Universidad Zaragoza
1542

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

PROYECTO FIN DE CARRERA

Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes

Autor

Carlos E. Jarauta Tramullas

Director

Antonio Bono Nuez

Ingeniería Técnica Industrial, Electrónica Industrial

Septiembre 2015

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

PROPUESTA y ACEPTACIÓN DEL PROYECTO FIN DE CARRERA DE INGENIERÍA TÉCNICA

DATOS PERSONALES

APELLIDOS, Nombre JARAUTA TRAMULLAS CARLOS E

Nº DNI 17715736J Dirección VIA HISPANIDAD, 58 8ºC

C.P. 50009 Localidad ZARAGOZA

Provincia ZARAGOZA Teléfono 619390827 NIA: 386528

Firma:

DATOS DEL PROYECTO FIN DE CARRERA

INGENIERIA TECNICA INDUSTRIAL, Especialidad ELECTRÓNICA INDUSTRIAL

TITULO SISTEMA DE DESARROLLO BASADO EN FREESCALE FREEDOM FRDM-KL25Z CON FINES DOCENTES

PROYECTO TIPO A ☐ TIPO B ☒

DIRECTOR ANTONIO BONO NUEZ

VERIFICACIÓN EN SECRETARÍA

El alumno reúne los requisitos académicos **(1)** para la adjudicación de Proyecto Fin de Carrera

SELLO DEL CENTRO

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

EL FUNCIONARIO DE SECRETARIA

Fdo.:

SE ACEPTA LA PROPUESTA DEL PROYECTO **(2)**	SE ACEPTA EL DEPÓSITO DEL PROYECTO
En Zaragoza, a 30 de JULIO de 2.015	En Zaragoza, a 30 de JULIO de 2.015
Fdo.: Antonio Bono Nuez	Fdo.: Antonio Bono Nuez
DIRECTOR DEL PFC	DIRECTOR DEL PFC

Departamento de Ingeniería Electrónica y Comunicaciones Universidad Zaragoza

(1) Requisitos académicos: tener pendientes un máximo de 24 créditos o dos asignaturas para finalizar la titulación.

(2) Para que la propuesta sea aceptada por el Director, es imprescindible que este impreso esté sellado por la Secretaría de la EINA una vez comprobados los requisitos académicos.

Dedicado a ti,

que no lo has podido compartir conmigo.

A mi compañera en todo y

a mis hijas lo mejor de mi vida.

Unas breves palabras, pero no por ello exentas de cariño y agradecimiento.

A mi Director de Proyecto Antonio por sus consejos, conocimientos y su amistad.

A Boni y a Tomás por su ciencia y humanidad.

A Álvaro, a Félix, a todos mis compañeros de trabajo por su ayuda y apoyo.

Y a toda la gente que en estos años en la Universidad, han compartido vivencias y momentos inolvidables en esta etapa de mi vida, intensa y a la vez gratificante.

"La ciencia puede divertirnos y fascinarnos, pero es la Ingeniería la que cambia el mundo"

Isaac Asimov

Contenido

Memoria

0. Resumen

Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes, es un proyecto realizado con el fin de obtener un sistema de desarrollo versátil y universal, que permita la realización de las prácticas de la asignatura Sistemas Electrónicos Programables (SEP) perteneciente al Grado de Ingeniería Electrónica y Automática de la Universidad de Zaragoza.

El objetivo principal del proyecto es la realización de un sistema de desarrollo que nos posibilite la migración de las prácticas de SEP, realizadas actualmente sobre un microcontrolador de 8 bits, a un microcontrolador ARM Cortex M0+ de 32 bits. Debido a su mayor proliferación, amplia utilización, reducido precio, disponibilidad absoluta, bajo consumo y elevadas prestaciones.

Un microcontrolador es un computador completo (microprocesador, E/S, memoria, otros periféricos), aunque de limitadas prestaciones, que está contenido en el chip de un circuito integrado programable y se destina a gobernar una sola tarea con el programa que reside en su memoria.

La plataforma de desarrollo Freescale Freedom es un conjunto de herramientas de hardware y software para la evaluación y desarrollo. Es ideal para prototipado rápido de aplicaciones basadas en microcontroladores.

La tarjeta Freescale Freedom KL25Z, modelo FRDM-KL25Z, es un diseño rentable y capaz con un microcontrolador Kinetis serie L, uno de los primeros microcontroladores de la industria construido sobre el núcleo ARM ® Cortex ™ M0+.

FRDM-KL25Z puede utilizarse para evaluar dispositivos de la serie Kinetis L. Cuenta con un microcontrolador KL25Z128VLK, un dispositivo con una máxima frecuencia de operación de 48MHz, 128KB de memoria flash, un controlador USB, periféricos analógicos y digitales. La placa FRDM-KL25Z es compatible con el diseño de Arduino ™ R3.

FRDM-KL25Z es la primera plataforma de hardware de Freescale con OpenSDA (sistema abierto estándar serie integrado y adaptador debugger). Este circuito ofrece varias opciones para comunicaciones series, programación flash y control de ejecución de depuración.

El Sistema de Desarrollo basado en Freescale FRDM-KL25Z consta de una placa de periféricos universal, donde se implementan tanto los elementos ya desarrollados en las prácticas de SEP (pulsadores y LEDS de propósito general, display 7 segmentos, teclado, módulo LCD), como nuevos dispositivos electrónicos (memoria EEPROM serie, conversores ADC y DCA, potenciómetro digital, acelerómetro) y diversos sensores (temperatura, presión, humedad).

Además de la placa de circuito impreso, se han realizado varias librerías en lenguaje C, sobre la herramienta de diseño CodeWarrior Development Studio (v10.6_SE), que permiten la comunicación de los periféricos con el sistema de desarrollo Freescale FRDM-KL25Z.

1. Introducción

Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes, es un proyecto realizado con el fin de obtener un sistema de desarrollo versátil y universal, que permita la realización de las prácticas de la asignatura Sistemas Electrónicos Programables (SEP) perteneciente al Grado de Ingeniería Electrónica y Automática de la Universidad de Zaragoza.

SEP se trata de una asignatura anual, tiene 10 créditos ECTS de Tecnología Especifica, se imparte en el tercer curso del Grado de Ingeniería Electrónica y Automática. Está articulada en dos partes. La primera parte se desarrolla en el semestre de otoño y corresponde a la temática de Sistemas Electrónicos. Cuenta con 60 horas presenciales (3h semanales en aula y 5 sesiones de prácticas de 3 horas), corresponde su impartición al área de Tecnología Electrónica.

La segunda parte se desarrolla en el semestre de primavera y corresponde con la temática de Sistemas Empotrados, cuenta con 40 horas presenciales (2h semanales de aula y 4 sesiones de prácticas de 2,5 horas), corresponde su impartición al área de Ingeniería de Sistemas y Automática.

SEP tiene como objetivos el diseño de sistemas electrónicos basados en circuitos programables, constituyendo un sistema empotrado, lo que da lugar a la programación de algoritmos de control. Para ello se requieren conocimientos de electrónica digital, sistemas automáticos y programación.

Para superar la asignatura los alumnos deberán demostrar que distinguen los tipos de circuitos integrados de memoria y diseñan el circuito correspondiente a un mapa de memoria, comprenden la estructura y el funcionamiento básico de un microprocesador. Reconocen microcontroladores, programan dispositivos electrónicos programables y configurables utilizando con soltura sus herramientas de desarrollo.

Conocen las técnicas de conexión de periféricos básicos, diseñan sus circuitos y programas drivers de bajo nivel, diseñan y verifican sistemas electrónicos digitales. Tienen que saber aplicar las técnicas de gestión temporal en la programación de sistemas de tiempo real y por último tienen que saber diseñar y programar una aplicación de tiempo real empotrada.

Existen otras asignaturas dentro de la docencia que se imparte en el departamento de Ingeniería Electrónica y Comunicaciones que pueden utilizar el sistema de desarrollo aquí implementado.

Laboratorio de Diseño Electrónico, Instrumentación Electrónica, tanto las que se imparte en el Grado de Ingeniería Electrónica y Automática como las que pertenece al grado de Ingeniería de Tecnologías y Servicios de Telecomunicación.

El Sistema de Desarrollo basado en Freescale FRDM-KL25Z consta de una placa de periféricos universal, donde se implementan tanto los elementos ya desarrollados en las prácticas de SEP (pulsadores y LEDS de propósito general, display 7 segmentos, teclado, módulo LCD), como nuevos dispositivos electrónicos (memoria EEPROM serie, conversores ADC y DCA, potenciómetro digital, acelerómetros) y diversos sensores (temperatura, presión, humedad).

Además de la placa de circuito impreso, se han realizado varias librerías en lenguaje C, sobre la herramienta de diseño CodeWarrior Development Studio (v10.6_SE), que permiten la comunicación de los periféricos con el sistema de desarrollo Freescale FRDM-KL25Z.

2. Objetivos

El objetivo principal del proyecto es la realización de un sistema de desarrollo que nos posibilite la migración de las prácticas de SEP, realizadas actualmente sobre un microcontrolador de 8 bits, a un microcontrolador ARM Cortex M0+ de 32 bits.

Estas prácticas que se desarrollan actualmente sobre la tarjeta DEMO9S08QG8 de Freescale van a pasar a implementarse sobre una plataforma que incluya un microcontrolador de 32 bits.

Como objetivos del proyecto cabe destacar los siguientes:

- Realizar un estudio comparativo de las familias de microcontroladores basadas en la arquitectura ARM Cortex M0+, incluyendo la elección de la placa DEMO del sistema de desarrollo y evaluación.

- Seleccionar los componentes adecuados y diseñar la PCB del Sistema de Desarrollo, mediante Eagle 7.2.0.

- Construir un prototipo funcional de este Sistema de Desarrollo.

- Desarrollar y testear tanto la placa de periféricos como la plataforma seleccionada.

- Crear una documentación que sirva como ayuda para el alumno, y facilite la utilización del Sistema de Desarrollo y de la placa DEMO seleccionada.

- Implementar varias librerías sobre CodeWarrior Development Studio para este Sistema de Desarrollo.

La necesidad de reducir el consumo de nuestra aplicación, trabajar con tensiones de alimentación menores, la aparición en el mercado de versiones más potentes y de menor coste, con encapsulados cada vez más pequeños, obliga a migrar hacia placas de desarrollo y evaluación cada vez más potentes de precio más reducido.

La placa de desarrollo y evaluación Freescale FRDM-KL25Z nos permite dicha migración a un coste muy reducido, (actualmente la tarjeta comprada en cualquier proveedor cuesta del orden de 15 euros IVA incluido) e incluye un microcontrolador MKL25Z128VLK4 de la familia Kinetis.

Dotaremos a nuestro sistema de desarrollo de diferentes periféricos que posteriormente pasare a explicar, del mismo modo se acompaña el trabajo con distintas librerías en lenguaje C que nos permitan la comunicación y verificación del funcionamiento del sistema de desarrollo implementado.

3. Estudio comparativo

Debido a los avances tecnológicos, se ha pasado en 25 años a una reducción en el tamaño del silicio de 1um a 40nm, multiplicando por 1000 el número de transistores, igualmente dividiendo por 1000 el tamaño de algunas funciones y multiplicando por 1000 la calidad del diseño.

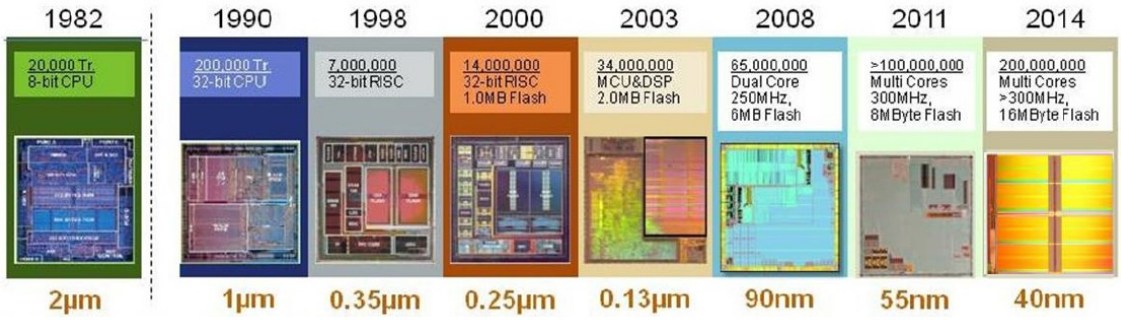


Figura 1. Evolución del silicio en los últimos 25 años.

Un microcontrolador es un computador completo (microprocesador, E/S, memoria, otros periféricos), aunque de limitadas prestaciones, que está contenido en el chip de un circuito integrado programable y se destina a gobernar una sola tarea con el programa que reside en su memoria.

Sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar.

Hay dos arquitecturas distintas relacionadas con el uso y distribución de la memoria:

Arquitectura de von Neumann: Tradicionalmente los sistemas con microprocesadores se basan en esta arquitectura, en la cual la unidad central de proceso (CPU), está conectada a una memoria principal única donde se guardan las instrucciones del programa y los datos. A dicha memoria se accede a través de un sistema de buses único (control, direcciones y datos). En un sistema con arquitectura Von Neumann el tamaño de la unidad de datos o instrucciones está fijado por el ancho del bus que comunica la memoria con la CPU.

Arquitectura Harvard: Este modelo, que utilizan los Microcontroladores PIC, tiene la unidad central de proceso (CPU) conectada a dos memorias (una con las instrucciones y otra con los datos) por medio de dos buses diferentes. Ambos buses son totalmente independientes lo que permite que la CPU pueda acceder de forma independiente y simultánea a la memoria de datos y a la de instrucciones.

El microcontrolador es un sistema cerrado. Todas las partes del computador están contenidas en su interior y sólo salen al exterior las líneas que gobiernan los periféricos. Existe una gran diversidad de microcontroladores.

La clasificación más importante será entre microcontroladores de 8, 16 ó 32 bits.

