= (stemp/10)+48; //unidad
        stemp= stemp-((f[2]-48)*10);
        f[3]= '.';
        f[4]= (stemp)+48;//decima
        f[5]= 128;//Simbolo °
        f[6]= 'C';
        f[7]= 0;
    }
    /*_________________GRADOS_FAHRENHEIT_________________*/
    if(zz1==1)
    {
        procesamiento=(((procesamiento/10)*1.8)+32)*10;
        stemp= (unsigned int long) procesamiento;//pasamos el valor
entero de la funcion de transferencia a nuestra variable de analisis
sin signo
        if(procesamiento<1000)//Cuando no hay centenas a representar
        {
             f[0]= ' ';
        }
        else if(procesamiento>=1000)//Temperatura positiva
        {
                f[0]= (stemp/1000L)+48;//centena
                stemp= stemp-((f[0]-48)*10000L);
        }
        f[1]= (stemp/100)+48;//decena
        stemp= stemp-((f[1]-48)*100);
        f[2]= (stemp/10)+48;//unidad
        stemp= stemp-((f[2]-48)*10);
        f[3]= '.';
        f[4]= stemp+48;//decima
        f[5]= 128;//Simbolo °
        f[6]= 'F';
        f[7]= 0;
    }
    if((stemp3>=0)&(stemp3<=1000))//Temperatura dentro del rango de
valores
    {
        /*Si ambas cadenas son diferentes, se representa por pantalla,
en caso de ser iguales se evita representar de nuevo*/
        if(strcmp(f,f1)!=0)
        {
            TFT_printBIG(x, y, f);
            strcpy(f1,f); //copiamos la cadena f en f1
        }
    }
    fin3://No hacemos nada
        zz1=zz1;
}
/*Representa la potencia en vatios con 6 digitos y una precision de
milesimas
```

```
 *x=posicion coordenada X
 *y=pocision coordenada Y */
void potencia(int x, int y)//WHITE
{
    FG_Color= WHITE;
    if(watts<=153080L)
    {
        p[0]= (watts/100000L)+48;//centena
        watts= watts-((p[0]-48)*100000L);
        p[1]= (watts/10000L)+48;//decena
        watts= watts-((p[1]-48)*10000L);
        p[2]= (watts/1000L)+48;//unidad
        watts= watts-((p[2]-48)*1000);
        p[3]= '.';
        p[4]= (watts/100)+48;//decima
        watts= watts-((p[4]-48)*100);
        p[5]= (watts/10)+48;//centesima
        watts= watts-((p[5]-48)*10);
        p[6]= (watts/1)+48;//milesima
        p[7]= 'W';
        p[8]= 0;
        /*Si ambas cadenas son diferentes, se representa por pantalla,
en caso de ser iguales se evita representar de nuevo*/
        if(strcmp(p,p1)!=0)
            {
                TFT_print(x, y, p);
                strcpy(p1,p); //copiamos la cadena p en p1
            }
    }
    else if (watts>153080L)//Se ha producido un error
    {
        TFT_print(x, y," ¡ERROR!");
    }
}
/*Representa mili vatios hora y mili amperios hora con 5 digitos
 *x=posicion coordenada X
 *y=pocision coordenada Y
 *z=0 letra pequeña, z=1 letra grande
 *w=0 mili vatios hora, w=1 mili amperios hora*/
void vatiosHamperiosH(int x, int y, int z, int w)
{
    unsigned int long analisis;
    if(w==0)//mili vatios hora
    {
        FG_Color= WHITE;
        analisis=mwattsh;
    }
        else if (w==1)//mili amperios hora
        {
            FG_Color= RED;
            analisis=mamph;
        }
    p[0]= (analisis/10000000L)+48;//decena de mil
    analisis= analisis-((p[0]-48)*10000000L);
    p[1]= (analisis/1000000L)+48;//unidad de mil
    analisis= analisis-((p[1]-48)*1000000L);
    p[2]= (analisis/100000L)+48;//centena
    analisis= analisis-((p[2]-48)*100000L);
    p[3]= (analisis/10000L)+48;//decena
    analisis= analisis-((p[3]-48)*10000L);
    p[4]= (analisis/1000L)+48;//unidad
    if(w==0)//mili vatios hora
```

```
    {
        p[5]= 'm';
        p[6]= 'W';
        p[7]= 'h';
        p[8]= 0;
        /*Si ambas cadenas son diferentes, se representa por pantalla,
en caso de ser iguales se evita representar de nuevo*/
        if(strcmp(p,ph1)!=0)
        {
            if(z==0)TFT_print(x, y, p);//tamaño pequeño
            else if(z==1)TFT_printBIG(x, y, p);//tamaño grande
            strcpy(ph1,p); //copiamos la cadena ph en ph1
        }
    }
        else if (w==1)// mili amperios hora
        {
            p[5]= 'm';
            p[6]= 'A';
            p[7]= 'h';
            p[8]= 0;
            /*Si ambas cadenas son diferentes, se representa por
pantalla, en caso de ser iguales se evita representar de nuevo*/
            if(strcmp(p,ah1)!=0)
            {
                if(z==0)TFT_print(x, y, p);//tamaño pequeño
                else if(z==1)TFT_printBIG(x, y, p);//tamaño grande
                strcpy(ah1,p); //copiamos la cadena ph en ah1
            }
        }
}
/*Representa la resistencia con 6 digitos y una precision de decimas
 *x=posicion coordenada X
 *y=pocision coordenada Y*/
void resistencia(int x, int y)
{
    res=(vin*100)/ibat;

    p[0]= (res/1000000L)+48;//decena de mil
    res= res-((p[0]-48)*1000000L);
    p[1]= (res/100000L)+48;//unidad de mil
    res= res-((p[1]-48)*100000L);
    p[2]= (res/10000L)+48;//centena
    res= res-((p[2]-48)*10000L);
    p[3]= (res/1000)+48;//decena
    res= res-((p[3]-48)*1000);
    p[4]= (res/100)+48;//unidad
    res= res-((p[4]-48)*100);
    p[5]= '.';
    p[6]= (res/10)+48;//decima
    p[7]= 127;//simbolo ohmnio
    p[8]= 0;
    /*Si ambas cadenas son diferentes, se representa por pantalla, en
caso de ser iguales se evita representar de nuevo*/
    if(strcmp(p,r1)!=0) //dibujar unicamente cuando varíe el valor
    {
        FG_Color= BLUE;
        TFT_print(x, y, p);
        strcpy(r1,p); //copiamos la cadena r en r1
    }
}
/*Representa el reloj con una precision de segundos
 *x=posicion coordenada X
```

```
 *y=pocision coordenada Y*/
void reloj(int x , int y)
{
    segs1=segs;
    min1=min;
    hora1=hora;
    /*Horas*/
    p[0]= (hora/10)+48;
    hora1= hora-((p[0]-48)*10);
    p[1]= (hora1/1)+48;
    p[2]= ':';
    /*Mnutos*/
    p[3]= (min/10)+48;
    min1= min-((p[3]-48)*10);
    p[4]= (min1/1)+48;
    p[5]= ':';
    /*Segundos*/
    p[6]= (segs/10)+48;
    segs1= segs-((p[6]-48)*10);
    p[7]= (segs1/1)+48;
    p[8]= 0;
    FG_Color= WHITE;
    TFT_print(x,y,p);
}

/*Funcion para variar un caracter de las cadenas almacenadas de los
valores,
 *de esta manera, aunque no varie el valor a representar, si se llama
a esta funcion previamente
 *se representa de nuevo por pantalla al menos una vez todos los
valores*/
void resetcadenas(void)
{
    c1[5]='Z';
    e1[5]='Z';
    f1[5]='Z';
    p1[5]='Z';
    ph1[5]='Z';
    ah1[5]='Z';
    r1[5]='Z';
    /*Reinicion de variable a 0 para no llamar todo el rato a
resetcadenas()*/
    tres=0;
}
/*_____________________________________________________________________
_____________________________*/
/*_____________________________________SECCION_B_____________________
_____________________________*/
/*_____________________________________________________________________
_____________________________*/
/*Representa el marco base de botones con relacion a su rotacion*/
void marco(void)
{
    if(rotacion==0)
    {
        dimensiones(3,2,110,32,MAGENTA);//cuadrado de fondo OK
        dimensiones(5,1,106,1,MAGENTA);//Raya superior OK
        dimensiones(5,34,106,1,MAGENTA);//Raya inferior OK
        dimensiones(2,4,1,28,MAGENTA);//Raya vertical izq OK
        dimensiones(113,4,1,28,MAGENTA);//Raya vertical der OK
        dimensiones(5,201,106,1,MAGENTA);//Raya superior NEXT
        dimensiones(5,234,106,1,MAGENTA);//Raya inferior NEXT
```

```
        dimensiones(2,204,1,28,MAGENTA);//Raya vertical izq NEXT
        dimensiones(113,204,1,28,MAGENTA);//Raya vertical der NEXT
        FG_Color = WHITE;
        BG_Color = MAGENTA;
        TFT_printBIG(29 ,2, "OK");//arriba izquierda
        TFT_printBIG(3 ,202, "NEXT");//abajo izquierda
        BG_Color = BLACK;
    }
    else if(rotacion==1)
    {
        dimensiones(3,202,110,32,MAGENTA);//cuadrado de fondo OK
        dimensiones(5,201,106,1,MAGENTA);//Raya superior OK
        dimensiones(5,234,106,1,MAGENTA);//Raya inferior OK
        dimensiones(2,204,1,28,MAGENTA);//Raya vertical izq OK
        dimensiones(113,204,1,28,MAGENTA);//Raya vertical der OK
        dimensiones(128,201,106,1,MAGENTA);//Raya superior NEXT
        dimensiones(128,234,106,1,MAGENTA);//Raya inferior NEXT
        dimensiones(125,204,1,28,MAGENTA);//Raya vertical izq NEXT
        dimensiones(236,204,1,28,MAGENTA);//Raya vertical der NEXT
        FG_Color = WHITE;
        BG_Color = MAGENTA;
        TFT_printBIG(29 ,202, "OK");//abajo izquierda
        TFT_printBIG(126 ,202, "NEXT");//abajo derecha
        BG_Color = BLACK;
    }
    else if(rotacion==2)
    {
        dimensiones(126,202,110,32,MAGENTA);//cuadrado de fondo OK
        dimensiones(128,201,106,1,MAGENTA);//Raya superior OK
        dimensiones(128,234,106,1,MAGENTA);//Raya inferior OK
        dimensiones(125,204,1,28,MAGENTA);//Raya vertical izq OK
        dimensiones(236,204,1,28,MAGENTA);//Raya vertical der OK
        dimensiones(128,1,106,1,MAGENTA);//Raya superior NEXT
        dimensiones(128,34,106,1,MAGENTA);//Raya inferior NEXT
        dimensiones(125,4,1,28,MAGENTA);//Raya vertical izq NEXT
        dimensiones(236,4,1,28,MAGENTA);//Raya vertical der NEXT
        FG_Color = WHITE;
        BG_Color = MAGENTA;
        TFT_printBIG(152 ,202, "OK");//abajo derecha
        TFT_printBIG(126 ,2, "NEXT");//arriba derecha
        BG_Color = BLACK;
    }
    else if(rotacion==3)
    {
        dimensiones(126,2,110,32,MAGENTA);//cuadrado de fondo OK
        dimensiones(128,1,106,1,MAGENTA);//Raya superior OK
        dimensiones(128,34,106,1,MAGENTA);//Raya inferior OK
        dimensiones(125,4,1,28,MAGENTA);//Raya vertical izq OK
        dimensiones(236,4,1,28,MAGENTA);//Raya vertical der OK
        dimensiones(5,1,106,1,MAGENTA);//Raya superior NEXT
        dimensiones(5,34,106,1,MAGENTA);//Raya inferior NEXT
        dimensiones(2,4,1,28,MAGENTA);//Raya vertical izq NEXT
        dimensiones(113,4,1,28,MAGENTA);//Raya vertical der NEXT
        FG_Color = WHITE;
        BG_Color = MAGENTA;
        TFT_printBIG(152 ,2, "OK");//arriba derecha
        TFT_printBIG(3 ,2, "NEXT");//arriba izquierda
        BG_Color = BLACK;
    }
}
/*Regula el brillo de la pantalla variando el ciclo de trabajo de la
señal PWM*/
```

```
void brillo(void)
{
    switch (brightness) {
       case 0:
            OC1R=2;
            break;
       case 1:
            OC1R=124;
            break;
       case 2:
            OC1R=248;
            break;
       case 3:
            OC1R=372;
            break;
       case 4:
            OC1R=496;
            break;
       case 5:
            OC1R=624;
            break;
    }
}
/*Representa el mastil del eje de ordenadas*/
void eje(void)
{
    int i;
    dimensiones(82,46,2,144,WHITE);//mastil vertical
    for(i=0;i<10;i++)//escalones
    {
        dimensiones(81,46+(i*16),6,2,WHITE);
    }
}
/*Representa una malla de puntos en la grafica para un mejor analisis
de sus valores representados*/
void cuadrante(void)
{
    int i,j;
    for(i=0;i<10;i++)//filas
    {
        for(j=1;j<11;j++)//columnas
        {
            dimensiones(86+(j*15),46+(i*16),2,2,WHITE);
        }
    }
}
/*Representa el eje de ordenadas de las graficas con sus valores
correspondientes
 *valor= Valor maximo del eje de ordenadas
 *y= Diferencia entre valores
 *z=0 Voltaje, z=1 Corriente*/
void ejeValores(unsigned int valor, unsigned int y, int z)
{
    FG_Color = WHITE;
    BG_Color = BLACK;
    int i;
    unsigned int x;
    for(i=0;i<10;i++)
    {
        if(z==0)//valores de voltaje
        {
            v[0]= (valor/10000)+48;//decena
```

```
            x= valor-((v[0]-48)*10000);
            v[1]= (x/1000)+48; //unidad
            x= x-((v[1]-48)*1000);
            v[2]= '.';
            v[3]= (x/100)+48;//decima
            x= x-((v[3]-48)*100);
            v[4]= (x/10)+48; //centesima
            v[5]= 'V';
        }
            else if(z==1)//valores de corriente
            {
                v[0]= (valor/1000)+48;//unidad
                x= valor-((v[0]-48)*1000);
                v[1]= '.';
                v[2]= (x/100)+48;//decima
                x= x-((v[2]-48)*100);
                v[3]= (x/10)+48;//centesima
                x= x-((v[3]-48)*10);
                v[4]= (x/1)+48; //milesima
                v[5]= 'A';
            }
        v[6]= 0;
        TFT_print(0 ,38+(i*16), v);
        valor=valor-y; //Se resta la diferencia entre valores
    }
}
int x4=0,lm1=0,graf=0,lm=1;//x4 posicion del punto en la grafica/lm1
direccion de valores muestreados finales o almacenados/graf contador
de valores muestreados/lm cantidad de valores a mostrar en un unico
punto
unsigned int gvolt[3600],gcor[3600];//valores obtenidos en 1 hora
unsigned int
maximoV=0,minimoV=25357,maximoI=0,minimoI=6037,saltoV,saltoI;//maximo
y minimo de la grafica de V y I, y los saltos para su eje y
representacion
unsigned int maximoV1=0,minimoV1=25357,maximoI1=0,minimoI1=6037;//max
y min comparativos para evitar redibujar la grafica todo el rato
int xmax=236;//valor maximo de X para la grafica
int posicionHora=0, elegirTiempo=0, intervalo=1;
int posicion10min=0,posicion20min=0,posicion30min=0, posicion1hora=0;
int activadoCambio=0,activadoCambio2=0,lecturaMemoria=0;
unsigned int long direccionV[18],direccionI[18];//almacenamiento de
hasta 18 horas
unsigned int long direccion=0;//direccion de programa
unsigned int pepepep=0;

/*Almacena los valores de Tension y Corriente de la ultima hora
muestreada
 *se almacena en dos paginas los valores de tension, y a continuacion
en otras dos paginas
 *se almacenan los valores de corriente*/
void GuardarFlashHora(void)
{
    int i=0,z=0,z2=0,z3=0,z4=0,x=0;
    unsigned int mayor=0,menor=25357,analizar=0;
    /*Cuando se almacena la primera hora*/
    if(posicionHora==0)
    {
        /*se borra la pagina a continuacion del codigo principal de
programa y obtenemos su direccion*/
        direccionV[posicionHora]=StartWritteFlash(direccion,0);
        direccion=direccionV[posicionHora];
```

```
        /*Calculo de todas las direcciones para almacenamiento de 18
horas*/
        for (i=0 ; i<17; i++)
        {
            /*Hay una diferencia de direccion de 2048 en decimal entre
paginas, puesto que se escriben dos paginas para Tension y otras 2
para Corriente
             *la diferencia entre la direccionV y la direccionI será
de 4096, y la diferencia entre las direcciones de V o I será de 8192*/
            direccionI[i]=(direccionV[posicionHora]+4096L)+(i*8192L);
            direccionV[i+1]=direccionV[posicionHora]+((i+1)*8192L);
        }
        direccionI[i]=(direccionV[posicionHora]+4096L)+(i*8192L);
    }
        /*Una vez almacenada ya una hora*/
        else if(posicionHora!=0)
        {
            /*Se envia la direccion de pagina anterior a la
direccionV, puesto que esta aun no se ha escrito, para que se borre la
direccionV*/
            direccion=StartWritteFlash((direccionV[posicionHora]-
2048L),1);
            direccion=direccionV[posicionHora];
        }

    for(z3=0;z3<4;z3++)
    {
        z2=0;//reinicio variable de direccion
        /*Se almacenan 1024 valores por pagina, escritos de dos en
dos*/
        if((z3==0)|(z3==2))z4=512;
            /*Se almacenan 776 valores en la siguiente pagina,
escritos de dos en dos*/
            else if ((z3==1)|(z3==3))z4=388;
        /*Bucle de escritura por pagina de memoria*/
        for(z=0;z<z4;z++)
        {
            /*Se obtiene el maximo y minimo de cada 4 segundos*/
            for (i=0 ; i<4; i++)
            {
                if(z3<=1)
analizar=gvolt[x];/*_____________________TENSION_____________________*
/
                    else if
(z3>=2)analizar=gcor[x];/*_____________CORRIENTE_____________________*/
                if (analizar> mayor) mayor=analizar;
                if (analizar< menor)menor=analizar;
                x++;
            }
            /*Escritura de valor maximo y minimo en memoria*/
            WritteFlash((direccion+(z2*4)),mayor,menor);
            z2++;
            /*Reinicio de variables para el analisis de escritura*/
            mayor=0;
            menor=25357;
        }
        /*Actualización para obtener la siguiente direccion de memoria
para su borrado*/
        if((z3==0)|(z3==2))
        {
            direccion=StartWritteFlash(direccion,1);
            if(z3==0)direccion=(direccionV[posicionHora]+2048L);
```

```
            else if(z3==2)direccion=(direccionI[posicionHora]+2048L);
        }
            /*direccion de memoria donde se almacenan los valores de
corriente*/
            else if (z3==1)
            {
                /*borrado de pagina*/
                direccion=StartWritteFlash(direccion,1);
                direccion=direccionI[posicionHora];
                x=0;
            }
    }
    /*Incremento variable de direccion del numero de horas
acumuladas*/
    posicionHora++;
}
unsigned int MaximoEje=0,MinimoEje=25357;
/*Calcula los limites superiores e inferiores del eje de ordenadas, y
en caso de diferir
 *de los que se tienen previamente, representa el nuevo escalado y
devuelve un 1
 *mayor= valor mayor a analizar
 *menor= valor menor a analizar
 *x=0 Tension, x=1 Corriente*/
int HallarMAXyMIN ( unsigned int mayor,unsigned int menor, int x)
{
    int j,j1,j2,result=0;
    unsigned int maximo=0,minimo=0,maximo1=0,minimo1=0,taSalto;
    /*_____________________TENSION_____________________*/
    /*________VALOR MAXIMO DE LA GRAFICA 25.380V_______*/
    if(x==0)
    {
        j1=95;//Rango del bucle para el analisis de maximos y minimos,
(j1-1)*taSalto= (95-1)*270=25380 valor maximo a analizar
        j2=283;//Rango del bucle para el analisis del tamaño de salto
entre valores del eje de ordenadas, (j2-1)*taSalto= (283-1)*90=25380
valor maximo a analizar
        taSalto=270;//Tamaño  minimo para analizar el limite superior
e inferior, multiplo de 90
        /*Actualizacion de variables para su correspondiente
analisis*/
        maximo=maximoV;
        minimo=minimoV;
        maximo1=maximoV1;
        minimo1=minimoV1;
    }
        /*_____________________CORRIENTE_____________________*/
        /*________VALOR MAXIMO DE LA GRAFICA 6.030A________*/
        else if(x==1)//corriente
        {
            j1=68;//(j1-1)*taSalto= (68-1)*90=6030
            j2=68;//(j2-1)*taSalto= (68-1)*90=6030
            taSalto=90;
            maximo=maximoI;
            minimo=minimoI;
            maximo1=maximoI1;
            minimo1=minimoI1;
        }
    /*Se calcula el limite superior de la grafica*/
    if(maximo<mayor)
    {
        for (j=1 ; j<j1; j++)
```

```
        {
            if(mayor<=(j*taSalto))break;//Sale cuando cae dentro del
valor
        }
        maximo=j*taSalto;//limite superior
        /*Se almacena el maximo en su correspondiente variable*/
        if(x==0)maximoV=maximo;
            else if(x==1)maximoI=maximo;
    }
    /*Se calcula el limite inferior de la grafica*/
    if(minimo>menor)
    {
        for (j=1 ; j<j1; j++)
        {
            if(menor<=(j*taSalto))break;//Sale cuando cae dentro del
valor
        }
        minimo=(j-1)*taSalto;//limite inferior
        /*Se almacena el minimo en su correspondiente variable*/
        if(x==0)minimoV=minimo;
            else if(x==1)minimoI=minimo;
    }
    /*Se cambia la escala si varian el maximo o el minimo*/
    if((maximo1!=maximo) | (minimo1!=minimo))
    {
        /*bucle para calcular la diferencia entre cada valor
representado del eje de ordenadas*/
        for (j=1 ; j<j2; j++)
        {
            if((maximo - minimo)<=(j*90))break;
        }
        /*La diferencia entre valores de la grafica, es el tamaño de
un cuadrante, puesto que se divide entre 9 cuadrantes (j*90)/9=j*10*/
        taSalto= j*10;//Se obtiene la diferencia entre valores
        if(x==0)saltoV= taSalto;
            else if(x==1)saltoI= taSalto;
        /*Se representa el eje de ordenadas*/
        ejeValores(maximo,taSalto,x);
        /*Se almacenan en sus correspondiente variables*/
        if(x==0)
        {
            maximoV1=maximo;
            minimoV1=minimo;
        }
            else if(x==1)
            {
                maximoI1=maximo;
                minimoI1=minimo;
            }
        result= 1;//Significa que se ha cambiado la escala de la
grafica
    }
    return result;
}
/*Funcion para recorrer valores de un intervalo y se obtienen los
maximos y minimos.
 *La finalidad es para que a la hora de representar un intervalo de
tiempo, se establezca un escalado del eje de ordenadas final y evitar
redibujados innnecesarios.
 *De esta manera al llamar a la funcion valorgrafica() para
representar cada punto, y dentro de esta se llame a HallarMaxyMin(),
```

```
 *esta no nos devuelva un 1. Puesto que cuando HallarMaxyMin()
devuelve un 1, supone un cambio de maximo y minimo del eje de
ordenadas,
 *y por lo tanto volver a representar toda la grafica hasta ese
momento sucendiendo todas las veces que en ese intervalo variase el
maximo y minimo */
void RecorrerValores(int x,int y,int z)
{
    int i=0,cont=0;
    uint32_t direccion;
    unsigned int analizar=0;
    if(z==0)//valores del vector
    {
        for (i=0 ; i<y; i++)
        {
            if(x==0)
analizar=gvolt[lm1];/*_____________________TENSION_____________________
_*/
                else if
(x==1)analizar=gcor[lm1];/*_____________CORRIENTE_____________________*/
            /*Solo se realiza si hay una variable almacenada*/
            if(analizar != 0)
            {
                if (analizar> MaximoEje)MaximoEje=analizar;
                if (analizar< MinimoEje)MinimoEje=analizar;
                lm1++;
                cont++;
            }
            /*En caso de no haber variable a analizar, se sale de la
funcion*/
            else if (analizar == 0)goto out;
        }
    }
    else if(z==1)//valores de la memoria de programa
    {
        /*Se obtiene la direccion de las variables a analizar*/

if(x==0)direccion=direccionV[elegirTiempo];/*_____________________TENS
ION_____________________*/
            else if (x==1)
direccion=direccionI[elegirTiempo];/*__________CORRIENTE_____________
_____*/
        /*Se saca el valor mayor y menor*/
        for (i=0 ; i<y; i++)
        {
            /*Puesto que se han almacenado el maximo y minimo cada vez
que se escribia en la memoria de programa
             *leyendo desde la posicion inicial de la direccion de
programa obtendremos el valor maximo,
             *y en la siguiente direccion el valor minimo, y asi
sucesivamente */
            if (((ReadFlash(direccion+(2*lm1)))&0x0000FFFF)>
MaximoEje)
            {
                MaximoEje=(ReadFlash(direccion+(2*lm1)))&0x0000FFFF;
            }
            if (((ReadFlash(direccion+(2+(2*lm1))))&0x0000FFFF)<
MinimoEje)
            {