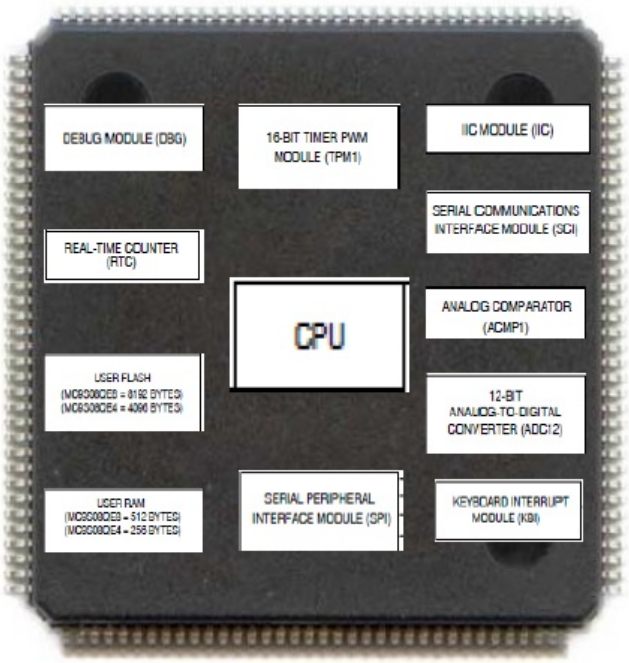


Figura 2. Estructura interna de un microcontrolador.

Actualmente para la realización de las prácticas de la asignatura SEP, se dispone de la placa DEMO9S08QG8 de Freescale, que incluye un microcontrolador MC9S08QG8, el cual pertenece a la familia HCS08. Esta placa DEMO nos permite trabajar de manera sencilla debido a los diferentes bloques que nos ofrece.

Es un producto de Freescale Semiconductor, empresa estadounidense que fue creada a partir de la división del departamento de semiconductores de Motorola, sus características principales son:

Encapsulado DIP16, memoria Flash 8KB, memoria RAM 512B, oscilador interno de 32KHz, 12 GPIO, un puerto de entrada única y otro de salida, bloque de comunicación serie SCI, SPI y I2C, bloque KBI de 8 entradas, bloque PWM con 2 canales y 16 bits, bloque ADC (conversor analógico digital) de 10 bits/8 bits.

Los microcontroladores de 8 bits se utilizan en aplicaciones de bajas prestaciones, muy robustas, alta inmunidad al ruido, de muy bajo coste. Sistemas funcionando a 5 voltios de alimentación con gran escalabilidad.

Por otro lado, los microcontroladores de 32 bits han proliferado en los últimos años, debido a una reducción de costes continua en su fabricación, se usan en aplicaciones de medias y altas prestaciones, sustituyendo en este momento a los microcontroladores de 8 bits por precio y prestaciones.

Integran todo tipo de comunicaciones (Ethernet, USB, Wifi,…), están dotados de un interface de usuario avanzado. Tienen gran capacidad de direccionamiento, son capaces de ejecutar algoritmos matemáticos, disponen de coma flotante (FPU), encapsulados más pequeños, bajo consumo (nuevas tecnologías de silicio) y bajo coste.

Mayores velocidades de trabajo típicas (del orden de 20MHz), capacidad de direccionamiento elevada, integración de sistemas operativos en tiempo real.

Disponen de capacidad de gráficos (SVGA), transacciones interna de datos en paralelo que permite el aumento de la capacidad de proceso, aparición de familias especificas según la aplicación a la que va a ser destinada (comunicaciones, control digital de señales, control de motores, wireless,…).

Cambiar a un microcontrolador de 32 bits nos va a permitir aumentar las prestaciones, reducir el consumo, tener menor tamaño, y posibilidad de ampliar y actualizar las prácticas de la asignatura de SEP.

Principales características micros 32 bits ARM Cortex M0+:

- Núcleo ARM Cortex M0+ a 48MHz.
- Conversor analógico digital de alta velocidad 12/16 bits y digital analógico de 12 bits.
- Comparadores analógicos de alta velocidad.
- Interface de comunicaciones módulos USB, módulos UART, módulos SPI, módulos I2C.
- Alimentación unipolar entre 1,7 y 3,6V.
- Acceso al puerto entradas/salidas hasta un 50% más rápido que el estándar.
- Arquitectura simplificada: 56 instrucciones y 17 registros permiten una programación fácil y un empaquetado eficiente de datos de 8, 16 y 32 bits en memoria.
- Hasta 4 canales DMA para periféricos y servicio de memoria con la mínima intervención de la CPU.
- Extrema eficiencia dinámica: núcleo de 32-bit ARM Cortex-M0 combinado con Tecnología flash de Freescale 90 nm, ofrece ahorro energético 50% respecto un microcontolador de 8 bits.
- Memoria Flash entre 8KB y 256KB, memoria RAM entre 1KB a 32KB SRAM.
- COP Watchdog interno con reloj independiente.
- Módulos de temporizador de propósito general, control PWM, y control específico para motores.
- Pines de salida GPIO con funcionalidad de interrupciones.
- Múltiples opciones interna y externamente de generación de reloj (módulo de MCG con FLL y PLL para sistemas y reloj de CPU), (oscilador de 1 kHz RC de baja potencia para RTC y COP Watchdog).
- Oscilador de cristal externo programable entre 32 kHz y 40 kHz o bien entre 3 MHz y 32 MHz.
- Buses de 16 y 32 bits.

3.1 Familias microcontroladores 32 bits

Un microcontrolador es un circuito integrado que nos ofrece las posibilidades de un pequeño computador. Es decir, que en su interior podemos encontrar un procesador, memorias, y varios periféricos. La arquitectura de un procesador está formada por:

- La arquitectura del set de instrucciones (ISA).

Cuando hablamos de la arquitectura del set de instrucciones, se tendrá en cuenta el tipo de datos con que trabaja (todas las arquitecturas que analizaremos son de 32 bits, todas van a soportar trabajar con datos de 8, 16 y 32 bits), el tipo de instrucciones propiamente dicho, los registros y los modos de direccionamiento, las excepciones/interrupciones y su manejo.

Los sets de instrucciones suele estar comúnmente separados en dos grupos:

- RISC: Reduce Instruction Set Computer. La filosofía de estos dispositivos se concentra en reducir la complejidad de las instrucciones desempeñadas por el hardware, lo que conlleva un aumento de la complejidad del compilador.

- CISC: Complex Instruction Set Computer. Esta filosofía se basa en aumentar la complejidad del hardware para la funcionalidad de las instrucciones, conlleva un set de instrucciones más complicado, pero produce que el compilador utilice menos recursos.

Todas las arquitecturas de procesadores que analizaremos poseen un set de instrucciones tipo RISC. Las principales características de una arquitectura tipo RISC son:

- Gran cantidad y uniformidad de registros, que pueden almacenar datos y direcciones.

- Arquitectura Load-Store. Modos de direccionamiento simples. Cantidad de campos y tamaños de instrucciones fijos.

- Reducido número de instrucciones, las cuales debido a su simplicidad pueden ser ejecutadas en un solo ciclo de máquina. El compilador sintetizara operaciones complejas en múltiples operaciones sencillas.

- Buses de comunicaciones.

Dependerá del tipo de arquitectura Von Neumann o Harvard.

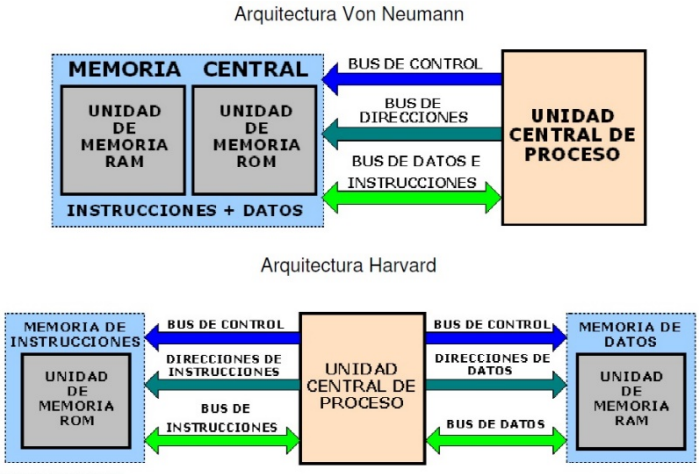


Figura 3. Buses de comunicación en función de la arquitectura de un microcontrolador.

- La segmentación (pipeline).

Segmentación o pipeline, es un método por el cual se consigue aumentar el rendimiento de algunos sistemas electrónicos digitales. La segmentación consiste en descomponer la ejecución de cada instrucción en varias etapas para poder empezar a procesar una instrucción diferente en cada ciclo de máquina y de esta forma trabajar con varias simultáneamente.

Algunos microprocesadores tienen las siguientes etapas en una instrucción:

IF (búsqueda), ID (decodificación), EX (ejecución de unidad aritmético lógica), MEM (memoria), WB (escritura).

Cada una de estas etapas de la instrucción usa en exclusiva un hardware determinado del procesador, de tal forma que la ejecución de cada una de las etapas en principio no interfiere en la ejecución del resto.

Algunos procesadores poseen una segmentación que permite comenzar más de una instrucción por ciclo de máquina, es decir tiene "n" etapas de pipeline y de esta forma conseguir mayor paralelismo.

- El soporte de memoria (virtual/protegida).

La o las unidades de soporte de memoria son dispositivos de hardware formado por un grupo de circuitos integrados, responsable del manejo de los accesos a la memoria por parte de la Unidad de Procesamiento Central (CPU). Entre las funciones de estos dispositivos se encuentran la traducción de las direcciones lógicas a direcciones físicas, la protección de la memoria, el control de caché.

En la actualidad muchos procesadores separan la funciones de traducción de direcciones de memoria y de protección de memoria en dos unidades, llamando a la primera MMU (VMSA como lo suele llamar ARM) y a la segunda MPU (PMSA como lo suele llamar ARM).

Caché: Un caché es un sistema especial de almacenamiento de alta velocidad. Puede ser tanto un área reservada de la memoria principal como un dispositivo de almacenamiento de alta velocidad independiente.

TCM: Memoria fuertemente acoplada (en inglés: Tightly Coupled Memory), es la memoria del tipo que tiene los microcontroladores, a la que se accede directamente.

3.1.1 Arquitectura ARM

En 1990, ARM una empresa surgida de Acorn Computers Limited of Cambridge, England, diseñó el primer microcontrolador comercial de 32 bits de ancho de palabra basado en arquitectura RISC3.

El concepto tecnológico - comercial novedoso que implementaron fue el de diseñar el núcleo de la familia de microcontroladores sin fabricarlos, cedieron la producción a empresas de semiconductores de primera línea (NXP, Atmel, Texas Instruments, Freescale, Analog Devices).

Cada fabricante respetaría el núcleo original y el repertorio de instrucciones, pero pudieron incorporar sus propios periféricos (conversores A/D y D/A, UART, SPI, I2C, etc.). Fue

un estímulo al crecimiento de soluciones en hardware, pero sin embargo, produjeron incompatibilidades entre componentes similares de distintas marcas.

De la arquitectura RISC tomaron:

- Arquitectura *load-store*: Las instrucciones que acceden a memoria son distintas de las instrucciones que procesan los datos. Formato fijo de palabra de instrucción. Todas de 32 bits.

- Máquina de tres direcciones (1er operando, 2º operando y resultado) en la que en la instrucción se trabajará con registros pero no directamente con memoria.

A su vez incorporaron algunos conceptos novedosos como son:

- Las instrucciones siguen un formato repetitivo.

- Implementación de varios modos de trabajo o jerarquías. Aparece por primera vez en un microcontrolador el modo supervisor y el modo usuario. El primero, con plenos poderes, supervisará la actuación de los programas de usuario.

- Excepciones: Son los casos particulares de control de flujo en los que un efecto posiblemente no deseado de la ejecución del programa lleva a una situación de fallo, llevan a que el programa se desvíe de su curso original y pase a ejecutar un tramo de programa que atienda esta situación excepcional. Las interrupciones son un caso particular de las excepciones.

- Thumb: Aparece la posibilidad de operar con códigos de operación comprimidos a 16 bits.

- Bajo consumo: Se partió de la necesidad de bajo consumo, por lo que se bajó la tensión de alimentación y un interesante manejo de la frecuencia de trabajo (ya que la potencia disipada es función directa de la frecuencia) durante el tiempo de ejecución. Se llega a un consumo típico de 0,28 mW/MHz.

- Interrupciones: Se dispondrá de dos tipos de interrupciones IRQ y FIQ las cuales tendrán habilitaciones separadas y FIQ será de mayor prioridad que IRQ. Se tomarán como casos particulares de excepciones.

- Eficiencia en la generación de código en C: Esta arquitectura fue concebida para trabajar en lenguaje C con una densidad de código muy superior a los microcontroladores de 8 bits.

- Herramientas de depuración de bajo costo e incorporadas: Cada microcontrolador dispone de una interfaz JTAG, originada para depurar DSPs y que permiten generar herramientas de depuración en tiempo casi-real de muy bajo costo.

ARM posee un gran número de arquitecturas, las más difundidas son:

ARMv4T (ARM7TDMI y ARM9T).
ARMv5TEJ (ARM926EJ y ARM7EJ).
ARMv6 (ARM11).
ARMv6-M (Cortex-M0).
ARMv7 (Cortex-M3).

- Modos de operación

La mayoría de los ARM poseen múltiples modos de operación:

- Todos poseen un modo usuario (User) en el cual el procesador tiene acceso restringido a distintos recursos del sistema (memoria, registros) y no puede cambiar de modos. Este es el modo en que corren la mayoría de las aplicaciones.

- El modo de sistema (System) es un modo privilegiado que permite el uso de los recursos restringidos, con la salvedad de algunos registros destinados a los distintos modos en los cuales puede entrar el procesador a raíz de una excepción. Esto lo hace adecuado para sistemas operativos.

- Los modos restantes (FIQ, IRQ, Supervisor, Abort, Undefined, Monitor), son modos privilegiados y se accede a ellos a través de excepciones, es por ello que se los denomina modos de excepción. Tienen acceso a los recursos restringidos del sistema y pueden cambiar de modo libremente.

- Registros

La familia AR6 dispone de 16 registros de 32 bits cada uno de ellos que se designan desde R0 a R15. En principio son todos idénticos y sólo dos tienen funciones específicas que son el R15, que se emplea como Contador de Programa (Program Counter), y el R14 (Link Register), que es utilizado para almacenar la dirección de retorno cuando se llama a una subrutina o se genera una excepción. Cabe destacar que es responsabilidad del programador resguardar las direcciones de retorno en el caso de pretender implementar subrutinas anidadas.

Si bien puede utilizarse como puntero a la pila cualquier registro, se empleará el registro R13 por omisión y para ciertas instrucciones como tal.