MinimoEje=((ReadFlash(direccion+(2+(2*lm1))))&0x0000FFFF);
            }
```

```
            lm1=lm1+2;
        }
    }
    out:
      i=0;
}
char es[8];
/*Almacena en la cadena "es" el rango de tiempo a representar
 *i= inicio del rango
 *x= intervalo de tiempo
 *y =0 minutos, y=1 horas*/
void intervaloChar(int i, int x, int y)
{
    int j;
    /*Inicio de intervalo*/
    es[0]= (i/10)+48;
    j= i-((es[0]-48)*10);
    es[1]= (j)+48;
    es[2]= '-';
    /*Fin de intervalo*/
    if(y==0)
    {
        es[3]= ((i/10)+x)+48;
        j= i-(((es[3]-x)-48)*10);
    }
        else if(y==1)
        {
            i=i+x;
            es[3]= (i/10)+48;
            j= i-((es[3]-48)*10);
        }
    es[4]= (j)+48;
    if(y==0)//minutos
    {
        es[5]= 'M';
        es[6]= 'i';
        es[7]= 'n';
    }
        else if(y==1)//horas
        {
            es[5]= ' ';
            es[6]= 'H';
            es[7]= ' ';
        }
    es[8]=0;//fin de cadena
}
/*Representa el intervalo de tiempo en el que se encuentra la grafica
dependiendo de su rotacion*/
void posicionIntervalo(void)
{
    int i,posicion=0;
    /*Tiempo real*/
    if(intervalo==0)
    {
        if(rotacion==0)TFT_print(131, 210, "IN TIME ");//abajo derecha
            else if(rotacion==1)TFT_print(131, 10, "IN TIME
");//arriba derecha
            else if(rotacion==2)TFT_print(8, 10, "IN TIME ");//arriba
izquierda
            else if(rotacion==3)TFT_print(8, 210, "IN TIME ");//abajo
izquierda
    }
```

```
        /*Intervalos de 10, 20 y 30 minutos*/
        else if((intervalo==1) | (intervalo==2) | (intervalo==3))
        {
            /*En la ultima hora de muestreo*/
            if(activadoCambio==0)
            {
                /*valor inicial donde empieza el rango de tiempo*/
                i=((lm1/(intervalo*600))*10)*intervalo;
            }
                /*Analizando horas almacenadas y hora final*/
                else if(activadoCambio!=0)
                {
                    if(intervalo==1)posicion=posicion10min;
                        else if(intervalo==2)posicion=posicion20min;
                        else if(intervalo==3)posicion=posicion30min;
                    /*valor inicial donde empieza el rango de tiempo*/
                    i=(posicion*10)*intervalo;
                }
                /*Se obtiene la cadena para representar el rango del
intervalo en __minutos__*/
                intervaloChar(i,intervalo,0);
                /*Representa el rango de tiempo en __minutos__*/
                if(rotacion==0)TFT_print(131, 210, es);//abajo derecha
                    else if(rotacion==1)TFT_print(131, 10,
es);//arriba derecha
                    else if(rotacion==2)TFT_print(8, 10, es);//arriba
izquierda
                    else if(rotacion==3)TFT_print(8, 210, es);//abajo
izquierda
                /*Se obtiene la cadena para representar el rango del
intervalo en __horas__*/
                if(activadoCambio==0)intervaloChar(posicionHora,1,1);
                    else
if(activadoCambio!=0)intervaloChar(posicion1hora,1,1);
                /*Representa el rango de tiempo en __horas__*/
                if(rotacion==0)TFT_print(131, 10, es);//arriba derecha
                    else if(rotacion==1)TFT_print(8, 10, es);//arriba
izquierda
                    else if(rotacion==2)TFT_print(8, 210, es);//abajo
izquierda
                    else if(rotacion==3)TFT_print(131, 210,
es);//abajo derecha
        }
        /*Intervalo de 1 hora*/
        else if(intervalo==4)
        {
            /*En la ultima hora de muestreo*/
            if(activadoCambio==0)intervaloChar(posicionHora,1,1);
                /*Analizando horas almacenadas y hora final*/
                else
if(activadoCambio!=0)intervaloChar(posicion1hora,1,1);
            /*Representacion intervalo*/
            if(rotacion==0)TFT_print(131, 10, es);//arriba derecha
                    else if(rotacion==1)TFT_print(8, 10, es);//arriba
izquierda
                    else if(rotacion==2)TFT_print(8, 210, es);//abajo
izquierda
                    else if(rotacion==3)TFT_print(131, 210,
es);//abajo derecha
        }
}
int bucle=0;//cantidad de puntos que se representan
```

```
/*Funcion de actualizacion de parametros para presentar o representar
las graficas
 *Su finalidad es analizar todos los valores del fragmento de tiempo
que se va a analizar, ya sea un rango no completo de valores o uno
completo,
 *actualizando su escala y el valor de las variables lm1, lm, x4 y
bucle */
void grafica(void)//explicar
codigo_____________________________________________________________
_____________________________________________
{
    int y,i,i1=0,x,suma,valor1;
    lm1=0;//direccion de valores muestreados finales o almacenados
    x4=0;//posicion de cada punto de la grafica en el eje X, variando
desde 0 a 149
    dimensiones(86,46,152,146,BLACK);//borramos pantalla
    eje();//Representa el mastil del eje de ordenadas
    cuadrante();//Representa una malla de puntos en la grafica
    /*Cuando pulsamos OK en una grafica, visualizamos las horas
analizadas hasta el momento para elegir la que queremos analizar
     *independientemente del intervalo en el que se encuentre*/
    if((activadoCambio==1) & (activadoCambio2==0)) goto mostrarHora;
    /*_________________Tiempo real_________________*/
    if(intervalo == 0)
    {
        suma=150;//Cantidad de valores que se muestran en los 150
puntos de la grafica
        i1=25;//Valor del bucle para calcular la posicion de los
valores a mostrar por pantalla (i1-1)*suma=3600
        lm=1;//Bloque de valores a dibujar por cada punto, en este
caso 1 punto de la grafica representa 1 valor muestreado
    }
        /*_____________________________10 , 20 y 30
minutos_____________________________*/
        else if((intervalo==1) | (intervalo==2) | (intervalo==3))
        {
            suma=600*intervalo;//Cantidad de valores que se muestran
en los 150 puntos de la grafica
            i1=6/intervalo;//Valor del bucle para calcular la posicion
de los valores a mostrar por pantalla
            lm=4*intervalo;//Bloque de valores a dibujar por cada
punto
            /*Cuando se cambia la funcion del boton NEXT*/
            if(activadoCambio!=0)
            {
                /*_____Ultima hora que se esta muestreando_____*/
                if(lecturaMemoria==0)
                {
                    /*Posicion inicial donde están los valores a
representar por pantalla*/
                    if(intervalo==1)lm1=posicion10min*suma;
                        else if(intervalo==2)lm1=posicion20min*suma;
                        else if(intervalo==3)lm1=posicion30min*suma;
                    /*Cantidad de valores que hay muestreados en ese
rango de tiempo hasta el momento*/
                    if((lm1==0) & (graf<suma))y=graf;//Cuando los
valores a analizar no completan el rango de tiempo que se muestra
                    else if((lm1==0) & (graf>=suma))y=suma;//Cuando
ese rango esta completo, se analiza todo el bloque de valores (suma)
                        else if((lm1!=0) & ((graf-lm1)
>=suma))y=suma;//Cuando ese rango esta completo, se analiza todo el
bloque de valores (suma)
```

```
                        else if((lm1!=0) & ((graf-lm1)<suma))y=graf-
lm1;//Cuando los valores a analizar no completan el rango de tiempo
que se muestra
                }
                /*_____________Horas ya almacenadas_____________*/
                /*Cada hora guardada en memoria tiene un 1 punto por
cada 4 segundos*/
                else if(lecturaMemoria==1)
                {
                    elegirTiempo=posicion1hora;//Se carga la hora que
se analiza en elegirTiempo
                    lm=intervalo;//Bloque de valores a dibujar por
cada punto
                    /*Fija la posicion inicial donde están los valores
a representar por pantalla*/
                    if(intervalo==1)lm1=posicion10min*(suma/2);
                        else
if(intervalo==2)lm1=posicion20min*(suma/2);
                        else
if(intervalo==3)lm1=posicion30min*(suma/2);
                    y=150*intervalo;//Cantidad de valores que hay
almacenados en el intervalo
                }
                goto fin;//se salta el calculo de "y" y "lm1" cuando
activadoCambio=0 y activadoCambio2=0
            }
        }
    /*_____________________________Tiempo real y 10/20/30
minutos_____________________________*/
    if((intervalo==0) | (intervalo==1) | (intervalo==2) |
(intervalo==3))
    {
        valor1=suma;
        /*Bucle para calcular la posicion inicial a partir de la cual
se encuentran los valores a analizar*/
        for(i=0;i<i1;i++)
        {
            if(graf<suma)break;
            suma=suma+valor1;
        }
        lm1=suma-valor1;//Posicion inicial de los valores
        y=(graf-lm1);//Cantidad de valores que hay en el intervalo
        fin:/*_____________________________*/
        if(uno==1)x=0;//tension
            else if(dos==1)x=1;//corriente
        RecorrerValores(x,y,lecturaMemoria);//Obtencion de maximo y
minimo de los valores a analizar
        i=HallarMAXyMIN ( MaximoEje, MinimoEje, x);//Representa el eje
de ordenadas
        /*Cuando se visualiza el ultimo intervalo de tiempo que se
esta muestreando*/
        if(activadoCambio==0)
        {
            lm1=suma-valor1;//Posicion inicial de los valores
            y=(graf-lm1)/lm;//Numero de puntos a representar
        }
            /*_______Analisis de la ultima hora que se esta
muestreando______*/
            else if((activadoCambio!=0) & (intervalo!=0) &
(lecturaMemoria==0))
            {
```

```
                /*Posicion inicial donde están los valores a
representar por pantalla*/
                if(intervalo==1)lm1=posicion10min*suma;
                    else if(intervalo==2)lm1=posicion20min*suma;
                    else if(intervalo==3)lm1=posicion30min*suma;
                /*Numero de puntos a representar en ese rango de
tiempo hasta el momento*/
                if((lm1==0) & (graf<suma))y=(suma-graf)/lm;
                    else if((lm1==0) & (graf>=suma))y=suma/lm;
                    else if((lm1!=0) & ((graf-lm1) >=suma))y=suma/lm;
                    else if((lm1!=0) & ((graf-lm1)<suma))y=(graf-
lm1)/lm;
            }
            /*_______________Analisis de horas ya
almacenadas_________________*/
            else if((activadoCambio!=0) & (intervalo!=0) &
(lecturaMemoria==1))
            {
                y=150;//Numero de puntos a representar
                /*Posicion inicial donde están los valores a
representar por pantalla*/
                if(intervalo==1)lm1=posicion10min*(suma/2);
                    else if(intervalo==2)lm1=posicion20min*(suma/2);
                    else if(intervalo==3)lm1=posicion30min*(suma/2);
            }
    }
        /*_________________1 Hora_____________________________*/
        else if (intervalo==4)
        {
            mostrarHora:/*_____________________________*/
            if(uno==1)x=0;//tension
                else if(dos==1)x=1;//corriente
            if(lecturaMemoria==0)y=graf;//Cantidad de valores que hay
en el intervalo
                else if(lecturaMemoria==1)
                {
                    y=900;//Cantidad de valores que hay en el
intervalo
                    elegirTiempo=posicion1hora;//Se fija la hora que
se analiza
                }
            RecorrerValores(x,y,lecturaMemoria);//Obtencion de maximo
y minimo de los valores a analizar
            i=HallarMAXyMIN ( MaximoEje, MinimoEje, x);//Representa el
eje de ordenadas
            lm1=0;//Posicion inicial de los valores
            if(lecturaMemoria==0)
            {
                lm=24;//24 segundos por punto
                y=graf/lm;//Numero de puntos a representar
            }
            if(lecturaMemoria==1)
            {
                lm=6;//6 valores almacenados por punto, puesto que un
punto almacenado son 4 segundos -> 6*4=24 segundos por punto
                y=150;//Numero de puntos a representar
            }
        }
    /*Representa una cadena de caracteres con la posicion del
intervalo de tiempo en el que se encuentra la grafica*/
    FG_Color = YELLOW;
    posicionIntervalo();
```