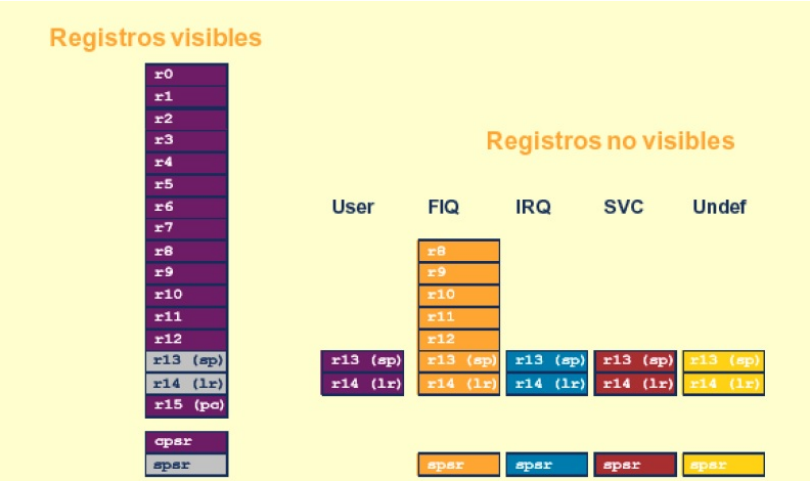


Figura 4. Registros de la arquitectura ARMv6-M.

En la figura 4 se observan los registros del microcontrolador. En todo momento se tendrán 16 registros disponibles más dos registros de estado que almacenarán los tradicionales *flags* de acarreo, cero, signo, etc. y los bits que indican el modo de trabajo y las máscaras de interrupción.

- La arquitectura del set de instrucciones (ISA).

Inicialmente los procesadores ARM tenían un set de instrucciones de 32 bits (set ARM). A partir de la arquitectura ARMv4T, se incorpora un set de instrucciones de 16 bits (set Thumb). Este nuevo set de instrucciones: reduce las funcionalidades del procesador, aumenta la cantidad de instrucciones para realizar una tarea particular, pero disminuye la densidad de código total.

A partir de la arquitectura ARMv6T2, se incorpora el set de instrucciones Thumb 2 (segunda generación del set Thumb), este set de instrucciones es un 25% más rápido que Thumb y un 30% más reducido que el set de instrucciones ARM. Maximiza el cache y el uso de la memoria TCM.

- Excepciones e interrupciones.

Una excepción es un evento que se produce durante la ejecución del programa y que necesita cambiar la secuencia normal de ejecución de las instrucciones, son un tipo especial de excepciones que son causadas por periféricos externos al núcleo, el controlador de interrupciones es el que produce la excepción.

Un concepto importante es la latencia, que definimos como el intervalo de tiempo desde que una señal de interrupción aparece hasta que se ejecuta la primera búsqueda de una instrucción asociada a la rutina de servicio de la interrupción (ISR). Dos métodos principales para reducir la latencia de las interrupciones:

- Interrupciones anidadas, permite que una interrupción ocurra cuando se está ejecutando una rutina de servicio de interrupción previa. Esto se logra reactivando las interrupciones una vez que se atiende la interrupción. Una vez que la interrupción anidada es completada, el control es devuelto a la rutina de servicio de la interrupción original.

- Interrupciones priorizadas, se programa el controlador para que ignore las interrupciones de menor o igual prioridad que la que se encuentre en curso. Esto reduce la latencia de interrupciones de mayor prioridad.

Los procesadores ARMv6-M y ARMv7-M soportan hasta 32 interrupciones externas, las excepciones son numeradas del 1 al 15 y las interrupciones del 16 en adelante, existen 4 niveles de prioridad.

- Segmentación

La mayoría de las instrucciones se ejecutan en un ciclo de máquina, los microcontroladores ARM tienen un pipeline que permite incrementar la cantidad de instrucciones ejecutadas por unidad de tiempo.

Esta arquitectura permite que simultáneamente el procesador está buscando un código de operación, está decodificando una instrucción leída anteriormente y ejecutando una instrucción aún anterior. Ello permite lograr ejecutar (aproximadamente) una instrucción por ciclo de máquina.

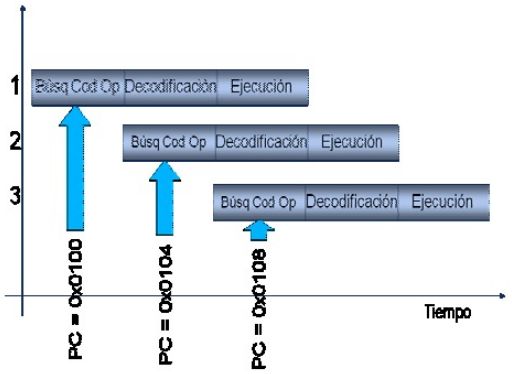


Figura 5. Pipeline de un controlador ARM.

Dentro de la arquitectura ARM tenemos diversos fabricantes, ATMEL, Freescale que producen sus propios microcontroladores. Del mismo modo, existe en el mercado diverso software (CodeWarrior, KEIL, IAR), herramientas (Opensource, OpenSDA), emuladores JTAG, compatibles con las familias ARM.

3.1.2 Arquitectura AVR32

Es una arquitectura de 32 bits producida por ATMEL. La arquitectura AVR32 tiene lo que el fabricante denomina 2 microarquitecturas, las cuales le permiten adaptar un mismo procesador a distintos tipos de aplicaciones.

- AVR32A, pensada para aplicaciones de microcontroladores, tales como AT32UC3A/B.
- AVR32B, pensada para aplicaciones de microprocesadores, tales como la línea AP7000.

- La arquitectura del set de instrucciones

El procesador puede trabajar con datos de 8, 16, 32 y 64 bits. El procesador AVR32A posee dos set de instrucciones.

- Un set de instrucciones compacto, el cual es de 16 bits.
- Un set de instrucciones extendido, el cual es de 32 bits.

Al igual que lo comentado para las arquitecturas ARM, esto permite reducir la densidad de código, todas las instrucciones deben ser puestas en memoria en 16 bits. Las instrucciones más extensas pueden ser colocadas en forma alineada o no.

- Modos de operación

Los procesadores AVR32 poseen múltiples modos de operaciones, los cuales los podemos separar en modos que tienen privilegios y modos sin privilegio. Los primeros son llamados modos de sistema.

El cambio de modo se puede llevar a cabo a través de software, o puede ser causado por una interrupción externa o un proceso de excepción.

En el modo de aplicación, modo sin privilegios, no se puede acceder a la mayoría de los registros de sistema, a la parte superior del registro de estado ni las áreas de memoria protegida.

- Registros

El microcontrolador AVR32A posee 23 registros de propósito general los cuales pueden almacenar tanto datos como direcciones. Todos ellos son de 32 bits y son comúnmente identificados con la letra "R".

Tres de ellos poseen funciones particulares, Stack Pointer, Link Register y Program Counter. Además de estos existe el registro de estado "SR", por lo que en todo momento habrá 17 registros activos.

En el microcontrolador, cuando hay un cambio de modo, producido por una interrupción o por comandos, los registros que nos interesen deben ser guardados. Solo el registro Stack Pointer posee registros de respaldo para los modos con privilegios.

Cuando sucede una excepción el registro de estado (SR) y el de la dirección de retorno son almacenados automáticamente en el Stack Pointer, si lo que sucede es una interrupción son almacenados automáticamente el registro de estado, la dirección de retorno y los registros R8-R12 y LR.

- Excepciones e interrupciones

En el datasheet del procesador AVR32 se los engloba a las excepciones e interrupciones como eventos. El procesador AVR32A posee 28 fuentes distintas de eventos. Cada una de las cuales posee una prioridad y dirección a la cual la subrutina accede cuando el evento es aceptado.

La mayoría de los eventos están ubicados secuencialmente, y en ellos hay una instrucción de salto a la rutina de servicio. Pero existen algunos pocos eventos que tiene espacio suficiente para escribir una pequeña rutina en ellos.

Posee un controlador de interrupciones, en el cual todas las fuentes de interrupción externa poseen una rutina de servicio vectorizadas.

El controlador permite configurar distintos niveles de prioridades e inclusive activar la ejecución de interrupciones anidadas. Como ejemplo el controlador AVR32UC3A maneja hasta 2048 líneas de interrupciones separadas en 64 grupos de 32 interrupciones.

Cuando sucede un evento, y este no está enmascarado, el hardware automáticamente inhibe la generación de eventos de igual o menor prioridad. Luego el contexto es automáticamente salvado. Por defecto son salvados el Status Register y el Program Counter.

- Bus de comunicaciones

Las Arquitecturas AVR32 poseen un bus de comunicaciones con arquitectura Harvard.

- Segmentación

La Arquitectura AVR32A posee 3 etapas de segmentación, mientras que la AVR32B posee 7.

3.1.3 Arquitectura Coldfire

Coldfire es una arquitectura de Freescale. La primera versión del set de instrucciones de Coldfire fue desarrollada como una versión reducida del set del M68000. A medida que la familia de procesadores Coldfire fue creciendo nuevos set de instrucciones se fueron desarrollando para mejorar la densidad de código.

Freescale ofrece varios núcleos con arquitectura Coldfire. La principal diferencia entre los núcleos de Coldfire es la organización del pipeline.

- La arquitectura del set de instrucciones

Los núcleos Coldfire están basados en el concepto de set de instrucciones reducido de longitud variable. La utilización de este tipo de instrucciones permite un aumento de la densidad de código. Mientras muchas de las operaciones que realiza el núcleo utilizan opcodes de 16 bits, la longitud de las instrucciones puede variar entre 16, 32 o 48 bits dependiendo de los modos de direccionamiento.

- Modos de operación y Registros

Los procesadores Coldfire poseen dos modos de operación: modo Supervisor y modo usuario.

En modo usuario todos los núcleos poseen:

- 8 registros de 32 bits para datos.
- 8 registros de 32 bits para direcciones.
- 1 registro de 32 bits como PC.
- 1 registro de 8 bits como registro de condición.

En modo supervisor posee un conjunto de registros adicionales, los cuales no son todos implementados en las distintas versiones.

- Excepciones e interrupciones

Una vez que la excepción es detectada, el procesador realiza los siguientes pasos, lo que no incluye la ejecución de la rutina de excepción:

- Se copia el contenido del Status Register (SR), se conmuta a modo supervisor, se deshabilita el modo trace y se configura la máscara de interrupciones para evitar ser interrumpido por alguna excepción de menor prioridad.
- El procesador determina el número del vector de excepción, para cada una de las excepciones que pertenecen al núcleo existe un número de vector determinado. Si corresponde a una interrupción de un periférico, el procesador realiza un ciclo de reconocimiento de la interrupción para obtener el número del periférico y de esta forma determinar el vector de excepción.
- El procesador salva el contexto (SR y PC).

La tabla de vectores de excepciones contiene 256 entradas. Los primeros 64 vectores son reservados para excepciones del procesador. Los restantes 192 vectores son reservados para el usuario y pueden ser asignados a los periféricos.

Los núcleos Coldfire implementan un controlador de interrupciones de 64 entradas, las cuales están organizadas en 7 niveles de prioridad. Si suceden interrupciones con el mismo nivel de prioridad, se atenderá primero la que tenga número mayor.

- Bus de comunicaciones

Salvo los Coldfire V4, el resto de los procesadores poseen una arquitectura de Bus de comunicaciones tipo Von Neumann.

- Segmentación

Todos los núcleos Coldfire incluyen dos bloques de pipeline independientes y desacoplados. Un pipeline de búsqueda de instrucciones, un pipeline de ejecución.

Ambos bloques de pipeline están conectados por un buffer de instrucciones tipo FIFO, el cual permite búsqueda adelantada de instrucciones.

3.1.4 Arquitectura Power PC

Las Power Architectures las podemos dividir en dos grandes grupos:

- Power, arquitectura creada por IBM para estaciones de trabajo y servers.
- Power PC, arquitectura creada por AIM (Apple, IBM y Motorola) para aplicaciones embebidas y computadoras portátiles.

A diferencia de las arquitecturas previamente vistas, existen varias empresas que evolucionan esta arquitectura. Inclusive existe una organización “power.org” cuya función es desarrollar productos con esta arquitectura y definir especificaciones para nuevas arquitecturas.

Núcleos en la actualidad:

- e200, e300 and e300e/PPC G2, e500, e600/PPC G4, e5500 – Freescale.
- POWER5, POWER6, POWER7, PowerPC 970– IBM.
- PPC400, Titan - Applied Micro Circuits Corporation (AMCC).
- PowerPC 7xx - IBM / Motorola (Freescale).
- Cell - Sony, IBM, Toshiba.

- La arquitectura del set de instrucciones

Existen varias arquitecturas del set de instrucciones, en función de los núcleos existentes. Los núcleos e200 implementan la arquitectura detallada en la especificación ISA v2.03. Esta arquitectura detalla 4 tipos de instrucciones básicas, a las cuales suele asociarse un conjunto de registros.

- Instrucciones de salto.
- Instrucciones de punto fijo, 32 registros de propósito general (GPR) de hasta 64 bits.
- Instrucciones de punto flotante, 32 registros de punto flotante (FPR) de hasta 64 bits.
- Instrucciones de vectores, 32 registros de vectores (VR) de hasta 128 bits.

Independiente de los registros para cada tipo de instrucciones, estas arquitecturas implementan otros registros como LR (Link Register), CTR (Counter Register), etc. La cantidad de bits de estos registros dependerá de cada implementación.

Estas arquitecturas pueden trabajar con datos enteros de 8, 16, 32 y 64 bits, de punto flotante de 32 y 64 bits y vectores de 8, 16, 32 y 128 bits.

- Modos de operación

Los procesadores Power PC poseen dos modos de operación: modo Supervisor y modo usuario.

- Registros

En función del modo de operación existe un conjunto de registros que se pueden acceder desde el procesador. Todas las arquitecturas e200 poseen 32 registros de propósito general.

Ninguna de las arquitecturas e200 implementa los registros de punto flotante ni los registros de vectores. Todas las instrucciones aritméticas que ejecute el núcleo se realizaran sobre los registros de propósito general.

Los núcleo e200z3 y e200z6 implementan registros de propósito general de 64 bits con el fin de soportar las instrucciones de las unidades de punto flotante (FPU).

- Bus de comunicaciones

La gran mayoría de los procesadores Power PC posee una arquitectura de Bus tipo Von Neumann.

- Excepciones e interrupciones

El mecanismo de excepción de la arquitectura Power PC produce un cambio de modo del procesador, el cual pasa a modo supervisor. Cuando sucede una interrupción, la información del estado del procesador es salvada en los registros “Machine State Save/Restore” (SRR0 y SRR1, CSRR0 y CSRR1, DSRR0 y DSRR1 según sea el tipo de interrupción) y el procesador comienza la ejecución de la subrutina de interrupción.

Para prevenir perdidas de la información, la subrutina de interrupciones debe salvar la información almacenada en los registros “Machine State Save/Restore”.

El hardware soporta el anidado de interrupciones críticas dentro de interrupciones no críticas y el de interrupciones de debug dentro de las otras dos.

Los pasos en un proceso de excepción son: reconocimiento de la excepción, se obtiene la dirección de la subrutina de la interrupción y se salva en contexto. Ejecución de la subrutina de interrupciones, como es lógico pensar las excepciones poseen prioridades.

3.2 Elección Placa de Sistema de Desarrollo y Evaluación

Una vez presentado el estudio de varias familias de microcontroladores de 32 bits, he decidido que el microcontrolador sería un ARM Cortex-M0+, debido a:

- Mayor proliferación, amplia utilización.
- Reducido precio.
- Disponibilidad absoluta.
- Bajo consumo.
- Elevadas prestaciones.

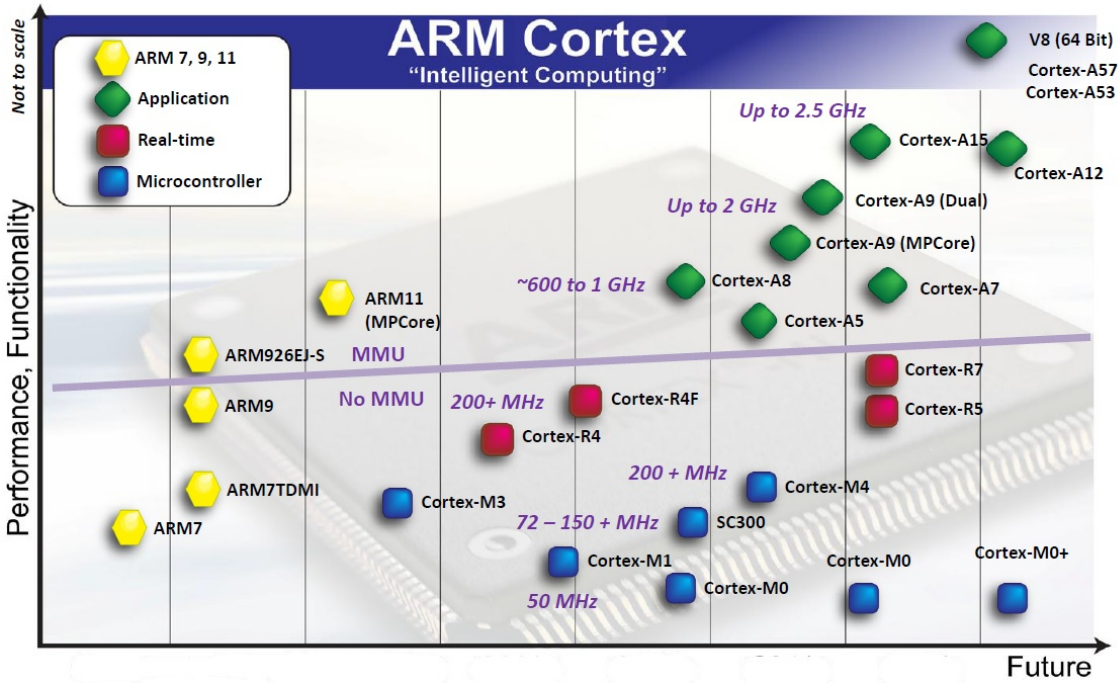


Figura 6. Visión global de los microcontroladores ARM Cortex.

Se trata de un dispositivo que permite trabajar con microcontroladores de 32 bits de una forma sencilla y económica, siendo el primer escalón en los microcontroladores de 32 bits.

Una vez tomada esta decisión queda elegir la placa de desarrollo y evaluación que basándose en esta arquitectura nos permita disponer de un entorno de trabajo. Haciendo uso de la herramienta de diseño Design Center disponible en la página web de Farnell element14 http://www.element14.com/community/community/designcenter que permite una búsqueda sobre dispositivos reales.

A partir de esta herramienta se incluye una tabla en la que se pueden observar diversas opciones de placas de desarrollo y evaluación con sus características básicas. Obsérvese que

una vez decidido por la arquitectura he optado por elegir la placa de desarrollo por lo que el microcontrolador viene determinado por esta elección.

La tabla está organizada de la siguiente forma:

- La columna que aparece en primer lugar es el fabricante de la placa de desarrollo y evaluación.
- A continuación, la referencia de dicho fabricante.
- Después el código Farnell (recordemos que se trata de placas reales que se pueden conseguir fácilmente).
- El precio venta al público viene posteriormente.
- Inmediatamente seguido del tipo de microcontrolador que viene con la placa.
- La tensión de alimentación es el siguiente parámetro que aparece en la tabla.
- La frecuencia interna de trabajo del microcontrolador.
- La cantidad de memoria Flash y SRAM.
- El entorno de programación.
- Por último las características y los periféricos que vienen incluidos en cada una de las placas de desarrollo y evaluación.

Con los datos obtenidos de la tabla, hay que priorizar entre las diversas opciones presentadas, el primer parámetro que se puede utilizar es la tensión de alimentación. Este criterio no es suficientemente discriminatorio ya que todas las placas de desarrollo y evaluación funcionan a la tensión de 3,3V que se quería utilizar como alimentación.

El siguiente parámetro es el precio. Dado que va a ser un sistema de desarrollo para prácticas no se podía elegir una placa de un precio excesivo, ya que se supone que hay que mantener un mínimo de equipos disponibles y un stock lo suficientemente amplio. Las placas que en la tabla aparecen sin sombreado, se han descartado por precio.

Los siguientes criterios no parecen lo suficientemente discriminatorios, la frecuencia de trabajo del microcontrolador, la memoria Flash, la memoria SRAM,…

Tampoco la cantidad de periféricos que la propia placa de desarrollo y evaluación podía incluir, ya que primero, cuanto mayor era el precio, más periféricos tenían. Y segundo y más importante, el sistema de desarrollo aquí implementado debe dotar al conjunto de los periféricos que no tenga la placa de desarrollo y evaluación elegida.

Respecto del entorno de desarrollo, todas soportan CodeWarrior IDE, que se trata del entorno de trabajo seleccionado debido a su actual utilización en las prácticas de SEP.

Por último, el parámetro que ha permitido tomar la decisión ha sido la posibilidad de volcar código desde el programa MATLAB.

Se ha elegido la placa de desarrollo y evaluación FRDM-KL25Z de Freescale porque además, estaba disponible en el área de Tecnología Electrónica.

Fabricante	Referencia modelo	Cod. Farnell	Precio	MCU	Tensión input	Xtal	Flash	SRAM	Entorno programación	Caracteristicas y Periféricos
Freescale Semiconductor	TWR-KM34Z50M, KINETIS M SERIES TOWER MCU MODULE	2366322	131,97 €	KM34Z5128CLL5	2V7 to 3V6	50 MHz	128 KB	16KB	Codewarrior IDE / Eclipse	ARM® Cortex®-M0 core, 100LQFP, interface USB con conector mini-AB, LCD con 160-segmentos, JTAG/SWD (Open SDA), acelerómetro (MMA8491Q), 4 LEDs programables, potenciometro, (RTC), generador senoidal con interface usb para emulación de señales AC, soporte IRDA, NTC
Freescale Semiconductor	MRB-KW019032EU, Modular Reference Board for Kinetis KW0x	2401843	82,17 €	MKW01Z128CHN	1V8 to 3V6	48 MHz	128 KB	16KB	Code Warrior IDE / Eclipse	ARM Cortex-M0+, Salida programable entre –18 dBm hasta +17 dBm en escalones de 1 dB, 2 conectores antena SMA, Alta sensibilidad: down to –120 dBm at 1.2 kbps, puerto JTAG en placa, puerto USB, LEDs e interruptores
Freescale Semiconductor	FRDM-KL25Z Freedom development platform for Kinetis KL25	2191861	11,97 €	KL25Z128VLK4	3V3 to 9V	48 MHz	128KB	16KB	Code Warrior, IAR Embedded Workbench, Keil MDK	ARM® Cortex®-M0 core, 64QFP, Support from Embedded Coder includes automated build and execution, processor-optimized code, zona de contacto capacitivo, acelerometro MMA8451Q, LED RGB, USB o alimentación externa, conectores entradas/salidas accesibles, Infrared comunicaciones, botón Reset , medida y sensor de temperatura, BLDC motor control, compatible con Arduino™ R3, nuevo OpenSDA debug interface
Freescale Semiconductor	FRDM-KE02Z40M DEV BOARD, KE02 SERIES MCU	2406739	13,66 €	MKE02Z64VQH4	2V7 to 5V5	40 MHz	64 KB	4KB	Code Warrior, IAR Embedded Workbench, Keil MDK	ARM® Cortex®-M0 core, 64QFP, zona de contacto capacitivo, acelerometro MMA8451Q, LED RGB, USB o alimentación externa, conectores entradas/salidas accesibles, IrDA transmisor y receptor, Infrared comunicaciones, botón Reset , medida y sensor de temperatura, BLDC motor control, compatible con Arduino™ R3, nuevo OpenSDA debug interface
Freescale Semiconductor	FRDM-KE04Z DEV BOARD, MKE04Z8VFK4 EMI/MOTOR CNTRL	2406740	13,66 €	MKE04Z8VFK4	3V3 to 9V	48 MHz	8KB	1KB	Code Warrior, IAR Embedded Workbench, Keil MDK	ARM® Cortex®-M0 core, 24QFN, zona de contacto capacitivo, acelerometro MMA8451Q, LED RGB, USB o alimentación externa, conectores entradas/salidas accesibles, IrDA transmisor y receptor, Infrared comunicaciones, botón Reset , medida y sensor de temperatura, BLDC motor control, compatible con Arduino™ R3, New OpenSDA debug interface
Freescale Semiconductor	FRDM-KE02Z ARM, KINETIS,KE02Z FREEDOM DEV BOARD	2341582	12,29 €	Kinetis E Series KE02 family MCU	3V3 to 9V	20 MHz	64 KB	4KB	Code Warrior, IAR Embedded Workbench, Keil MDK	ARM® Cortex®-M0 core, 64QFP, zona de contacto capacitivo, 12-bit ADC, 2 comparadores analógicos, SPI, módulo I2C , 3 módulos UART, USB o alimentación externa, conector de entradas/salidas, acelerometro MMA8451Q, sensor y medida de temperatura, botón Reset, LED RGB, Infrared communication, compatible con Arduino™ R3
Freescale Semiconductor	FRDM-KL02Z ARM, KINETIS, KL02Z FREEDOM DEV BOARD	2318317	9,99 €	MKL02Z32VFM4	3V3 to 9V	48MHz	32KB	4KB	Code Warrior, IAR Embedded Workbench, Keil MDK	ARM® Cortex®-M0 core, 32QFP, zona de contacto capacitivo, acelerometro MMA8451Q, LED RGB, batería externa o alimentación externa, conectores entradas/salidas y del MCU accesibles, botón Reset, compatible con Arduino™ R3, OpenSDA debug interface

NXP	OM13053 EVAL, LPCXPRESSO, LPC812, CORTEX M0+	2254492	16,03 €	LPC812	3V3 to 5V	12MHz	16KB	4KB	Eclipse GNU toolchain, Code Warrior IDE	NXP's LPC812 Cortex-M0+, TSSOP20, alimentación externa 3.15V-3.3V, o conector USB y conexión JTAG en placa, UART, conectores entradas/salidas accesibles, RGB LED, Potenciometro
NXP	OM13008, LPCXpresso Board for LPC1227	2103799	19,13 €	LPC1227	3V3 to 5V	45MHz	128KB	8KB	Eclipse GNU toolchain, Code Warrior IDE	ARM Cortex M0, conector JTAG en placa, alimentación externa 3.15V-3.3V, o conector USB y conexión JTAG en placa, UART, conectores entradas/salidas accesibles, RGB LED
NXP	OM13025,598 EVAL, LPC11A14 KSK	2249917	193,71 €	LPC11A14	3V3 to 5V	50MHz	32KB	8KB	Eclipse GNU toolchain, Code Warrior IDE	Cortex-M0, conector JTAG, LCD 8 caracteres, display 7 segmentos, potenciometro, 2 entradas ADC, zumbador, botón Reset, 8 LEDs, interface comunicación, conector USB, RS232/485, LED encendido, JTAG, USB o alimentación externa, acelerometro, área de prototipado
NXP	OM13053,598, Evaluation board for LPC812	2254492	22,69 €	LPC812	3V3 to5V	12MHz	16KB	4KB	Eclipse GNU toolchain, Code Warrior IDE	Cortex-M0+, TSSOP20, 3.15V-3.3V alimentación externa, JTAG en placa, UART, todos los pines del LPC812 accesibles, RGB LED, potenciometro
NXP	OM13035,598, Evaluation Board for LPC1115	2103787	24,30 €	LPC1115	3V3 to 5V	12MHz	64KB	8KB	Eclipse GNU toolchain, Code Warrior IDE	Cortex-M0, LPC1115 en LQFP48, alimentación externa 3.15V-3.3V, o USB, todos los pines accesibles, conector JTAG en placa
NXP	OM13016, NGX LPCXpresso BaseBoard	2103794	47,62 €	LPC11U14	3V3 to 5V	50MHz	32KB	8KB	Eclipse GNU toolchain, Code Warrior IDE	Cortex M0, periféricos configurables mediante jumpers, alimentación externa: DC 7,5V con LED indicador, en placa regulador lineal de +3,3V/500mA y de +5v/500mA, conector USB, todos los pines del micro controlador accesibles, 2 puertos RS232, conector VGA, conector PS/2, conector JTAG, conector SD/MMC, conector RJ45, conector USB tipo B, 256Kb I2C basado EEPROM, amplificador de audio conexón jack, LCD 2 lineas X 16 caracteres con control de iluminación, ISP
NXP	OM13014 LPC11U14, LPCXPRESSO KIT DESARROLLO	1972564	16,01 €	LPC11U14	3V3 to 5V	12MHz	32KB	6KB	Eclipse GNU toolchain, Code Warrior IDE	Cortex-M0, LQFP48, alimentación externa 3.15V-3.3V, o bien desde USB via JTAG probe (LPC-LINK), conector Mini-B USB para LPC11U14 USB interface, todos los pines del micro controlador LPC11U14 accesibles en conector de expansión, Embedded JTAG (LPC-LINK), LED, Eclipse-based IDE, JTAG Debugger en placa
NXP	OM 11086,598 Keil LPC1114 Evaluation Board	2103802	193,65 €	LPC11U14	3V3 to 5V	50MHz	32KB	8KB	Eclipse GNU toolchain, Code Warrior IDE	ARM Cortex-M0, 48-pin LQFP, CAN 2.0 Interface, USB 2.0 Device, Serial Interface, Potenciometro para entrada, 42 GPIO accesibles, 8 LEDs, 4 pulsadores (2 GPIO, ISP, & reset), Power via USB connector, Debug Interface Connectors
NXP	OM1308, Keil LPC11U14 Evaluation Board	1972565	139,76 €	LPC11U14	3V3 to 5V	50MHz	32KB	8KB	Eclipse GNU toolchain, Code Warrior IDE	ARM Cortex-M0, 48-pin LQFP, CAN 2.0 Interface, USB 2.0 Device, Serial Interface, Potenciometro para entrada, 42 GPIO accesibles, 8 LEDs, 4 pulsadores (2 GPIO, ISP, & reset), Power via USB connector, Debug Interface Connectors