```
    //actualizacion de valores para el recorrido de variables
    MaximoEje=0;
    MinimoEje=25357;
    /*Numero puntos que han de representarse*/
    bucle=y;
}
/*Funcion de representado de cada punto de la gráfica
 *mayor=valor maximo del punto
 *menor=valor minimo del punto
 *x=0 Tension, x=1 Corriente*/
void posicionPunto(unsigned int mayor,unsigned int menor, int x)
{
    /*Analisis de posicion de el punto a representar en el eje Y,
siendo
     *el rango del eje Y de 144,donde 0 es el valor maximo y 144 el
valor minimo*/
    int z,j,z4=0,y4=0;
    unsigned int analizar=0,minimo,salto;
    for(z=0;z<2;z++)
    {
        j=8;//Para que en caso de que maximo-resto sea 0, no nos sume
nada a y4
        if(z==0)analizar= mayor;
            else if(z==1)analizar= menor;
        /*_____________________TENSION_____________________*/
        if(x==0)
        {
            minimo=minimoV;
            salto=saltoV;
        }
            /*_____________________CORRIENTE_____________________*/
            else if(x==1)
            {
                minimo=minimoI;
                salto=saltoI;
            }
        if((analizar-minimo)==0)z4=144;
            else if((analizar-minimo)!=0)
            {
                /*analiza en cual de los 9 cuadrantes de salto se
situa el punto*/
                for (j=1 ; j<10; j++)
                {
                    if((analizar-minimo)<=(j*salto))break;
                }
                if(j==10)j=9;//En caso de no entrar dentro del
cuadrante, se fija j=9 para no obtener valores negativos y dibujar
fuera de la grafica
                z4=((9-j)*16);//una vez tenemos el cuadrante, sabemos
que esta situado a partir de (j-1)*16
                analizar= (analizar-minimo) - ((j-1)*salto);// nos
queda el resto dentro del cuadrante
                for (j=1 ; j<9; j++) //analizamos en los 16 puntos
dentro de cada cuadrante, en saltos de 2, dividiendo cada sato en 8
puntos
                {
                    if((analizar)<=(j*((salto)/8)))break;//comprobamos
el resto dentro del cuadrante
                }
                if(j==9)j=8;//En caso de no entrar dentro del rango,
se fija j=8 para no obtener valores negativos y dibujar fuera de la
grafica
```

```
            }
        /*se calcula la posicion del punto con valor maximo*/
        if(z==0)y4=z4 + (16-(j*2));
        /*se calcula la altura del punto hasta el minimo*/
            else if(z==1) z4=(z4 + (16-(j*2)))- y4;
    }
    /*Se representa el punto, dependiendo de si es de V o I, sumandole
un offset de 46 al eje Y y otro de 86 al eje X*/
    if(x==0)dimensiones(x4+86,y4+46,2,z4+2,GREEN);
        else if(x==1)dimensiones(x4+86,y4+46,2,z4+2,RED);
}
/*Funcion para dibujar cada punto de la grafica
 *x=0 Tension, x=1 Corriente*/
void valorgrafica(int x)
{
    unsigned int i=0,cont=0,result=0;
    unsigned int mayor=0,menor=25357;
    unsigned int analizar=0;
    unsigned int long direccion;
    /*Siempre que la gráfica no sea completada*/
    if((x4+86)<xmax)
    {
        /*____________________ULTIMA HORA____________________*/
        if(lecturaMemoria==0)
        {
            /*Se saca el mayor y menor de un bloque de lm valores*/
            for (i=0 ; i<lm; i++)
            {
                if(x==0)
analizar=gvolt[lm1];/*_____________________TENSION_____________________
_*/
                    else if
(x==1)analizar=gcor[lm1];/*_____________CORRIENTE_____________________*/
                /*Si hay variable almacenada se saca el valor mayor y
menor*/
                if(analizar != 0)
                {
                    if (analizar> mayor)mayor=analizar;
                    if (analizar< menor)menor=analizar;
                    lm1++;
                    cont++;
                }
                /*En caso de no haber variable del bloque a analizar,
se sale de la funcion valorgrafica() y se vuelve
                 *al punto anterior hasta que se pueda completar el
bloque de lm variables a mostrar por punto*/
                else if (analizar == 0)
                {
                    lm1=lm1-cont;
                    goto fin; //no dibujamos el punto
                }
            }
        }
        /*____________________MEMORIA____________________*/
        else if(lecturaMemoria==1)
        {

if(x==0)direccion=direccionV[elegirTiempo];/*_____________________TENS
ION_____________________*/
            else if (x==1)
direccion=direccionI[elegirTiempo];/*_____________CORRIENTE__________
_________*/
```

```
            /*Se lee el bloque de lm variables almacenadas para sacar
el valor mayor y menor*/
            /*lm=1 -> 10min___lm=2 -> 20min __lm=4 -> 40 min__lm=6 ->
60 min*/
            for (i=0 ; i<lm; i++)
            {
                if (((ReadFlash(direccion+(2*lm1)))&0x0000FFFF)>
mayor)
                {
                    mayor=(ReadFlash(direccion+(2*lm1)))&0x0000FFFF;
                }
                if (((ReadFlash(direccion+(2+(2*lm1))))&0x0000FFFF)<
menor)
                {

menor=((ReadFlash(direccion+(2+(2*lm1))))&0x0000FFFF);
                }
                lm1=lm1+2;
            }
        }
        /*Calculo y representado del maximo y minimo del eje de
ordenadas de la gráfica*/
        result=HallarMAXyMIN(mayor,menor,x);
        /*En caso de que varíen, puesto que se actualiza, se vuelve a
redibujar toda la grafica respecto a su nuevo escalado*/
        if(result==1)goto fin2;
        /*Enviando el mayor y menor valor obtenido de las variables
analizadas, se representa un punto en la grafica*/
        posicionPunto(mayor,menor,x);
        x4++;//se incrementa la posicion para la representacion del
siguiente punto
        fin:
          x4=x4;//no se hace nada
    }
    /*Cuando llegamos al final de la grafica*/
    if ((x4+86)>=xmax)
    {
        /*Si estamos analizando cualquier intervalo, permanecemos en
el, evitamos el cambio de grafica*/
        if(activadoCambio!=0)goto fin3;
        fin2:
        if(x==0)uno=1;//tension
            else if(x==1)dos=1;//corriente
        /*Se llama a grafica() para obtener los parametros necesarios
para redibujar o empezar un nuevo intervalo de tiempo*/
        grafica();
        fin3:
           x4=x4;//no se hace nada
    }
}
/*Llama a la funcion valorgrafica(x) un maximo de 150 veces para
representar los puntos de la gráfica*/
void redibujado(void)
{
    int x=0;
    if(uno==1)x=0;//tension
    else if(dos==1)x=1;//corriente
    /*Repeticion tantas veces como puntos a representar*/
    while(bucle--)
    {
        valorgrafica(x);
    }
```

```
    /*Actualiza variables para no representar constantemente*/
    uno=0;
    dos=0;
}
/*Actualizacion de pantalla
 *x=posicion de pantalla*/
void pantalla(int x)
{
    char b1[1];
    switch (x) {
/*Tension de carga, corriente, temperatura, potencia, vatios hora,
amperios hora y resistencia*/
       case 0:
           if(tres==1)
           {
               resetcadenas();
               FG_Color = GREEN;
               TFT_putchBIG(190,46,134);//Simbolo tension
               FG_Color = CYAN;
               TFT_putchBIG(190,82,133);//Simbolo corriente
               FG_Color = YELLOW;
               TFT_putchBIG(190,118,131);//Simbolo termometro
           }
           /*Centrado con x=42 y x=18*/
           valortension(26,46);
           corriente(26,82);
           temperatura(0,118);
           potencia(124,177);
           vatiosHamperiosH(9,177,0,0);//mili vatios hora
           vatiosHamperiosH(9,160,0,1);//mili amperios hora
           resistencia(124,160);
            break;
/*Tension de carga, corriente, tension de D+ y tension de D-*/
       case 1:
           if (tres==1)
           {
               resetcadenas();
               FG_Color = GREEN;
               TFT_putchBIG(190,46,134);//Simbolo tension
               FG_Color = CYAN;
               TFT_putchBIG(190,82,133);//Simbolo corriente
               FG_Color = YELLOW;
               TFT_putchBIG(190,118,131);//Simbolo termometro
               FG_Color =GREEN;
               TFT_putchBIG(32,160,135);//Simbolo USB
               dimensiones(40,170,16,4,RED);//BORRADO PANTALLA
COMPLETA
               dimensiones(40,178,16,4,BLUE);//BORRADO PANTALLA
COMPLETA

           }
           valortension(26,46);
           corriente(26,82);
           temperatura(0,118);
           FG_Color = RED;
           TFT_print(76 ,160,"D+:");
           tensiondatos(116,160,1);// D+
           FG_Color = BLUE;
           TFT_print(76 ,177,"D-:");
           tensiondatos(116,177,0);// D-
            break;
/*Vatios hora, Amperios hora y reloj*/
```

```
        case 2:
            if (tres==1)
            {
                resetcadenas();
                FG_Color = BROWN;
                TFT_putch(40,130,129);//Simbolo reloj
            }
            vatiosHamperiosH(15,50,1,0);//mili vatios hora
            vatiosHamperiosH(15,88,1,1);//mili amperios hora
            reloj(68,130);
            corriente(42,157);
             break;
/*Grafica de V*/
        case 3:
            /*Una vez entramos en la pantalla de grafica V, se dibujan
todos los puntos hasta donde nos llegamos*/
            if (uno==1)
            {
                 /*Es necesario para actualizar el maximo y minimo de
cada pantalla, puesto que si no realizamos esto
                  cuando queramos visualizar con mas afinidad acorde a
la grafica visualizada, veriamos con los maximos y
                  minimos mayores que se hayan obtenido*/
                maximoV=0;
                minimoV=25357;
                /*Aunque se actualice maximoV y minimoV, puesto que
son escalones, si no varia lo suficiente para cambiarlos, no se
redibuja
                 por lo que necesitamos variar maximoV1 y minimoV1
para redibujar*/
                maximoV1=0;
                minimoV1=25357;
                /*Calculo del numero de puntos a dibujar y rangos
segun sus intervalos*/
                grafica();
                /*llamada a la funcion valorgrafica() para dibujar los
puntos, el numero de veces obtenido en grafica()*/
                redibujado();
            }
            /*Una vez redibujado todo hasta el momento en el que nos
llegamos*/
            /*Llamamos para representar el siguiente punto de la
grafica si es posible*/
            valorgrafica(0);
            /*Redibuja si el escalado de la grafica cambia*/
            if (uno==1)redibujado();
             break;
/*Grafica de I*/
        case 4:
            /*mismas explicaciones que la grafica de V*/
            if (dos==1)
            {
                 maximoI=0;
                 minimoI=6037;
                 maximoI1=0;
                 minimoI1=6037;
                 grafica();
                 redibujado();
            }
            valorgrafica(1);
            if (dos==1)redibujado();
             break;
```