ATMEL	ATSAMD20-XPRO, SAM D20 Xplaines Pro Evaluation Kit	2334311	35,14 €	SAMD20J18	3V3 to 5V	48MHz	256KB	8KB	ATMEL STUDIO 6, IAR Embedded Workbench	ARM Cottex M0+, botón Reset, botón adicional, LED amarillo, LED verde, 32.768kHz crystal, USB interface, Embedded Debugger, Auto-ID para identificación de tarjeta en Atmel Studio 6.1, Data Gateway Interface: SPI, I²C, 4 GPIOs, Virtual COM port (CDC) , USB powered
ATMEL	ATSAMD21-XPRO, SAM D21 Xplaines Pro Evaluation Kit	2407175	39,70 €	SAMD21J18A	3V3 to 5V	48MHz	256KB	8KB	ATMEL STUDIO 6, IAR Embedded Workbench	ARM Cottex M0+, botón Reset, botón adicional, LED amarillo, LED verde, 32.768kHz crystal, USB interface, Embedded Debugger, Auto-ID para identificación de tarjeta en Atmel Studio 6.1, Data Gateway Interface: SPI, I²C, 4 GPIOs, Virtual COM port (CDC) , USB powered
CYPRESS	CY8CKIT-040, PSoC 4000 Pioneer Development Kit	2428290	31,61 €	PSoC 4 (4000 family)	1,8V to 5V	16MHz	16KB	2KB	PSoC Creator IDE	ARM Cottex M0, PSoC 5LP, fuente de alimentación interna, Programming interfaces, conectores compatibles Arduino, conector PSoC 5LP GPIO, LEDs, botón Reset, Cypress ferroelectric RAM (F-RAM)
CYPRESS	CY8CKIT-042, PSoC 4 Pioneer Kit	2311054	21,96 €	PSoC® CY8C4245AXI	1,8V to 5V	48MHz	32KB	4KB	PSoC Creator IDE	ARM® Cortex™-M0, 16-bit Timer/PWM, Low Power Comparator, 12-bit ADC, Programmable Analog Block, Programmable Digital Block, sistema I/O versatil, tensión de alimentación 1.71 – 5.5 V, compatible Arduino™, conector compatible Digilent® Pmod™, conector GPIO Cypress PSoC 5LP
STM	NUCLEO-F030R8, STM32 Nucleo Board for STM32F030R8T6 MCU	2394225	9,40 €	STM32	3V3 to 5V	48MHz	64KB	8KB	Code Warrior, IAR Embedded Workbench, Keil MDK	ARM Cottex M0, LQFP64, compatible Arduino Uno Revision 3, pines STM32 I/O totalmente accesibles, mbed-enabled, en placa ST-LINK/V2-1 debugger/programmer con SWD connector, alimentación flexible en placa; USB VBUS, External VIN (7V<VIN<12V), External 5V y +3.3V, 3 LEDs ,USB communication, LED indicación en uso, power LED, 2 pulsadores: USER and RESET, USB, Integrated Development Environments (IDEs) including IAR, Keil, GCC-based IDEs
STM	STM320518-SK/IAR, IAR Starter Kit for STM32F0 Series (STM32F051R8)	2318892	200,67 €	STM32F051R8	3V3 to 5V	48MHz	64KB	8KB	Code Warrior, IAR Embedded Workbench, Keil MDK	Cortex-M0, LEDs de estado, audio (headphone jack), 2 pulsadores, 1 botón Reset, 6 pulsadorres capacitivos, LCD, Debug interface JTAG/SWD, Communication interface; 1 UART, conectores HDMI-CEC, soluciones multiples de alimentación: JTAG/SWD, USB, externa power jack, potenciometro, acelerometro de 3 ejes
STM	STM320518-EVAL, STM320518-EVAL Evaluation Board for SMT32F0 series (STM32F051R8)	2134412	245,12 €	STM32F051R8	3V3 to 5V	48MHz	64KB	8KB	Code Warrior, IAR Embedded Workbench, Keil MDK	Cortex-M0, 5 V diferentes opciones de alimentación, jack de alimentación, conector ST-LINK/V2 USB connector, altavoz y microfono Audio conectado a STM32F051R8T6, 2Gbyte de MicroSD card con SPI interface, sensor de temperatura, RF EEPROM accesible I2C interface serie, RS232 and RS485, IrDA transceiver, IR LED and IR receiver, SWD debug, LCD conectado a STM32F051R8T6 mediante SPI interface, Joystick control en 4 direcciones y selector, Reset, pulsadores, 4 LEDs, tensión del MCU: fijo de 3.3 V, o ajustable desde 2 hasta 3.6 V, zona de contacto capacitiva, LDR, potenciometro, 2conectores HDMI, conector para control de motor

EMBEST	COOKIE NUMICRO EVB	2136556	29,36 €	M0516LBN	3V3 to 5V	50MHz	64KB	8KB	Code Warrior, IAR Embedded Workbench, Keil MDK	ARM Cortex MCUs compatibles (M0/M3/M4), totalmente compatible con Arduino™ , alimentación entre 3.3V and 5V, seleccionado mediante jumper, en la placa CoLinkEx para USB-JTAG/SW debugging, 2 UARTs, 2 SPIs, 1 I2C, 14 entradas/salidas digitales (de las cuales 6 pueden ser usadas como salidas PWM, 6 entradas analógicas, jack de alimentación, ICSP
KEIL	MCB11C14, Evaluation Board for LPC11C14	2309957	127,50 €	LPC11C14FB48	3V3 to 5V	50MHz	32KB	8KB	Code Warrior, IAR Embedded Workbench, Keil MDK	ARM Cortex-M0, CAN 2.0 Interface, Serial Interface, potenciometro para entrada ADC, 42 GPIO, 8 LEDs, 4 pulsadores (2 GPIO, ISP y reset), alimentación via conector USB, Debug Interface Connectors
KEIL	MCB11C14, Evaluation Board for LPC11C14	2309958	331,50 €	LPC11C14FB48	3V3 to 5V	50MHz	32KB	8KB	Code Warrior, IAR Embedded Workbench, Keil MDK	ARM Cortex-M0, CAN 2.0 Interface, Serial Interface, potenciometro para entrada ADC, 42 GPIO, 8 LEDs, 4 pulsadores (2 GPIO, ISP y reset), alimentación via conector USB, Debug Interface Connectors
KEIL	MCB11U10UME, Evaluation Board for LPC11U14	2309962	188,70 €	LPC11U14/201	3V3 to 5V	50MHz	32KB	6KB	Code Warrior, IAR Embedded Workbench, Keil MDK	NXP LPC11U14/201 Cortex-M0, USB 2.0 Device, Serial Interface, potenciometro para entrada ADC, 42 GPIO, 8 LEDs, 4 pulsadores (2 GPIO, ISP y reset), alimentación via conector USB connector, Debug Interface Connectors
KEIL	MCB1200U, Evaluation Board for NXP LPC122x family	2309964	341,70 €	LPC1227FBD64	3V3 to 5V	33MHz	128KB	8KB	Code Warrior, IAR Embedded Workbench, Keil MDK	ARM Cortex-M0, LQFP, Serial Interface, potenciometro para entrada ADC, 55 GPIO, 8 LEDs, 4 botones (2 GPIO, ISP y reset), alimentación via conector USB, Debug Interface

4. Placa de Desarrollo y Evaluación Freescale FRDM-KL25Z

La plataforma de desarrollo Freescale Freedom es un conjunto de herramientas de hardware y software para la evaluación y desarrollo. Es ideal para prototipado rápido de aplicaciones basadas en microcontroladores.

La tarjeta Freescale Freedom KL25Z, modelo FRDM-KL25Z, es un diseño rentable y capaz con un microcontrolador Kinetis serie L, uno de los primeros microcontroladores de la industria construido sobre el núcleo ARM ® Cortex ™ M0+.

FRDM-KL25Z puede utilizarse para evaluar dispositivos de la serie Kinetis L. Cuenta con un microcontrolador KL25Z128VLK, un dispositivo con una máxima frecuencia de operación de 48MHz, 128KB de memoria flash, un controlador USB, periféricos analógicos y digitales. La placa FRDM-KL25Z es compatible con el diseño de Arduino ™ R3, proporcionando una amplia gama de expansión. En la tarjeta se incluyen un LED RGB, un acelerómetro digital de 3 ejes y una zona táctil capacitiva.

FRDM-KL25Z es la primera plataforma de hardware de Freescale con OpenSDA (sistema abierto estándar serie integrado y adaptador debugger). Este circuito ofrece varias opciones para comunicaciones series, programación flash y control de ejecución de depuración.

Existe una amplia lista de documentos de referencia de la placa de Freescale Freedom FRDM-KL25Z. En ***FRDM-KL25Z User's Manual*** se dispone de un manual de usuario ampliado.

Todos los documentos están disponibles en **http://www.freescale.com/FRDM-KL25Z**.

Las características de la tarjeta FRDM-KL25Z incluyen:

• Microcontrolador Kinetis serie L, MKL25Z128VLK4 en un encapsulado LQFP 80.

• Zona táctil capacitiva.

• Acelerómetro digital MMA8451Q.

• LED RGB.

• Varias opciones de alimentación: USB, batería de botón, fuente externa.

• Fácil acceso al conector entradas/salidas, IO MCU compatible Arduino ™ R3.

• Interface programable OpenSDA con múltiples aplicaciones disponibles, incluyendo:

- Interfaz de programación flash del dispositivo de almacenamiento masivo.
- Interfaz de depuración PE, proporciona control de ejecución y compatibilidad con herramientas IDE.
- Interfaz de DAP CMSIS: nuevo estándar para la interfaz de depuración.

La figura 7 muestra un diagrama de bloques de la placa Freescale FRDM-KL25Z.

Los principales componentes y su ubicación en la placa se representan en la figura 8.

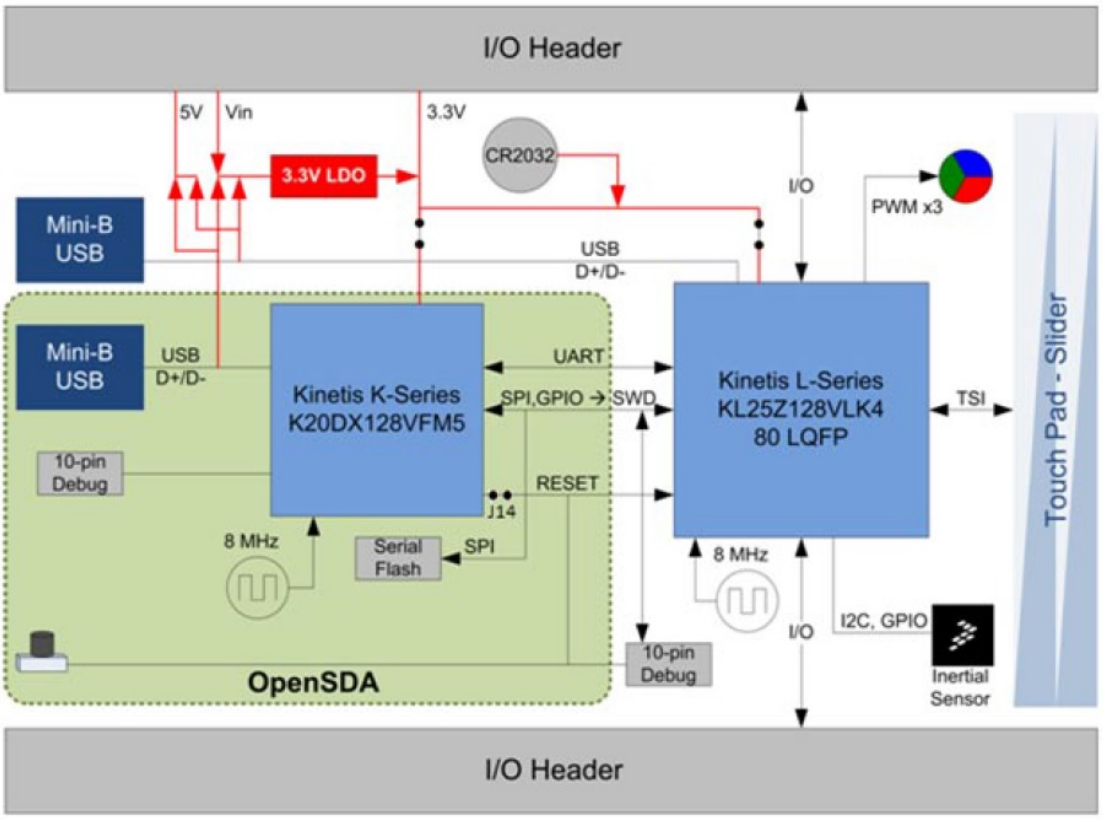


Figura 7. Diagrama de bloques de la placa FRDM-KL25Z de Freescale.

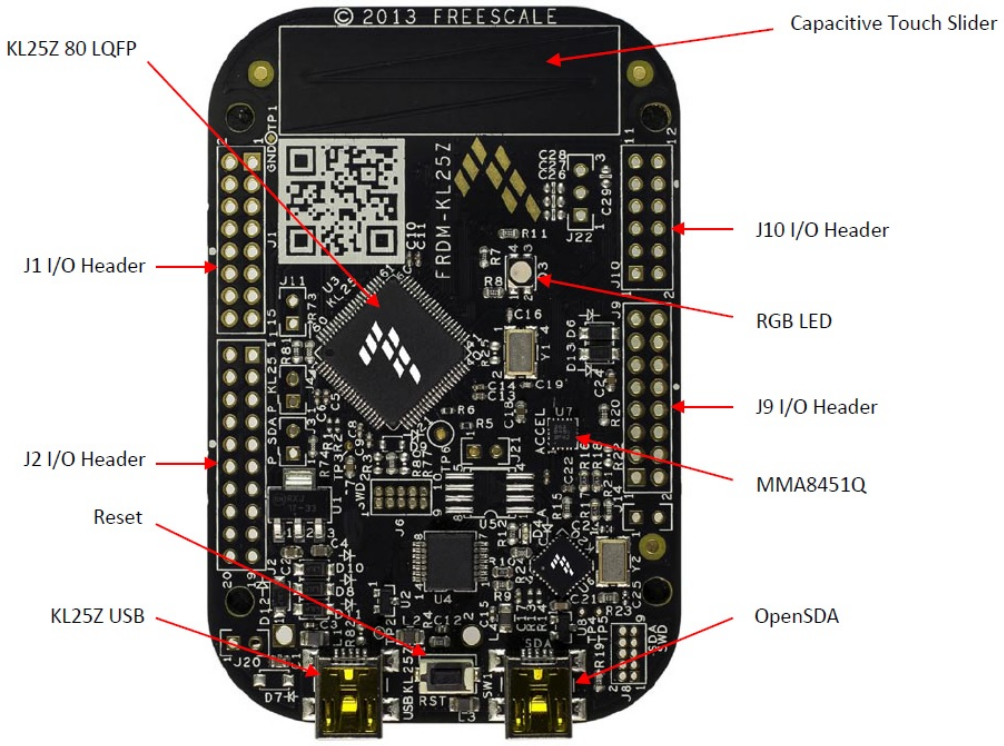


Figura 8. Principales componentes y su ubicación en la placa FRDM-KL25Z de Freescale.

4.1 Descripción del hardware

- Fuente de alimentación

Hay múltiples opciones de alimentación en la placa de desarrollo FRDM-KL25Z. Puede ser alimentada desde cualquiera de los conectores USB, el pin VIN del conector de IO, una pila de botón incluida, entre 1.71-3.6V de la fuente de los 3.3V pin del conector de IO. Las alimentaciones USB y VIN están reguladas en la placa mediante un regulador lineal 3.3V. Las otras dos fuentes no están reguladas.

- OpenSDA

OpenSDA es una serie de estándares abiertos y debugger. Es un sistema de comunicaciones serie entre un microcontrolador y el conector USB. OpenSDA cuenta con un dispositivo de almacenamiento masivo, que proporciona un mecanismo rápido y fácil para cargar distintas aplicaciones OpenSDA tales como programadores de flash. El circuito de hardware se basa en un microcontrolador de Freescale Kinetis K20 con 128 KB de flash incorporado y un controlador USB integrado. Se puede consultar el documento ***OpenSDA User's Guide*** para cualquier aclaración.

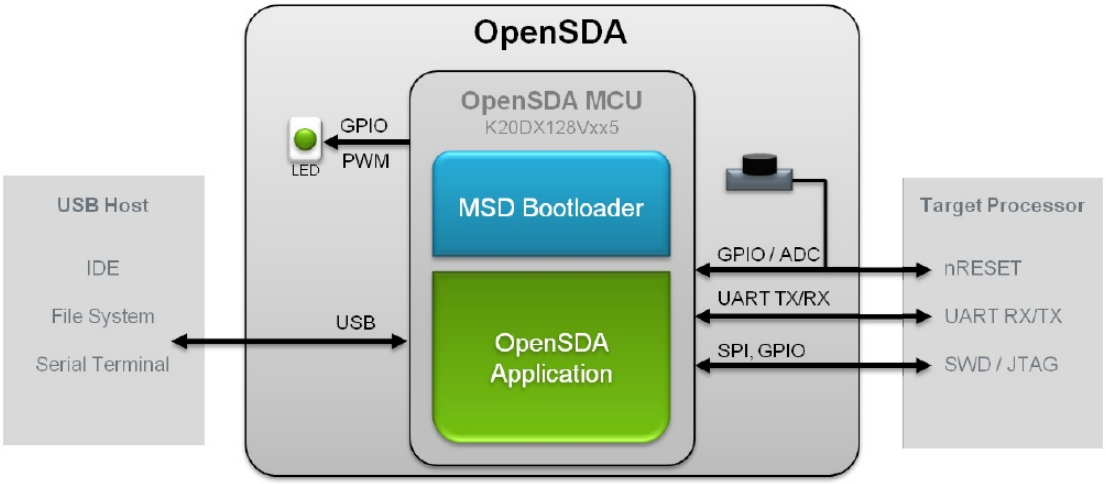


Figura 9. Diagrama de bloques OpenSDA.

OpenSDA está gestionado por un microcontrolador K20 de Kinetis construido sobre el núcleo ARM ® Cortex ™-M4. El circuito del OpenSDA incluye un LED de control (D4) y un pulsador (SW1). El pulsador es también la señal de Reset del microcontrolador KL25Z MCU. También puede utilizarse para colocar el circuito OpenSDA en modo Bootloader.

- Zona táctil capacitiva

Dos señales de entrada de sentido táctil (TSI), TSI0CH9 y TSI0CH10, están conectadas a unos electrodos capacitivos, configurados como un control táctil deslizante. Freescale proporciona una biblioteca de software para implementar la zona táctil capacitiva.

- Acelerómetro de 3 ejes

Se dispone en la placa de un acelerómetro digital Freescale MMA8451Q de baja potencia, tres ejes, está interconectado a través de un bus I2C y al conector de GPIO.

- LED RGB

Tres señales PWM están conectadas a un LED rojo, verde, azul.

- Conectores entradas y salidas

El microcontrolador KL25Z128VLK4 está disponible en un encapsulado 80-pin LQFP. Unos pines se utilizan internamente en la placa de desarrollo, pero muchos están conectados directamente a una de los cuatro conectores de entradas/salidas IO.

Los pines en el microcontrolador KL25Z llevan el nombre de su función de entrada-salida. Por ejemplo, el pin 1 en el puerto A se denomina PTA1. En su caso, los nombres de los contactos del conector de IO reciben el mismo nombre que el pin de KL25Z conectado a él.

Los conectores se denominan, J1 el primero de 16 pines a la izquierda, J2 el conector inferior de 20 pines, J9 el de 16 pines inferior derecha y por último J10 el de 12 pines.

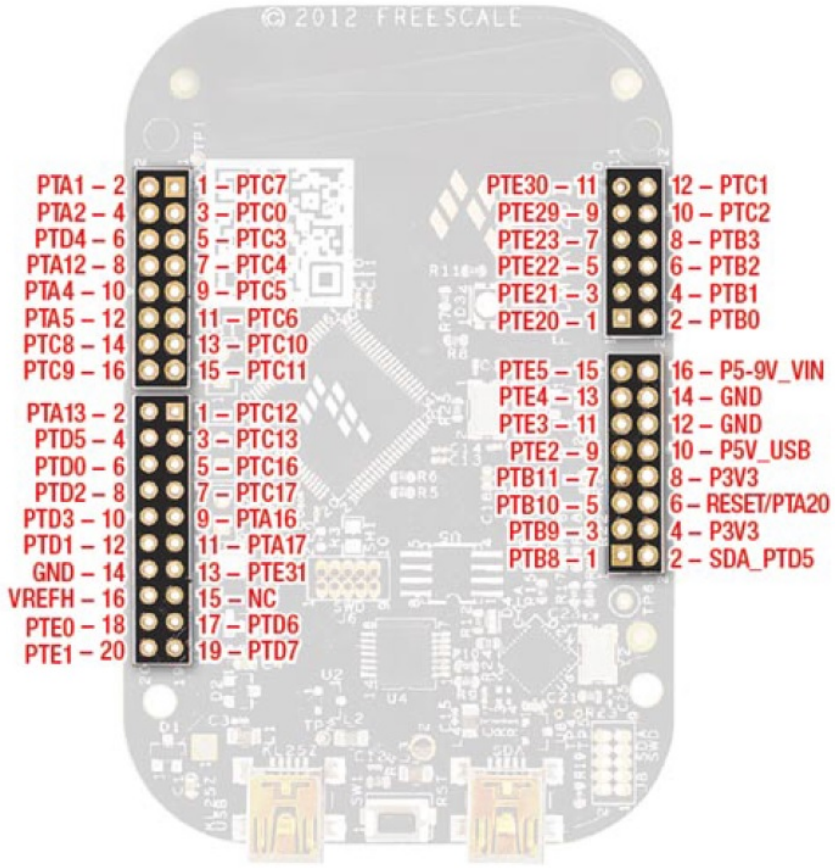


Figura 10. Conectores IO de la placa FRDM-KL25Z.

- Conversor analógico digital

Internamente el microcontrolador MKL25Z128VLK4 dispone de un conversor analógico digital. Para su correcto funcionamiento, el voltaje de referencia alto VREFH se puede seleccionar su conexionado, o bien al pin P3V3 del KL25Z, o bien a la salida externa AREF. El voltaje de referencia bajo se ha conectado a GND.

- Compatibilidad Arduino

Los conectores de IO en el FRDM-KL25Z se han dispuesto para permitir la compatibilidad con placas de periféricos del microcontrolador Arduino. Las filas exteriores de pines (los pines pares) en los conectores comparten el mismo espacio mecánico y colocación con los conectores de IO en el Arduino revisión 3 (R3) estándar.

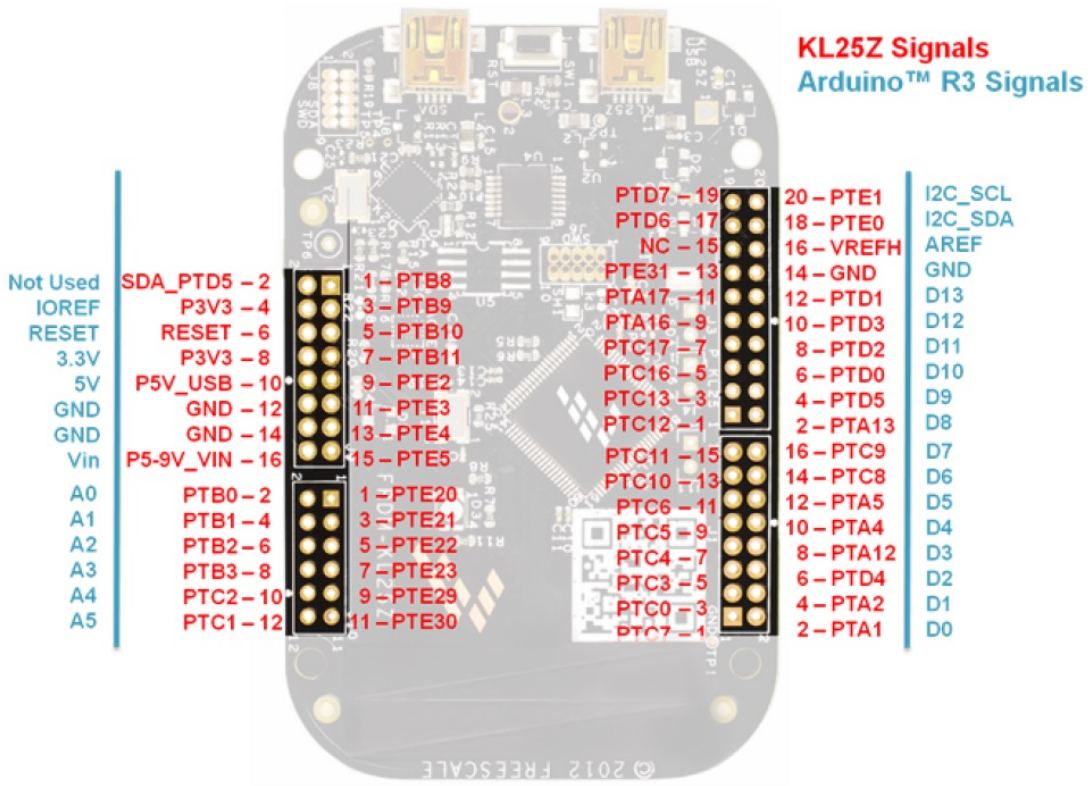


Figura 11. Conectores IO de la placa FRDM-KL25Z compatibles con Arduino.

4.2 Instalación Freescale FRDM-KL25Z

El procedimiento de instalación de la Plataforma Freescale FRDM-KL25Z está descrito en ***FRDM-KL25Z Quick Start Guide***, un resumen de este documento es el siguiente:

Al conectarse el cable USB en el conector mini USB OpenSDA, se enciende un LED verde. En el ordenador, la placa es un dispositivo de almacenaje mass-storage device, un volumen llamado FRDM-KL25Z.

Tiene preinstalada una demo con la activación del LED RGB de la placa, el color del LED cambia mediante los datos que le proporciona el acelerómetro interno. Deslizando el dedo por la zona capacitiva de la placa podemos modificar el brillo del LED.

Una vez instalado CodeWarrior Development Studio v10.6, conectar el cable USB y desde propiedades de Equipo abrir el Administrador de dispositivos para comprobar que la placa

se ha instalado correctamente. Para ello, en Jungo existe un dispositivo "PEMicro OpenSDA Debug Drive" y en los puertos (COM y LPT) aparece conectado un "OpenSDA-CDC Serial Port".

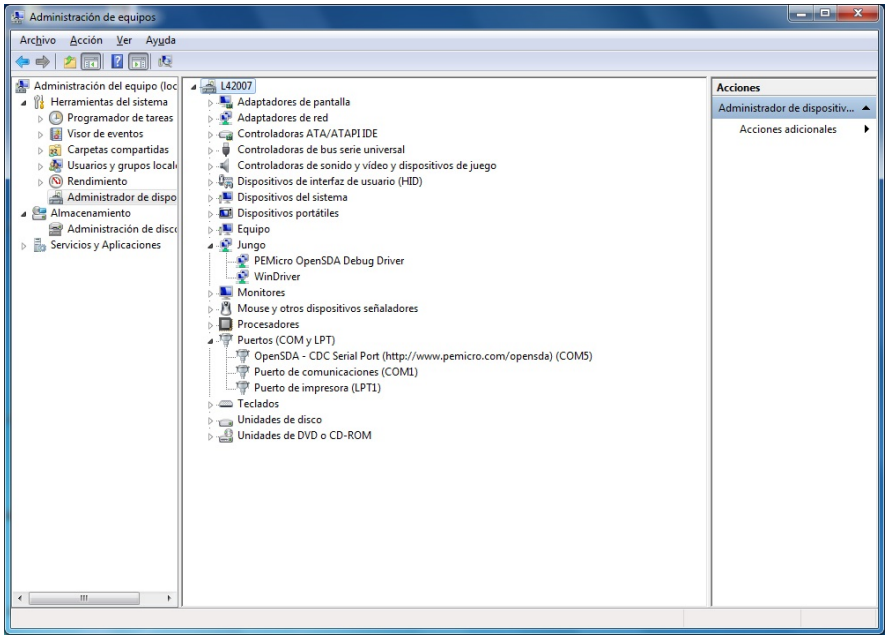


Figura 12. Instalación Freescale Freedom FRDM-KL25Z, administrador de dispositivos.