```
/*Pantalla de opciones*/
       case 5:
          /*Borra los de intervalos de tiempo de la pantalla
anterior(grafica I) una sola vez*/
          if (tres==1)
          {
              /*Cambiando el color frontal a negro borramos el
intervalo de tiempo*/
              FG_Color = BLACK;
              posicionIntervalo();
              tres=0;
          }
          /*_____________________INTERVALO_____________________*/
          FG_Color = RED;
          TFT_putchBIG(34,52,129);//Simbolo reloj
          if(intervalo==0) TFT_print(77 ,60, "TIEMPO REAL");
            else if(intervalo==1) TFT_print(77 ,60, "10 MINUTOS ");
            else if(intervalo==2) TFT_print(77 ,60, "20 MINUTOS ");
            else if(intervalo==3) TFT_print(77 ,60, "30 MINUTOS ");
            else if(intervalo==4) TFT_print(77 ,60, "1 HORA     ");
          /*_____________________BRILLO_____________________*/
          FG_Color = GREEN;
          TFT_putchBIG(34,97,130);//Simbolo brillo
          /*Para el dibujado del nivel de brillo*/
          b1[0]= brightness+48;
          b1[1]= 0;
          TFT_print(77,105,b1);
          TFT_print(97 ,105, "BRILLO");
          /*_____________________GRADOS_____________________*/
          FG_Color = YELLOW;
          TFT_putchBIG(34,150,131);//Simbolo termometro
          TFT_print(77 ,150, "GRADOS");
          if(zz1==0)TFT_print(77 ,167, "CENTIGRADOS");
          else if(zz1==1)TFT_print(77 ,167, "FAHRENHEIT");
            break;
    }

}
int opcion=0,comparar=0,cambio=0;
/*Actualizacion de parametros cuando se pulsa el boton 1*/
void pulsarBoton1(void)
{
    /*Cambio de pantalla y cambio de opciones en el menu de ajustes*/
    if(activadoCambio==0)
    {
        /*Cada vez que se pulsa NEXT, cambiamos de opcion*/
        if((m==5) &(opcion!=0))
        {
            opcion++;
            /*Marcado de opcion que podemos variar*/
            if(opcion==2)
            {
                FG_Color = BLACK;
                TFT_putchBIG(0,52,132);//Simbolo flecha borra la
opcion de intervalo de tiempo
                FG_Color = GREEN;
                TFT_putchBIG(0,97,132);//Simbolo flecha marca la
opcion de brillo
            }
                else if(opcion==3)
                {
                    FG_Color = BLACK;
```

```
                    TFT_putchBIG(0,97,132);//Simbolo flecha borra la
opcion de brillo
                    FG_Color = YELLOW;
                    TFT_putchBIG(0,150,132);//Simbolo flecha marca la
opcion de grados
                }
            if(opcion==4)
            {
                opcion=0;
                FG_Color = BLACK;
                TFT_putchBIG(0,150,132);//Simbolo flecha borra la
opcion de grados
            }
        }
        /*Si no entramos en opciones, NEXT cambia de pantalla */
        else if ((m==5) &(opcion==0))m++;
        /*Cambio de pantalla y borrado*/
        if(m!=5)
        {
            m++;//incremento posicion pantalla
            dimensiones(0,35,240,164,BLACK);//borrado pantalla
        }

        /*Actualizacion graficas*/
        if(m==3) uno=1;
            else if(m==4) dos=1;
        /*Reinicio posicion de pantalla*/
        if(m>=6) m=0;
        /*Para una primera representacion de valores y borrado de
valores previos*/
        if((m==0)| (m==1) | (m==2)| (m==5) )tres=1;
    }
    /*En el momento que el boton OK se presiona, el boton NEXT cambia
su funcionamiento para cambiar intervalos de tiempo dentro
     *de la hora establecida con el boton OK*/
    else if(activadoCambio==1)
    {
        /*En caso de visualizar horas en el intervalo, no es necesario
realizar nada de lo siguiente*/
        if(intervalo==4)goto fin4;
        // intervalo=1 -> 10 min /intervalo=2 ->20 min /intervalo=3 ->
30 min /intervalo=4 1 hora
        if(activadoCambio2!=0)//para empezar siempre en el intervalo
mas pequeño para representar
        {
            /*____________________ULTIMA HORA____________________*/
            if(posicion1hora==posicionHora)//cuando sea la ultima hora
que se esta muestreando
            {
                comparar=graf/600;//Bloques de 10 min que llevamos
muestreados
                /*Se incrementan las variables posicion si hay valores
para mostrar*/
                if((comparar>=1) & (intervalo==1))posicion10min++;
                else if((comparar>=2) &
(intervalo==2))posicion20min++;
                else if((comparar>=3) &
(intervalo==3))posicion30min++;
                /*Se actulizan a 0 las variables de posicion*/
                if(posicion10min==(comparar+1))posicion10min=0;
                else
if(posicion20min==((comparar/2)+1))posicion20min=0;
```

```
                else
if(posicion30min==((comparar/3)+1))posicion30min=0;
            }
            /*_____________________MEMORIA_____________________*/
            else if (posicion1hora<posicionHora)
            {
                /*Se incrementan las variables posicion*/
                if(intervalo==1)posicion10min++;
                else if(intervalo==2)posicion20min++;
                else if(intervalo==3)posicion30min++;
                /*Se actulizan a 0 las variables de posicion*/
                if(posicion10min==6)posicion10min=0;
                else if(posicion20min==3)posicion20min=0;
                else if(posicion30min==2)posicion30min=0;
            }
        }
        if(activadoCambio2==0)//Se dibuja una unica vez
        {
            /*Marcado selecion de minutos y borrado seleccion horas
dependiente de la rotacion de pantalla*/
            if(rotacion==0)
            {
                dimensiones(126,26,101,3,BLACK);
                dimensiones(126,226,114,3,CYAN);
            }
            else if(rotacion==1)
            {
                dimensiones(3,26,101,3,BLACK);
                dimensiones(126,26,114,3,CYAN);
            }
            else if(rotacion==2)
            {
                dimensiones(3,226,101,3,BLACK);
                dimensiones(3,26,114,3,CYAN);
            }
            else if(rotacion==3)
            {
                dimensiones(126,226,101,3,BLACK);
                dimensiones(3,226,114,3,CYAN);
            }
        }
        activadoCambio2=1;//Cambiamos el funcionamiento del boton OK
        fin4:
        /*Si se analiza en intervalos de 1 hora, al pulsar NEXT se
vuelve a la ultima hora muestreada*/
        if(intervalo==4)
        {
            /*Reinicio de variables*/
            activadoCambio2=0;
            activadoCambio=0;
            posicion1hora=0;
            lecturaMemoria=0;
            /*borrado marcado de horas al salir de seleccion*/
            if(rotacion==0)dimensiones(126,26,101,3,BLACK);
            else if(rotacion==1)dimensiones(3,26,101,3,BLACK);
            else if(rotacion==2)dimensiones(3,226,101,3,BLACK);
            else if(rotacion==3)dimensiones(126,226,101,3,BLACK);
        }
        /*Actualizacion graficas*/
            if(m==3) uno=1;
                else if(m==4) dos=1;
    }
```

```
}
/*Actualizacion de parametros cuando se pulsa el boton 2*/
void pulsarBoton2(void)
{
    /*Se elige la hora a analizar siempre que no sea en TIEMPO REAL
     *Siempre que se pulse el boton OK se muestra la hora a analizar
completa*/
    if(((m==3)|(m==4))&(intervalo!=0))
    {
        /*Si hay alguna hora almacenada*/
        if(posicionHora>=1)
        {
            if(activadoCambio!=0)posicion1hora++;//para empezar
siempre en la hora mas antigua y no incrementar posicion1hora
            lecturaMemoria=1;
        }
        /*Reiniciamos posicion1hora al llegar al final*/
        if(posicion1hora==(posicionHora+1))posicion1hora=0;
        /*si es la ultima hora no lee de memoria*/
        if(posicion1hora==posicionHora)lecturaMemoria=0;
        /*Marcado seleccion de horas*/
        if(activadoCambio==0)
        {
            if(rotacion==0)dimensiones(126,26,101,3,CYAN);
            else if(rotacion==1)dimensiones(3,26,101,3,CYAN);
            else if(rotacion==2)dimensiones(3,226,101,3,CYAN);
            else if(rotacion==3)dimensiones(126,226,101,3,CYAN);
        }
        /*Cambia el funcionamiento del boton Next como seleccion de
minutos*/
        activadoCambio=1;
        if(m==3) uno=1;
           else if(m==4) dos=1;
    }
    /*Una vez analizada la hora y sus minutos segun el rango, volvemos
al final de la hora muestreada en tiempo real*/
    if(activadoCambio2==1)
    {
        /*Reinicio de variables*/
        activadoCambio2=0;
        activadoCambio=0;
        posicion10min=0;
        posicion20min=0;
        posicion30min=0;
        posicion1hora=0;
        lecturaMemoria=0;
        /*borrado marcado de minutos al salir de seleccion*/
        if(rotacion==0)dimensiones(126,226,114,3,BLACK);
        else if(rotacion==1)dimensiones(126,26,114,3,BLACK);
        else if(rotacion==2)dimensiones(3,26,114,3,BLACK);
        else if(rotacion==3)dimensiones(3,226,114,3,BLACK);
    }
    /*Variamos la opcion en la que nos situamos*/
    if((m==5) &(opcion!=0))
    {
        /*Cambia el intervalo de tiempo a analizar en la grafica*/
        if(opcion==1)
        {
            intervalo++;
            if(intervalo==5)intervalo=0;
        }
        /*Cambia el nivel de brillo*/
```

```
        else if(opcion==2)
        {
            brightness++;
            if(brightness==6)brightness=0;
            brillo();
        }
        /*Cambia la visualizacion de los grados de centigrados a
fahrenheit*/
        else if(opcion==3)
        {
            zz1++;
            if(zz1==2)zz1=0;
        }
    }
    /*Entramos en las opciones*/
    else if((m==5) & (opcion==0))
    {
        FG_Color = RED;
        TFT_putchBIG(0,52,132);//Simbolo flecha marca la opcion de
intervalo de tiempo
        opcion++;
    }
    /*Cambio en la rotacion de la pantalla siempre que estemos en
cualquiera de las 3 primeras pantallas*/
    if((m==0)|(m==1)|(m==2))
    {
        rotacion++;
        dimensiones(0,0,240,240,BLACK);//BORRADO PANTALLA COMPLETA
        if(rotacion==4)rotacion=0;//cuando llega al limite se reinicia
        setRotation(rotacion);//Establecimiento de rotacion
        marco();
        /*Para representar de nuevo los valores que han sido borrados
al girar la pantalla*/
        tres=1;
    }
}

int pinta=0,muestra=0,media=0,cuenta=0,t=0;
unsigned int long mediaV=0,mediaI=0;
/*Almacena las variables muestreadas en cada interrupcion*/
void muestreo (void)
{
    /*Se almacenan los valores obtenidos del ADC*/
    vin1= vin1 + ADC1BUF4;
    vibat1= vibat1 + ADC1BUF5;
    stemp1= stemp1 + ADC1BUF9;
    /*se actualizan las banderas*/
    muestra=0;//Actualizacion de banderas
    media++;//cuenta del numero de muestras obtenidas
}
/*Hace una media de las variables muestreadas y aztualiza parametros*/
void calculovalores (void)
{
    /*se halla la media de los valores muestreados en medio segundo*/
    vin2=vin1/media;
    vibat2=vibat1/media;
    stemp2=stemp1/media;
    /*se reinician las variables*/
    vin1= 0;
    vibat1= 0;
    stemp1= 0;
```

```
    /*Actualizacion de parametros para su visualizado en pantalla y
almacenamiento de mili amperios y mili vatios hora*/
    valores();
    cuenta++;//Contador de cada medio segundo
    /*Se almacena el valor de V y I*/
    mediaV=mediaV+ vin;
    mediaI=mediaI+ ibat;
}
int main(void)
{
    int i;
    SYSTEM_Initialize(); //inicialización del dispositivo
    __delay_ms(300);// Tiempo de retardo para encender pantalla
    tft_init();// Inicialización de la pantalla
    AD1CON1bits.SAMP = 1;//inicializar muestreo
    dimensiones(0,0,240,240,BLACK);//BORRADO PANTALLA COMPLETA
    marco();//Dibujado del marco de pantalla
    while(1)
    {
        if(muestra==1)muestreo(); //Cuando muestra=1, han transcurrido
10 ms
        if(pinta==1) //Cuando pinta=1, ha transcurrido 0.5 segundos
        {
            calculovalores();
            if(cuenta==2)  //Cuando cuenta=2, ha transcurrido 1
segundo.
            {
                /*Se almacenan los valores de V y I en los vectores*/
                gvolt[t]=mediaV/2;
                gcor[t]=mediaI/2;
                t++;//Se incrementa la posicion del vector
                /*Reiniciamos banderas*/
                cuenta=0;
                mediaV=0;
                mediaI=0;
                graf=t;//cuenta total del numero de valores obtenidas
hasta el momento
                /*En el momento que llevemos 60 min almacenados(3600
segs), se guardan los maximos y minimos de cada 4 segs en la memoria
de programa*/
                if(graf==3600)
                {
                    /*Almacena hasta 18 horas*/
                    if(posicionHora<=17)GuardarFlashHora();//Se
almacena 1 hora en la memoria de programa
                    /*Reiniciamos banderas*/
                    t=0;
                    graf=0;
                    lm1=0;//Reinicio posicion inicial de valores
                    /*Si estamos visualizando la ultima hora, como
pasa a almacenarse, debemos activar la lectura de memoria*/

if((activadoCambio==1)&((posicion1hora+1)==posicionHora))lecturaMemori
a=1;
                    for(i=0;i<3600;i++)//limpiamos los vectores de
datos
                    {
                        gvolt[i]=0;
                        gcor[i]=0;
                    }
                }
            }
```

```
            pantalla(m);//actualizacion de pantalla
            /*Reinicio de banderas*/
            pinta=0;
            media=0;
        }
        /*_____________________________BOTON
NEXT_____________________________*/
        if(BTN1_GetValue()==0)
        {
            pulsarBoton1();//Ejecucion del boton
            i=0;
            /*Para evitar mas de 1 incremento por pulsacion, se
mantiene en bucle mientras se pulse el boton hasta un maximo de 1
segundo
             para evitar permanecer mas de 1 segundo con el sistema
anclado en el bucle se aplica el break*/
            while(BTN1_GetValue()==0)
            {
                i++;
                __delay_ms(10);
                if(i==100)break;//sale cuando lleva 1 segundo
            }
        }
        /*_____________________________BOTON
OK_____________________________*/
        if(BTN2_GetValue()==0)
        {
            pulsarBoton2();//Ejecucion del boton
            i=0;
            /*Igual que en el caso del boton anterior declarado*/
            while(BTN2_GetValue()==0)
            {
                i++;
                __delay_ms(10);
                if(i==100)break;//sale cuando lleva 1 segundo
            }
        }
    }
return 1;
}

/*Interrupcion del temporizador cada 10 ms*/
void __attribute__ ( ( interrupt, no_auto_psv ) ) _T2Interrupt (  )
{
    AD1CON1bits.SAMP = 1;//Activado muestreo
    count++;//Cuenta para cada medio segundo
    count1++;//Cuenta para cada segundo
    muestra=1;//activado flag de muestreo
    if(count1==100)//Activado cuando transcurre 1 segundo
    {
        /*Actualizado del reloj*/
        segs++;
        count1=0;
        if(segs== 60)
        {
            segs=0;
            min++;
            if(min==60)
            {
                min=0;
                hora++;
            }
```

```
        }
    }
    if(count==50) //Activado cuando transcurren 0.5 segundos
    {
        pinta=1;//Refresco de pantalla
        count=0;//Reinicio de variable
    }
    IFS0bits.T2IF = 0; //Actualiza el flag de la interrupcion a 0
}
```

8.2. Código adc1.c

```
#include <xc.h>
#include "adc1.h"


void ADC1_Initialize (void)
{
   //CSCNA enabled(Escanear pines seleccionados); NVCFG0 AVSS; PVCFG
AVDD; ALTS disabled; BUFM disabled;
   //SMPI Generates interrupt after completion of every 5th
sample/conversion operation; BUFREGEN enabled;
   AD1CON2 = 0xC10;

   // SAMC 31; EXTSAM disabled; PUMPEN disabled; ADRC FOSC/2; ADCS 15;
   /*Valor de Tad es 15* Tcy, y el tiempo de adquisicion son 31 Tad*/
   AD1CON3 = 0x1F0F;

   // CSS30 disabled; CSS29 disabled; CSS28 disabled;
   AD1CSSH = 0x00;

   //CSS9 enabled; CSS8 disabled; CSS7 disabled; CSS6 disabled; CSS5
enabled;
   //CSS4 enabled; CSS3 disabled; CSS2 disabled; CSS15 disabled; CSS1
enabled; CSS14 disabled;
   //CSS0 enabled; CSS13 disabled; CSS12 disabled; CSS11 disabled;
CSS10 disabled;
   /*Seleccion de canales a muestrear-AN0-AN1-AN4-AN5-AN9*/
   AD1CSSL = 0x233;

   // DMABL Allocates 1 word of buffer to each analog input;
   AD1CON4 = 0x00;

   // ASEN enabled; WM Legacy operation; ASINT No interrupt; CM Less
Than mode; BGREQ disabled;
   //CTMREQ disabled; LPEN disabled;
   AD1CON5 = 0x8000;

   //Actualizacion de buffers del ADC a 0
   ADC1BUF0 = 0;
   ADC1BUF1 = 0;
   ADC1BUF2 = 0;
   ADC1BUF3 = 0;
   ADC1BUF4 = 0;
   ADC1BUF5 = 0;
   ADC1BUF6 = 0;
   ADC1BUF7 = 0;
   ADC1BUF8 = 0;
   ADC1BUF9 = 0;
   ADC1BUF10 = 0;
   ADC1BUF11 = 0;
   ADC1BUF12 = 0;
```

```
    ADC1BUF13 = 0;
    ADC1BUF14 = 0;
    ADC1BUF15 = 0;
    /* ASAM enabled; DMABM disabled; ADSIDL disabled; DONE disabled;
    *DMAEN disabled; FORM Absolute decimal result, unsigned, right-
justified;
    *SAMP disabled; SSRC Clearing sample bit ends sampling and starts
conversion; MODE12 12-bit; ADON enabled;*/
    /*Se declara el ultimo, ya que tiene el bit para encender el
conversor(ADON)*/
    AD1CON1 = 0x8406;

}
```

8.3. Código clock.c

```
#include <stdint.h>
#include "xc.h"
#include "clock.h"
/*Configuración del reloj*/
void CLOCK_Initialize(void)
{
    // CPDIV 1:1; PLLEN disabled; DOZE 1:8; RCDIV LPRC; DOZEN
disabled; ROI disabled;
    CLKDIV = 0x3500;
    // STOR disabled; STORPOL Interrupt when STOR is 1; STSIDL
disabled; STLPOL Interrupt when STLOCK is 1; STLOCK disabled; STSRC
SOSC; STEN disabled; TUN Center frequency;
    OSCTUN = 0x00;
    // ROEN disabled; ROSEL FOSC; ROSIDL disabled; ROSWEN disabled;
ROOUT disabled; ROSLP disabled;
    REFOCONL = 0x00;
    // RODIV 0;
    REFOCONH = 0x00;
    // DCOTUN 0;
    DCOTUN = 0x00;
    // DCOFSEL 8; DCOEN disabled;
    DCOCON = 0x700;
    // DIV 0;
    OSCDIV = 0x00;
    // TRIM 0;
    OSCFDIV = 0x00;
    // AD1MD enabled; T3MD enabled; T1MD enabled; U2MD enabled; T2MD
enabled; U1MD enabled; SPI2MD enabled; SPI1MD enabled; I2C1MD enabled;
    PMD1 = 0x00;
    // IC3MD enabled; OC1MD enabled; IC2MD enabled; OC2MD enabled;
IC1MD enabled; OC3MD enabled;
    PMD2 = 0x00;
    // PMPMD enabled; RTCCMD enabled; CMPMD enabled; CRCMD enabled;
I2C2MD enabled;
    PMD3 = 0x00;
    // CTMUMD enabled; REFOMD enabled; LVDMD enabled;
    PMD4 = 0x00;
    // CCP2MD enabled; CCP1MD enabled; CCP4MD enabled; CCP3MD enabled;
CCP5MD enabled;
    PMD5 = 0x00;
    // SPI3MD enabled;
    PMD6 = 0x00;
    // DMA1MD enabled; DMA0MD enabled;
    PMD7 = 0x00;
    // CLC1MD enabled; CLC2MD enabled;
    PMD8 = 0x00;
```

```
    // CF no clock failure; NOSC PRIPLL; SOSCEN disabled; POSCEN
disabled; CLKLOCK unlocked; OSWEN Switch is Complete; IOLOCK not-
active;
    __builtin_write_OSCCONH((uint8_t) ((0x03 << _OSCCON_NOSC_POSITION
) >> 0x08 ));
    __builtin_write_OSCCONL((uint8_t) ((0x300 | _OSCCON_OSWEN_MASK) &
0xFF));
    // Wait for Clock switch to occur
    while (OSCCONbits.OSWEN != 0);
    while (OSCCONbits.LOCK != 1);// Esperamos a que el PLL se enganche
}
```

8.4. Código interrup_manager.c

```
#include <xc.h>

/*Configuración de las interrupciones*/
void INTERRUPT_Initialize (void)
{
    /*Se establece una prioridad de 1 a las interrupciones del
temporizador*/
        IPC1bits.T2IP = 1;
}
```

8.5. Código mcc.c

```
/*Configuración del microcontrolador*/

// FSEC
#pragma config BWRP = OFF    //Boot Segment Write-Protect bit->Boot
Segment may be written
#pragma config BSS = DISABLED    //Boot Segment Code-Protect Level
bits->No Protection (other than BWRP)
#pragma config BSEN = OFF    //Boot Segment Control bit->No Boot
Segment
#pragma config GWRP = OFF    //General Segment Write-Protect bit-
>General Segment may be written
#pragma config GSS = DISABLED    //General Segment Code-Protect Level
bits->No Protection (other than GWRP)
#pragma config CWRP = OFF    //Configuration Segment Write-Protect
bit->Configuration Segment may be written
#pragma config CSS = DISABLED    //Configuration Segment Code-Protect
Level bits->No Protection (other than CWRP)
#pragma config AIVTDIS = OFF    //Alternate Interrupt Vector Table
bit->Disabled AIVT

// FBSLIM
#pragma config BSLIM = 8191    //Boot Segment Flash Page Address Limit
bits->8191

// FOSCSEL
#pragma config FNOSC = FRC    //Oscillator Source Selection->FRC
#pragma config PLLMODE = PLL96DIV2    //PLL Mode Selection->96 MHz
PLL. Oscillator input is divided by 2 (8 MHz input)
#pragma config IESO = ON    //Two-speed Oscillator Start-up Enable
bit->Start up device with FRC, then switch to user-selected oscillator
source

// FOSC
#pragma config POSCMD = XT    //Primary Oscillator Mode Select bits-
>XT Crystal Oscillator Mode
#pragma config OSCIOFCN = ON    //OSC2 Pin Function bit->OSC2 is
general purpose digital I/O pin
```

```
#pragma config SOSCSEL = OFF    //SOSC Power Selection Configuration
bits->Digital (SCLKI) mode
#pragma config PLLSS = PLL_PRI    //PLL Secondary Selection
Configuration bit->PLL is fed by the Primary oscillator
#pragma config IOL1WAY = ON    //Peripheral pin select configuration
bit->Allow only one reconfiguration
#pragma config FCKSM = CSECMD    //Clock Switching Mode bits->Clock
switching is enabled,Fail-safe Clock Monitor is disabled

// FWDT
#pragma config WDTPS = PS32768    //Watchdog Timer Postscaler bits-
>1:32768
#pragma config FWPSA = PR128    //Watchdog Timer Prescaler bit->1:128
#pragma config FWDTEN = OFF    //Watchdog Timer Enable bits->WDT and
SWDTEN disabled
#pragma config WINDIS = OFF    //Watchdog Timer Window Enable bit-
>Watchdog Timer in Non-Window mode
#pragma config WDTWIN = WIN25    //Watchdog Timer Window Select bits-
>WDT Window is 25% of WDT period
#pragma config WDTCMX = WDTCLK    //WDT MUX Source Select bits->WDT
clock source is determined by the WDTCLK Configuration bits
#pragma config WDTCLK = LPRC    //WDT Clock Source Select bits->WDT
uses LPRC