Si por cualquier problema no se ha instalado correctamente, descargar e instalar los drives "**P&E OpenSDA USB drivers**" disponibles en la página web http://www.pemicro.com/openSDA.

Modo OpenSDA Bootloader

Para entrar en modo OpenSDA Bootloader, se realiza la siguiente operación:

Desconectar el cable USB del equipo, presionar y mantener pulsado el botón de Reset (SW1), conectar el cable USB entre el ordenador y el conector mini USB OpenSDA de la tarjeta, soltar el botón de Reset.

El LED de la placa parpadea cada 0,5 segundos, el dispositivo de almacenaje ahora tiene el nombre de BOOTLOADER.

Figura 13. Dispositivo de almacenaje Bootloader.

Si existe un problema al intentar cargar una aplicación en la tarjeta, puede aparecer el mensaje, no encuentra el dispositivo OpenSDA, si esto ocurre verificar la versión Bootloader. Para ello, en modo Bootloader se ejecuta el fichero SDA_INFO.HTML.

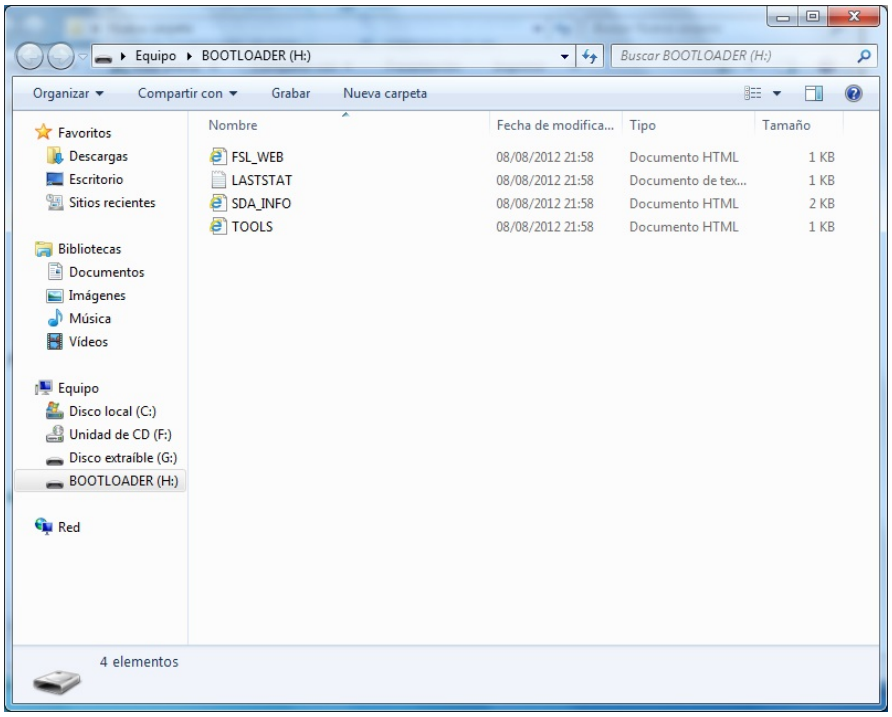


Figura 14. Fichero SDA_INFO en modo Bootloader.

Se abre una ventana del navegador donde se puede verificar la versión de Bootloader, la versión debe ser 1.11 o superior, si no es el caso descargar de http://www.pemicro.com/openSDA.

Figura 15. Comprobación versión Bootloader en plataforma Freescale Freedom FRDM-KL25Z.

Cargar una aplicación OpenSDA

Localizar las aplicaciones OpenSDA en la carpeta ***FRDM-KL25Z Quick Start Package***, que se encuentra en **http://www.freescale.com/FRDM-KL25Z**, copiar y pegar la aplicación en el drive Bootloader, desconectar el cable para salir del modo Bootloader y conectarlo de nuevo.

La nueva aplicación OpenSDA está cargada en la plataforma FRDM-KL25Z. Si es necesario instalar una nueva versión del MSD Flash Programmer, utilizar el mismo procedimiento que para cargar cualquier aplicación OpenSDA.

Se encuentra disponible una ampliación de este documento en el Anexo A1, Documento Ayuda.

4.3 Microcontrolador Kinetis MKL25Z128VLK4

El microcontrolador MKL25Z128VLK4 pertenece a la serie L de Kinetis, es una serie de ultra baja potencia, basada en microcontroladores Cortex-M0+. La serie incluye 5 familias de microcontroladores que ofrecen una amplia gama de productos.

Las series Kinetis L constan de periféricos comunes y mantienen entre si la estructura de pines lo que permite a los usuarios potenciales migrar de una familia a otra. Esta escalabilidad permite maximizar la reutilización de hardware y software.

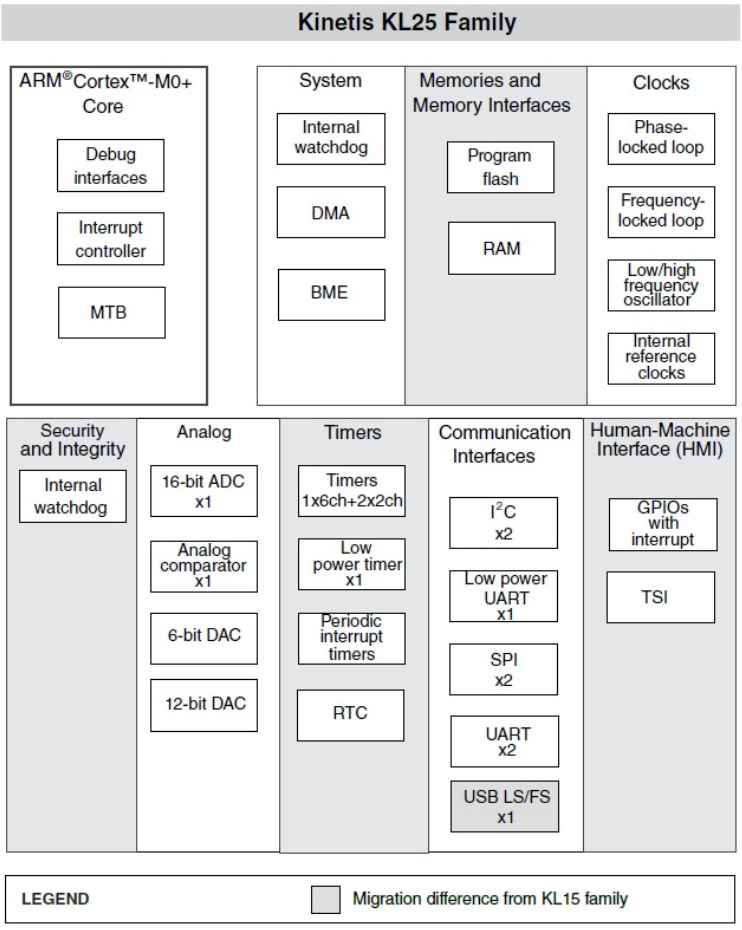


Figura 16. Diagrama de bloques funcional del microcontrolador MKL25Z128VLK4.

Las familias de la serie L Kinetis tienen diversas características comunes:

• Procesador ARM Cortex-Mo+ a 48MHz.
• Convertidores analógicos digitales de 12/16 bits, convertidores digital analógico de 12 bits.
• Comparadores analógicos de alta velocidad.
• Baja potencia.
• Timers, 6 clock/PWM(TPM), reloj interrupciones periódicas, reloj de tiempo real.
• Temporizadores de gran alcance para una amplia gama de aplicaciones.
• Interfaces de comunicación serie un módulo UART de baja potencia, 2 módulos UART, 2 módulos SPI de 8 bits, 2 módulos I2C.
• Fuente de alimentación simple: 1.71V - 3.6V con múltiples modos de alimentación.

Todos los microcontroladores de la familia Kinetis L combinan las últimas innovaciones con una amplia gama de conectividad, comunicación y periféricos. La familia KL0x es el punto de partida a la serie L de Kinetis y compatible pin a pin con la familia de S08PT de 8 bits. Las familias KL1x, KL2x, KL3x, y KL4x son compatibles entre sí y sus equivalente ARM Cortex-M4 de la serie K de Kinetis.

• Respecto del núcleo ARM Cortex-M0+ y la arquitectura del microcontrolador MKL25Z128VLK4:

- Núcleo ARM Cortex-M0+, frecuencia de CPU de hasta 48 MHz, frecuencia del bus 24MHz. Está basado en la arquitectura ARMv6. Compatible 100% con ARM Cortex-M0 y con ARM Cortex-M3 y M4, lo que permite reutilizar los compiladores y herramientas de depuración.

- Acceso a puertos de IO hasta un 50% más rápido que el estándar, mejora el tiempo de reacción ante sucesos externos.

- Reducido número de ciclos por instrucción (CPI), permitiendo la reducción de consumo de energía, densidad de código frente a microcontroladores de 8 y 16 bits.

- Optimizando el acceso a la memoria de programa se reduce accesos en ciclos alternos.

- Arquitectura simplificada, 56 instrucciones y 17 registros permiten una fácil programación y un uso eficiente de los datos en memoria de 8, 16 y 32 bits.

- Micro Trace Buffer, identificación y rápida corrección de fallos, BME, motor de manipulación de bits, reduce el tamaño del código.

- Hasta 4 canales DMA para periféricos y servicio de memoria con la mínima intervención de la CPU.

• Ultra baja potencia:

- Extrema eficiencia dinámica, ahorro energético 50% respecto núcleos de 8, 16 bits.

- Modos múltiples de baja potencia, incluyendo la nueva operación de opción de sincronización que reduce la potencia dinámica desconectando los relojes del bus y los sistemas de más bajo procesamiento del núcleo.

• Confiabilidad, seguridad y protección:

- Watchdog interno con reloj independiente.

• Memoria:

Memoria Flash de 128KB, SRAM 16KB, memoria caché de 64 B para optimizar el ancho de banda del bus y de la memoria flash.

System 32-bit Address Range	Destination Slave
0x0000_0000 - 0x07FF_FFFF	Program flash and read-only data (Includes exception vectors in first 196 bytes)
0x0800_0000 - 0x1FFF_EFFF	Reserved
0x1FFF_F000 - 0x1FFF_FFFF	SRAM_L: Lower SRAM
0x2000_0000 - 0x2000_2FFF	SRAM_U: Upper SRAM
0x2000_3000 - 0x3FFF_FFFF	Reserved
0x4000_0000 - 0x4007_FFFF	AIPS Peripherals
0x4008_0000 - 0x400F_EFFF	Reserved
0x400F_F000 - 0x400F_FFFF	General purpose input/output (GPIO)
0x4010_0000 - 0x43FF_FFFF	Reserved
0x4400_0000 - 0x5FFF_FFFF	Bit Manipulation Engine (BME) access to AIPS Peripherals for slots 0-127
0x6000_0000 - 0xDFFF_FFFF	Reserved
0xE000_0000 - 0xE00F_FFFF	Private Peripherals
0xE010_0000 - 0xEFFF_FFFF	Reserved
0xF000_0000 - 0xF000_0FFF	Micro Trace Buffer (MTB) registers
0xF000_1000 - 0xF000_1FFF	MTB Data Watchpoint and Trace (MTBDWT) registers
0xF000_2000 - 0xF000_2FFF	ROM table
0xF000_3000 - 0xF000_3FFF	Miscellaneous Control Module (MCM)
0xF000_4000 - 0xF7FF_FFFF	Reserved
0xF800_0000 - 0xFFFF_FFFF	IOPORT: GPIO (single cycle)

Figura 17. Mapa memoria del microcontrolador MKL25Z128VLK4.

• Sincronización y Control:

- Módulos de temporizador de propósito general, módulos PWM y funciones de control de motor.

- Temporizador de interrupciones para RTOS, módulos de conversión y temporizador de ADC.

• System:

- Puertos GPIO con funcionalidad de interrupción. Temperatura ambiente de funcionamiento desde-40 ° C a 105 ° C.

- Rango de voltaje de alimentación de 1.71 V a 3,6 V con flash programable hasta 1.71 V.

Se puede encontrar información ampliada en los siguientes documentos pdf, disponibles en http://www.freescale.com/FRDM-KL25Z:

- KL25P80M48SF0RM.pdf, ***KL25 Sub-Family Reference Manual***.
- KL25P80M48SF0.pdf, ***Kinetis KL25 Sub-Family***.
- KL25P80M48SF0 datasheet.pdf, ***KL25 Sub-Family Data Sheet***.
- ***FRDM-KL25Z Pinouts*** (Rev 1.0).pdf, Pines del microcontolador MKL25Z128VLK4.
- KLQRUG.pdf, ***Kinetis L Peripheral Module Quick Reference***.

5. Sistema de Desarrollo

Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes, es un proyecto realizado con el fin de obtener un sistema de desarrollo versátil y universal, que permita la realización de las prácticas de (SEP).

Este sistema de desarrollo consta de una placa de periféricos, para su diseño se ha partido de los periféricos utilizados típicamente en prácticas y se han añadido diversos dispositivos. Se han integrado en una placa, a continuación se explica y justifica la elección de dichos periféricos y su exposición.

El sistema de desarrollo consiste en la placa FRDM-KL25Z conectada sobre la placa de periféricos realizada. Se trata de que cada alumno cuando utilice el sistema de desarrollo pueda elegir el dispositivo y pueda seleccionar en que puerto conecta cada uno de los dispositivos que utilice durante la sesión de prácticas. Para ello las conexiones que se podían haber dejado fijas en la placa, se conectan realmente mediante cables preparados a tal efecto.

5.1 Estructura del Sistema de Desarrollo

En primer lugar se justifica la elección de la alimentación de la placa y se incluyen los siguientes periféricos:

- LEDS, pulsadores y conmutadores.
- Display 7 segmentos, LCD y Teclado
- Memoria EEPROM serie.
- Conversor analógico digital serie de 12 bits, conversor digital analógico serie.
- Potenciómetro digital y analógico, acelerómetro analógico.
- Sensores, temperatura, presión y humedad.