// FPOR
#pragma config BOREN = ON    //Brown Out Enable bit->Brown Out Enable
Bit
#pragma config LPCFG = OFF    //Low power regulator control->No
Retention Sleep
#pragma config DNVPEN = ENABLE    //Downside Voltage Protection Enable
bit->Downside protection enabled using ZPBOR when BOR is inactive

// FICD
#pragma config ICS = PGD1    //ICD Communication Channel Select bits-
>Communicate on PGEC1 and PGED1
#pragma config JTAGEN = OFF    //JTAG Enable bit->JTAG is disabled

// FDEVOPT1
#pragma config ALTCMPI = DISABLE    //Alternate Comparator Input
Enable bit->C1INC, C2INC, and C3INC are on their standard pin
locations
#pragma config TMPRPIN = OFF    //Tamper Pin Enable bit->TMPRN pin
function is disabled
#pragma config SOSCHP = ON    //SOSC High Power Enable bit (valid only
when SOSCSEL = 1->Enable SOSC high power mode (default)
#pragma config ALTI2C1 = ALTI2CEN    //Alternate I2C pin Location-
>SDA1 and SCL1 on RB9 and RB8

#include "mcc.h"
#include "clock.h"


void OSCILLATOR_Initialize(void)
{
    CLOCK_Initialize();
}
```

8.6. Código oc1.c

```
#include <xc.h>
```

```
#include "oc1.h"

/*Configuración del comparador*/
void OC1_Initialize (void)
{
    /*El periodo minimo de la señal PWM es la conversion que se aplica
en el timer2, en este caso es 1/256,
     *por lo tanto se tiene una frec de 62.5kHz con un periodo de 16us
*/
    OC1R = 124; //Ciclo de trabajo de PWM, con un maximo de OC1RS
    OC1RS = 625-1;//Define el periodo de la señal PWM, con un valor
maximo definido por el timer2(10ms)
                  //Con un valor de 625, el periodo maximo de señal de
10ms

    //ENFLT0 disabled; ENFLT1 disabled; OCFLT2 disabled; ENFLT2
disabled; OCSIDL disabled; OCM Edge-Aligned PWM mode;
    //OCFLT1 disabled; OCFLT0 disabled; OCTSEL TMR2 Seleccion del
timer 2; TRIGMODE Only Software;
    OC1CON1 = 0x06;//OCTSEL TMR2 (Seleccion del timer 2); OCM Edge-
Aligned PWM mode
    //SYNCSEL OC1; TRIGSTAT disabled; DCB Start of instruction cycle;
    //OCINV disabled; OCTRIG Sync; OC32 disabled; FLTOUT disabled;
    //OCTRIS disabled; FLTMD Cycle; FLTTRIEN disabled;
    OC1CON2 = 0x01;
}
```

8.7. Código pin_manager.c

```
#include <xc.h>
#include "pin_manager.h"
/*Configuración de pines*/
void PIN_MANAGER_Initialize (void)
{

/**************************************************************************
     * Setting the Output Latch SFR(s)

**************************************************************************/
    LATA = 0x0000;
    LATB = 0x0EC0;//Valor de pines de salida en
alta(BTN1,BTN2,RST,SD_CS,TFT_CS)(RB6-7-9-10-11)


/**************************************************************************
     * Setting the GPIO Direction SFR(s)

**************************************************************************/
    TRISA = 0x0017;//Pines de entrada(Dmas,Dmenos,OSCI)(RA0-1-2)/Pines
de salida(OSCO)(RA3)
    TRISB = 0xC0DF;//Pines de
entrada(PGD1,PGC1,VIN,VIBAT,BTN1,BTN2,SPI.MISO,STEMP)(RB0-1-2-3-4-6-7-
14-15)/
                   //Pines de
salida(LED,DC,RST,SD_CS,TFT_CS,SPI.CLOCK,SPI.MOSI)(RB5-8-9-10-11-12-
13)


/**************************************************************************
     * Setting the Weak Pull Up and Weak Pull Down SFR(s)
```

```
*****************************************************************/
    IOCPDA = 0x0000;
    IOCPDB = 0x0000;
    IOCPUA = 0x0000;
    IOCPUB = 0x00C0; //Pull UP interno (BTN1,BTN2)(RB6-7)


/*****************************************************************
     * Setting the Open Drain SFR(s)

*****************************************************************/
    ODCA = 0x0000;
    ODCB = 0x0000;


/*****************************************************************
     * Setting the Analog/Digital Configuration SFR(s)

*****************************************************************/
    ANSA = 0x0003;//Pin analogico(Dmas,Dmenos)(RA0-1) el resto se
quedan como digitales
    ANSB = 0x800C;//Pin analogico(VIN,VIBAT,STEMP)(RB2-3-15)


/*****************************************************************
     * Set the PPS

*****************************************************************/
    __builtin_write_OSCCONL(OSCCON & 0xbf); // unlock PPS

    RPOR5bits.RP11R = 0x0009;    //RB11->SPI1:SS1OUT
    RPOR2bits.RP5R = 0x000D;     //RB5->OC1:OC1
    RPINR20bits.SDI1R = 0x000E;  //RB14->SPI1:SDI1
    RPOR6bits.RP12R = 0x0008;    //RB12->SPI1:SCK1OUT
    RPOR6bits.RP13R = 0x0007;    //RB13->SPI1:SDO1

    __builtin_write_OSCCONL(OSCCON | 0x40); // lock PPS

}
```

8.8. Código spi1.c

```
#include <xc.h>
#include "spi1.h"

/*Configuración del Protocolo SPI1*/
void SPI1_Initialize (void)
{
    int i = 0;
    // AUDEN disabled; FRMEN disabled; AUDMOD I2S; FRMSYPW One clock
wide; AUDMONO stereo; FRMCNT 0; MSSEN disabled;
    //FRMPOL disabled; IGNROV disabled; SPISGNEXT not sign-extended;
FRMSYNC disabled; URDTEN disabled; IGNTUR disabled;
    SPI1CON1H = 0x00;
    // WLENGTH 0;
    SPI1CON2L = 0x00;
    // SPIROV disabled; FRMERR disabled;
    SPI1STATL = 0x00;
    // SPI1BRGL 0;
    SPI1BRGL = 0x00;
```

```
    // SPITBFEN disabled; SPITUREN disabled; FRMERREN disabled; SRMTEN
disabled; SPIRBEN disabled; BUSYEN disabled;
    //SPITBEN disabled; SPIROVEN disabled; SPIRBFEN disabled;
    SPI1IMSKL = 0x00;
    // RXMSK 0; TXWIEN disabled; TXMSK 0; RXWIEN disabled;
    SPI1IMSKH = 0x00;
    // SPI1URDTL 0;
    SPI1URDTL = 0x00;
    // SPI1URDTH 0;
    SPI1URDTH = 0x00;
    // SPIEN disabled; DISSDO disabled; MCLKEN FOSC/2; CKP Idle:Low,
Active:High; SSEN disabled; MSTEN Master(se establece como maestro);
    // MODE16 disabled; SMP Middle; DISSCK disabled; SPIFE Frame Sync
pulse precedes; CKE Active to Idle;
    //MODE32 disabled; SPISIDL disabled; ENHBUF enabled; DISSDI
disabled;
    //Mode16 y Mode 32 a 0 establecen un ancho datos de comunicacion
de 8 bits
    SPI1CON1L = 0x121;
    i=SPI1BUFL; //Vaciamiento del buffer de transmision
    SPI1CON1Lbits.SPIEN=1; //inicializacion del protocolo de
transimision SPI
}
```

8.9. Código system.c

```
#include "pin_manager.h"
#include "clock.h"
#include "system.h"
#include "interrupt_manager.h"
#include "traps.h"
#include "adc1.h"
#include "spi1.h"
#include "oc1.h"
#include "memory/flash.h"
#include "tmr2.h"

/*Inicialización del microcontrolador*/
void SYSTEM_Initialize(void)
{
    PIN_MANAGER_Initialize();
    INTERRUPT_Initialize();
    CLOCK_Initialize();
    SPI1_Initialize();
    ADC1_Initialize();
    OC1_Initialize();
    TMR2_Initialize();
}
```

8.10. Código tmr2.c

```
#include <xc.h>
#include "tmr2.h"
/*Configuración del timer2*/
void TMR2_Initialize (void)
{

    //TCKPS 1:256(preescalado de fuente de reloj); T32 16 Bit(); TON
enabled (habilita el contador)
    //TSIDL disabled; TCS FOSC/2(Fuente de reloj); TECS SOSC; TGATE
disabled;
```

```
    T2CON = 0x30;
    //TMR2 0;
    TMR2 = 0x00;//Contador inicializado a 0
    //Period = 10 ms; Frequency = 16000000 Hz; PR2 625;
    //Con 625 ciclos de reloj de 16Mhz/256=62500Hz debido al
preescalado, la interrupcion del temporizador salta cada 10ms
    PR2= 625-1;// 625*(1/62500) = 10 ms
      T2CONbits.TON=1; //Activamos el timer
    IFS0bits.T2IF = false;//Puesta a 0 del flag de interrupcion
    IEC0bits.T2IE = true;//Bit de control para habilitacion de las
interrupciones del timer
 }
```

8.11. Código traps.c

```
#include <xc.h>
#include "traps.h"

#define ERROR_HANDLER __attribute__((interrupt,no_auto_psv))
#define ERROR_HANDLER_NORETURN ERROR_HANDLER __attribute__((noreturn))
#define FAILSAFE_STACK_GUARDSIZE 8

/**
 * a private place to store the error code if we run into a severe
error
 */
static uint16_t TRAPS_error_code = -1;

/**
 * Halts
 *
 * @param code error code
 */
void __attribute__((naked, noreturn, weak))
TRAPS_halt_on_error(uint16_t code)
{
    TRAPS_error_code = code;
#ifdef __DEBUG
    __builtin_software_breakpoint();
    /* If we are in debug mode, cause a software breakpoint in the
debugger */
#endif
    while(1);

}

/**
 * Sets the stack pointer to a backup area of memory, in case we run
into
 * a stack error (in which case we can't really trust the stack
pointer)
 */
inline static void use_failsafe_stack(void)
{
    static uint8_t failsafe_stack[32];
    asm volatile (
        "   mov    %[pstack], W15\n"
        :
        : [pstack]"r"(failsafe_stack)
    );
/* Controls where the stack pointer limit is, relative to the end of
the
```

```
 * failsafe stack
 */
    SPLIM = (uint16_t)(((uint8_t *)failsafe_stack) +
sizeof(failsafe_stack)
            - FAILSAFE_STACK_GUARDSIZE);
}

/** Oscillator Fail Trap vector**/
void ERROR_HANDLER_NORETURN _OscillatorFail(void)
{
    INTCON1bits.OSCFAIL = 0;  //Clear the trap flag
    TRAPS_halt_on_error(TRAPS_OSC_FAIL);
}
/** Stack Error Trap Vector**/
void ERROR_HANDLER_NORETURN _StackError(void)
{
    /* We use a failsafe stack: the presence of a stack-pointer error
     * means that we cannot trust the stack to operate correctly
unless
     * we set the stack pointer to a safe place.
     */
    use_failsafe_stack();
    INTCON1bits.STKERR = 0;  //Clear the trap flag
    TRAPS_halt_on_error(TRAPS_STACK_ERR);
}
/** Address Error Trap Vector**/
void ERROR_HANDLER_NORETURN _AddressError(void)
{
    INTCON1bits.ADDRERR = 0;  //Clear the trap flag
    TRAPS_halt_on_error(TRAPS_ADDRESS_ERR);
}
/** Math Error Trap Vector**/
void ERROR_HANDLER_NORETURN _MathError(void)
{
    INTCON1bits.MATHERR = 0;  //Clear the trap flag
    TRAPS_halt_on_error(TRAPS_MATH_ERR);
}
/** NVM Error Trap Vector**/
void ERROR_HANDLER_NORETURN _NVMError(void)
{
    INTCON4bits.SGHT = 0;  //Clear the trap flag
    TRAPS_halt_on_error(TRAPS_NVM_ERR);
}
```

8.12. Código writteflash.c

```
#include <p24FJ256GA702.h>
#include "flash.h"

/*Configuración de la escritura en memoria Flash*/
extern __attribute__((space(prog))) unsigned int long _PROGRAM_END;

/*El tamaño de la pagina que se borra es de 1024 instrucciones, por lo
tanto se pueden escribir 1024 valores
 *StartWritteFlash nos borra la siguiente pagina a continuacion de la
direccion de programa que se introduce en
 *last_code_location. La funcion devuelve la direccion de pagina que
se ha borrado para que a continuacion pueda escribirse
 *last_code_location= direccion de página de programa.
 *x=0 se obtiene la siguiente pagina a continuacion del codigo de
programa y se borra
```

```
 *x!=0 se utiliza la direccion enviada a la funcion, y se borra la
siguiente direccion de pagina a continuacion de esa*/
unsigned int long StartWritteFlash(unsigned int long
last_code_location, int x)
{
    unsigned int long flash_storage_address;//direccion de
almacenamiento de la memoria de programa
    unsigned int result;//las funciones devuelven un valor booleano,
siendo 1 si se realizan correctamente
    /*Obtenemos la direccion de almacenamiento a continuacion de la
memoria de programa(Flash)*/
    if(x==0)
    {
        last_code_location = ((unsigned int long)(__prog__ unsigned
int long*)&_PROGRAM_END) & 0xFFFFFF;
    }
    /*Con la direccion de programa obtenida cuando x=0 o x!=0,
obtenemos la direccion de pagina a continuacion de esta, que se
encuentra a 1024 instrucciones*/
    flash_storage_address =
FLASH_GetErasePageAddress(last_code_location) +
(2*FLASH_ERASE_PAGE_SIZE_IN_INSTRUCTIONS);
    /*Se desbloquea la memoria de programa para poder realizar el
borrado*/
    FLASH_Unlock(FLASH_UNLOCK_KEY);
    /*Borramos la pagina de la memoria de programa que comienza en esa
direccion*/
    result = FLASH_ErasePage(flash_storage_address);
    /*Bucle de control, de manera que hasta que no se haya borrado la
pagina no continua el programa*/
    while(NVMCONbits.WR!=0);
    /*Devuelve la direccion de pagina que ha sido borrada*/
    return flash_storage_address;
}


/*Cada vez que escribimos, almacenamos 2 variables de 16 bits en los
bits menos significativos de la palabra de instruccion <15,0>,
 *lo que quiere decir que se puede llamar un maximo de 512 veces a la
funcion WritteFlash almacenando 1024 valores por pagina
 *Una vez completada la pagina, solo se puede reescribir o escribir
otra pagina si se realiza un borrado de pagina completo previo
 *flash_storage_address=direccion de almacenamiento de los datos
 *dato1= dato de 16 bist a almacenar en (flash_storage_address)
 *dato2= dato de 16 bits a almacenar en (flash_storage_address+2)*/
void WritteFlash(unsigned int long flash_storage_address, unsigned int
dato1, unsigned int dato2)
{
    unsigned int result;
    /*Escritura de datos en la memoria de programa, devuelve 1 sis e
realiza correctamente*/
    result = FLASH_WriteDoubleWord16(flash_storage_address, dato1,
dato2);
    /*Bucle de control, de manera que hasta que no se hayan escrito
los datos no continua el programa*/
    while(NVMCONbits.WR!=0);
}

/*A la hora de leer un valor, debe haber un salto de direccion de 2,
pues cada instruccion tienes dos valores de direccion
 *Ex: una variable guardada en 0x4800, tendra LSB <15,0> en 0x4800, y
MSB <31,16> en 0x4801, en total 32 bits,
```

```
 *Por lo tanto la siguiente variable estará en la direccion 0x4802
 *flash_storage_address=direccion de progama donde se encuentra el
dato a leer
 *La funcion devuelve el valor del dato leido*/
unsigned int ReadFlash(unsigned int long flash_storage_address)
{
   unsigned int valor;
   /*Lectura del dato que se encuentra en la direccion de programa
flash_storage_address*/
   valor = FLASH_ReadWord16(flash_storage_address);
   /*Devuelve el dato leido*/
   return valor;
 }
```

8.13. Código adc1.h

```
#ifndef _ADC1_H
#define _ADC1_H

#include <stdbool.h>
#include <stdint.h>
#include <stdlib.h>

#ifdef __cplusplus  // Provide C++ Compatibility

    extern "C" {

#endif


void ADC1_Initialize (void);


#ifdef __cplusplus  // Provide C++ Compatibility

    }

#endif

#endif //_ADC1_H
```

8.14. Código clock.h

```
#ifndef CLOCK_H
#define     CLOCK_H

#ifndef _XTAL_FREQ
#define _XTAL_FREQ  32000000UL
#endif

void CLOCK_Initialize(void);
#endif      /* CLOCK_H */
```

8.15. Código interrup_manager.h

```
#ifndef _INTERRUPT_MANAGER_H
#define _INTERRUPT_MANAGER_H

void INTERRUPT_Initialize(void);
```

```
#endif
```

8.16. Código mcc.h

```
#ifndef MCC_H
#define MCC_H
#include <xc.h>
#include "system.h"
#include "clock.h"
#include "pin_manager.h"
#include <stdint.h>
#include <stdbool.h>
#include "tmr2.h"
#include "oc1.h"
#include "spi1.h"
#include "interrupt_manager.h"
#include "traps.h"
#include "adc1.h"
#include "memory/flash.h"

#ifndef _XTAL_FREQ
#define _XTAL_FREQ  32000000UL
#endif

void OSCILLATOR_Initialize(void) __attribute__((deprecated ("\nThis
will be removed in future MCC releases. \nUse CLOCK_Initialize (void)
instead. ")));

#endif      /* MCC_H */
```

8.17. Código oc1.h

```
#ifndef _OC1_H
#define _OC1_H

#include <xc.h>
#include <stdint.h>
#include <stdbool.h>

#ifdef __cplusplus  // Provide C++ Compatibility

    extern "C" {

#endif


void OC1_Initialize (void);

#ifdef __cplusplus  // Provide C++ Compatibility

    }

#endif

#endif //_OC1_H
```

8.18. Código pin_manager.h

```
#ifndef _PIN_MANAGER_H
#define _PIN_MANAGER_H

#include <xc.h>
```

```
void PIN_MANAGER_Initialize (void);

 #endif
```

8.19. Código spi1.h

```
#ifndef _SPI1_H
#define _SPI1_H

#include <xc.h>
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>

#ifdef __cplusplus  // Provide C++ Compatibility

    extern "C" {

#endif


#define SPI1_DUMMY_DATA 0x0

#define SPI1_FIFO_FILL_LIMIT 0x8

//Check to make sure that the FIFO limit does not exceed the maximum
allowed limit of 8
#if (SPI1_FIFO_FILL_LIMIT > 8)

    #define SPI1_FIFO_FILL_LIMIT 8

#endif

void SPI1_Initialize (void);


#ifdef __cplusplus  // Provide C++ Compatibility

    }

#endif

 #endif //_SPI1_H
```

8.20. Código system.h

```
#include "xc.h"
#include "stdint.h"

#ifndef SYSTEM_H
#define     SYSTEM_H

void SYSTEM_Initialize(void);
 #endif     /* SYSTEM_H */
```

8.21. Código tmr2.h

```
#ifndef _TMR2_H
#define _TMR2_H

#include <xc.h>
#include <stdint.h>
```

```
#include <stdbool.h>

#ifdef __cplusplus  // Provide C++ Compatibility

    extern "C" {

#endif

#define TMR2_INTERRUPT_TICKER_FACTOR    1


void TMR2_Initialize (void);


#ifdef __cplusplus  // Provide C++ Compatibility

    }

#endif

 #endif //_TMR2_H
```

8.22. Código traps.h

```
#ifndef _TRAPS_H
#define _TRAPS_H
#include <stdint.h>

/**
 * Error codes
 */
typedef enum
{
    /* ----- Traps ----- */
    TRAPS_OSC_FAIL = 0, /** Oscillator Fail Trap vector */
    TRAPS_STACK_ERR = 1, /** Stack Error Trap Vector */
    TRAPS_ADDRESS_ERR = 2, /** Address Error Trap Vector */
    TRAPS_MATH_ERR = 3, /** Math Error Trap Vector */
    TRAPS_NVM_ERR = 7, /** NVM Error Trap Vector */
} TRAPS_ERROR_CODE;

void __attribute__((naked, noreturn, weak))
TRAPS_halt_on_error(uint16_t code);

 #endif
```

8.23. Código writteflash.h

```
#ifndef WRITTEFLASH_H
#define WRITTEFLASH_H

unsigned int long StartWritteFlash(unsigned int long
last_code_location, int x);
void WritteFlash(unsigned int long flash_storage_address, unsigned int
dato1, unsigned int dato2);
 unsigned int ReadFlash(unsigned int long flash_storage_address);
```

8.24. Código flash.h

```
#ifndef FLASH_H
#define FLASH_H
```

```
#include <stdint.h>
#include <stdbool.h>
#define FLASH_WRITE_ROW_SIZE_IN_INSTRUCTIONS  128
#define FLASH_ERASE_PAGE_SIZE_IN_INSTRUCTIONS 1024

#define FLASH_ERASE_PAGE_SIZE_IN_PC_UNITS
(FLASH_ERASE_PAGE_SIZE_IN_INSTRUCTIONS*2)
#define FLASH_WRITE_ROW_SIZE_IN_PC_UNITS
(FLASH_WRITE_ROW_SIZE_IN_INSTRUCTIONS*2)
#define FLASH_HAS_ECC  1

#define FLASH_UNLOCK_KEY 0x00AA0055

#define PAGE_MASK (~((FLASH_ERASE_PAGE_SIZE_IN_INSTRUCTIONS*2) - 1))
void     FLASH_Unlock(uint32_t  key);
void     FLASH_Lock(void);

bool     FLASH_ErasePage(uint32_t address);

uint16_t FLASH_ReadWord16(uint32_t address);
uint32_t FLASH_ReadWord24(uint32_t address);

bool     FLASH_WriteDoubleWord16(uint32_t flashAddress, uint16_t
Data0, uint16_t Data1);
bool     FLASH_WriteDoubleWord24(uint32_t address, uint32_t Data0,
uint32_t Data1  );

/* Program the flash one row at a time. */

/* FLASH_WriteRow24: Writes a single row of data from the location
given in *data to
 *                   the flash location in address.  Since the flash
is only 24 bits wide
 *                   all data in the upper 8 bits of the source will
be lost .
 *                   The address in *data must be row aligned.
 *                   returns true if successful */

bool     FLASH_WriteRow24(uint32_t flashAddress, uint32_t *data);

/* FLASH_WriteRow16: Writes a single row of data from the location in
given in *data to
 *                   to the flash location in address. Each 16 bit
source data
 *                   word is stored in the lower 16 bits of each flash
entry and the
 *                   upper 8 bits of the flash is not programmed.
 *                   The address in *data must be row aligned.
 *                   returns true if successful */
bool     FLASH_WriteRow16(uint32_t address, uint16_t *data);

uint16_t FLASH_GetErasePageOffset(uint32_t address);
uint32_t FLASH_GetErasePageAddress(uint32_t address);


 #endif     /* FLASH_H */
```

9. Bibliografía

[1]. “Universal Serial Bus” [online-Wikipedia]
[2]. “Multímetro digital UM24 con conexión USB tipo A 2.0” [online-Amazon]
[3]. “Carga rápida más allá de Quick Charge: Que es, tipos y compatibilidad” [online-Wikiversus]
[4]. “¿Qué es la tecnología Qualcomm Quick Charge?” [online-Belkin]
[5]. “La Ley de Ohm” [online-Wikipedia]
[6]. **“**Equipos Electrónicos de medida y alimentación” Asignatura de E.T.S.I.T. de la Universidad de Valladolid.
[7]. “Microcontrolador PIC24FJ256GA702-I/SS” [online-Datasheet]
[8]. “Sensor de temperatura LMT85DCKT” [online-Datasheet]
[9]. “Amplificador operacional OPA348AIDBVTG4” [online-Datasheet]
[10]. “Modulo TFT-LCD IPS 1.33 pulgadas”[online-Aliexpress]
[11]. “Controlador ST7789VW” [online-Datasheet]
[12]. “Botón B3F-3150” [online-Datasheet]
[13]. “Conector USB tipo A 2.0 macho” [online-Farnell]
[14]. “Conector USB tipo A 2.0 hembra” [online- Farnell]
[15]. “Cristal QCL8.00000F18B23B” [online-Farnell]
[16]. “Calculadora de dimensiones de pista de PCB en relación a la temperatura” [online-Circuitcalculator]
[17]. “Calculadora de dimensiones de pistas de PCB” [online-PCBWay]