5.1.1 Alimentación de la placa

La tensión de alimentación es de 3,3V, tensión nominal de alimentación del sistema de desarrollo y evaluación Freescale FRDM-KL25Z. Se incluyen dos posibles alimentaciones, controladas por el Jumper J1. En la parte izquierda de la figura 18, un conector PCB de tornillo de paso 5,08mm que permite introducir directamente la fuente de alimentación del laboratorio. En el otro lado un conector Jack de alimentación donde se puede conectar un alimentador universal.

Se ha incluido el regulador de tensión de STMicroelectronics LD1117S33CTR. Está disponible en RS-Amidata y su referencia es 686-9388, se trata de un regulador de tensión lineal de caída mínima. Es capaz de suministrar una corriente de 1,3A, aunque se supone que no van a estar funcionando varios periféricos a la vez, se ha elegido de una corriente elevada para tener un cierto margen de seguridad.

Dado que la alimentación se produce mediante un alimentador universal no se han introducido diodos de rectificación. Solamente se incluye en la entrada del regulador de tensión un condensador electrolítico de filtrado de 1000uF de montaje superficial disponible en RS-Amidata con referencia 715-1196, un condensador electrolítico de filtrado a la salida del

regulador de 10uF montaje superficial disponible del mismo modo en RS-Amidata con referencia 739-4450, y un condensador de desacoplo de 100nF cerámico multicapa de encapsulado C1206 también disponible en RS-Amidata con referencia 264-4179.

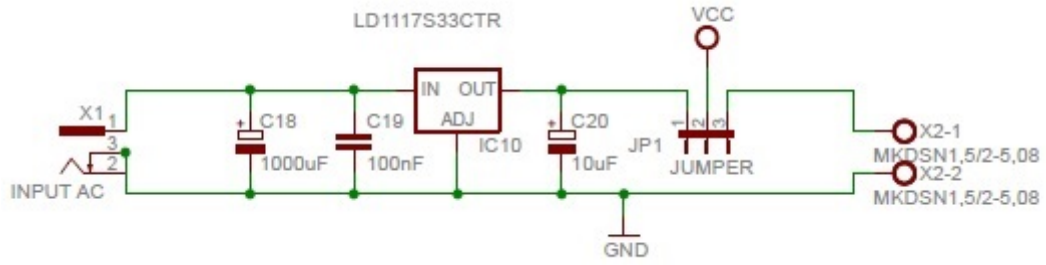


Figura 18. Esquema de la fuente de alimentación del sistema de desarrollo.

5.1.2 Sistema desarrollo y evaluación Freescale FRDM-KL25Z

El sistema de desarrollo y evaluación Freescale FRDM-KL25Z se conecta mediante los conectores J1, J2, J9 y J10. Se trata de conectores hembra del tipo PCB E-TEC Recto 72 pines 2 fila paso 2.54mm montaje Orificio Pasante, disponibles en RS-Amidata con la referencia 549-4949.

En el sistema de desarrollo se han incluido además otros conectores llamados TEST_J1, TEST_J2, TEST_J9 y TEST_J10 que permiten realizar medidas con el osciloscopio en cualquiera de los pines accesibles del sistema de desarrollo y evaluación FRDM-KL25Z. Estos conectores son machos del tipo 3M Recto 40 pines 2 filas paso 2.54mm terminación Soldada, disponible en RS-Amidata con el código 827-7772.

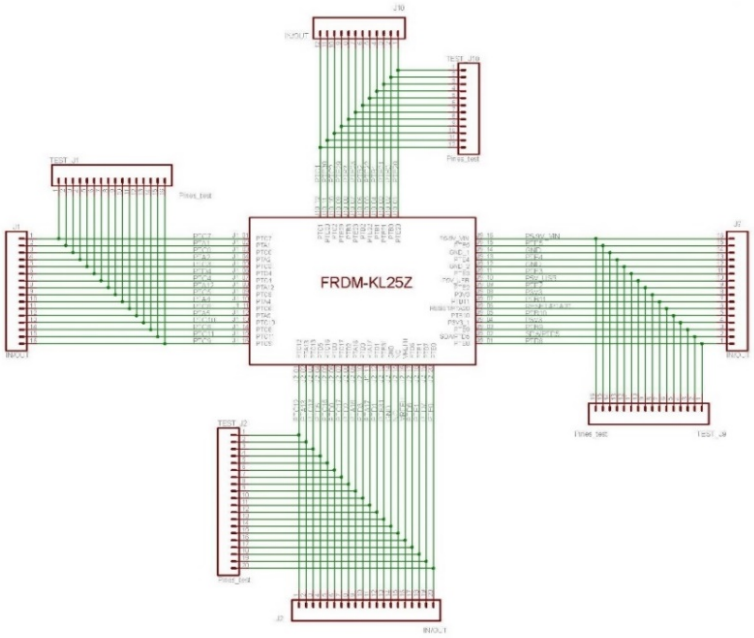


Figura 19. Conexión del sistema de desarrollo y evaluación FRDM-KL25Z en la placa.

5.1.3 LEDS

Se incluyen en la placa 8 leds rojos de 5mm Ø, se estudió la sustitución de los leds de agujero pasante por unos de montaje superficial, pero dado que no tenemos problema de espacio en la placa y que no se producía una mejora sustancial con su cambio se decidió utilizar los leds disponibles.

Del mismo modo, las resistencias utilizadas son de montaje superficial R1206.

$$R_{77} = \frac{V - V_Z}{I} = \frac{3{,}3V - 1{,}8V}{7mA} = 220\Omega$$

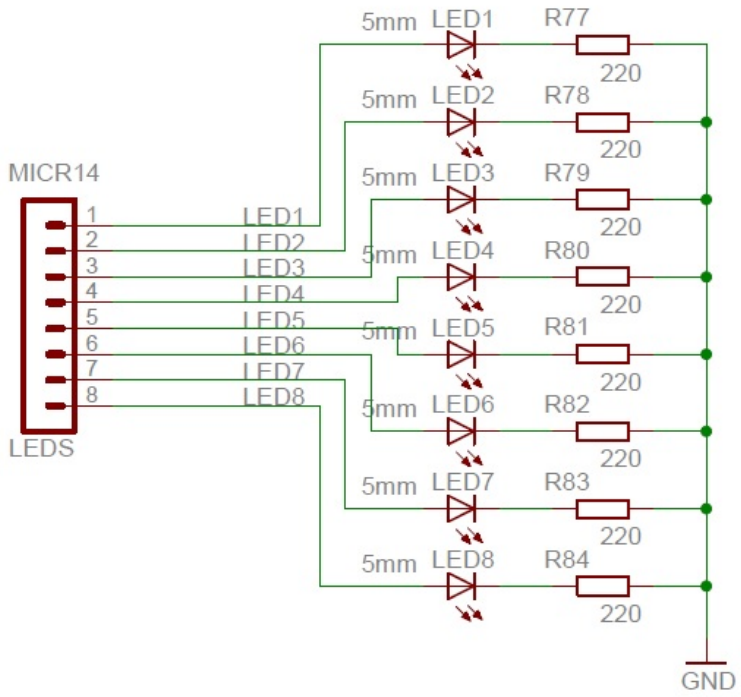


Figura 20. Conexión de los LEDS en el sistema de desarrollo.

5.1.4 Pulsadores

En la placa se han instalado 8 pulsadores subminiatura tipo tecla, disponibles en Diotronic código DTS62N de 6,2mm X 6,2mm.

Se han dispuesto sendas resistencias de protección de la entrada IO del sistema de desarrollo y evaluación FRDM-KL25Z de 100Ω. Del mismo modo, resistencias de 10KΩ de pull-up en cada una de las entradas. Ambas resistencias son de montaje superficial de encapsulado R1206.

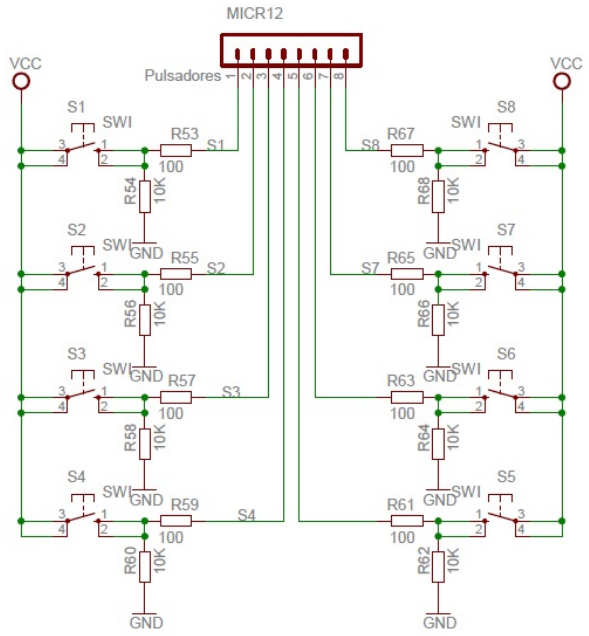


Figura 21. Esquema eléctrico de los pulsadores en el sistema de desarrollo.

5.1.5 Conmutadores

Para completar la dotación básica del sistema de desarrollo se incluyen 8 conmutadores de paso 2,54mm del fabricante EOZ, Interruptores de Actuador Deslizante de Conmutación SIL SPST PCB que se encuentran en RS-Amidata código 204-7865.

Se han incluido resistencias de montaje superficial de 100Ω, para proteger la entrada IO del sistema de desarrollo y evaluación FRDM-KL25Z. Son de encapsulado R1206.

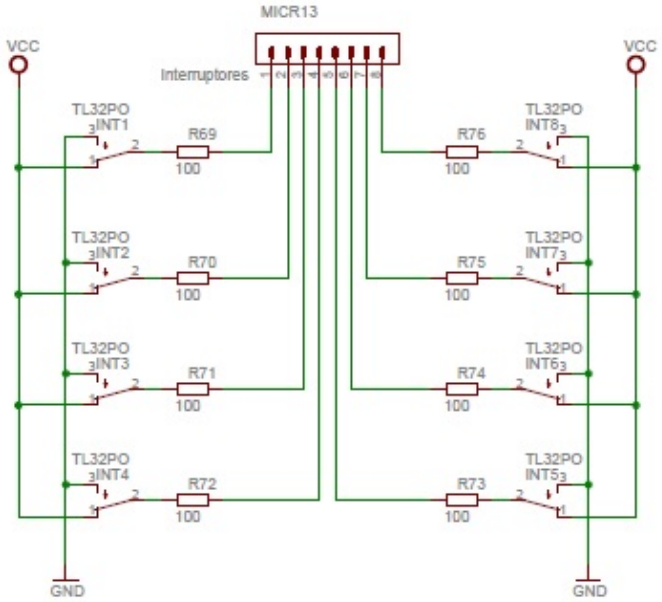


Figura 22. Esquema eléctrico de los conmutadores en el sistema de desarrollo.

5.1.6 Display 7 segmentos

Dentro de la visualización mediante displays se han implementado dos circuitos, una visualización dinámica y una visualización estática, el funcionamiento de ambos es similar. Consiste en enviar desde el puerto IO una codificación BCD que mediante un decodificador BCD a 7 segmentos se visualiza en un display.

La visualización dinámica consiste en activar alternativamente cada uno de los displays, lo suficientemente rápido, de forma que para el ojo humano los cuatro displays están encendidos a la vez. Se ha desarrollado el circuito ya existente en las prácticas de SEP.

Primero se ha comprobado que el único circuito digital CD4511, conversor BCD 7 segmentos, se podía alimentar a 3,3V. Una vez comprobado, se ha sustituido por una versión en montaje superficial que tiene un encapsulado TSSOP 16 pines. Se encuentra disponible en RS-Amidata con el código 809-6220P.

Cada uno de los componentes se ha sustituido por su versión SMD. Así el transistor NPN es un BC847BW de la firma NXP disponible en RS-Amidata con el código 303-1596, el transistor PNP es un BC857BW también es de NXP y se encuentra en RS-Amidata referencia 711-4884. El encapsulado de ambos es SOT323.

El display es de cátodo común fabricado por Avago Technologies y disponible en RS-Amidata con el código 435-6751. Se trata del display que se utiliza en las prácticas de SEP. Se ha planteado cambiarlo por un display de montaje superficial, pero no se producía una mejora sustancial respecto del anterior, por lo que se ha mantenido en nuestro diseño.

Se han calculado las resistencias para cada segmento con la nueva alimentación de 3,3V.

$$R_{13} = \frac{V - V_Z}{I} = \frac{3{,}3V - 1{,}8V}{10mA} = 165\Omega$$

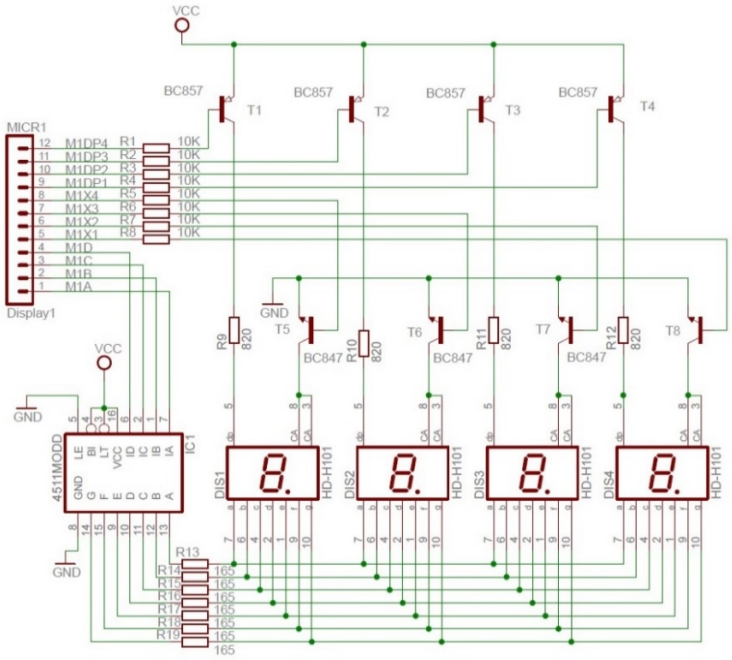


Figura 23. Visualización dinámica mediante displays en el sistema de desarrollo.

Dado que en la plataforma Freescale FRDM-KL25Z existen disponibles un número elevado de puertos IO, contrariamente a lo que sucedía con la placa DEMO9S08QG8, se ha implementado en el sistema de desarrollo una visualización estática.

En este tipo de visualización, cada display tiene asociado un decodificador BCD 7 segmentos que siempre se encuentra activado.

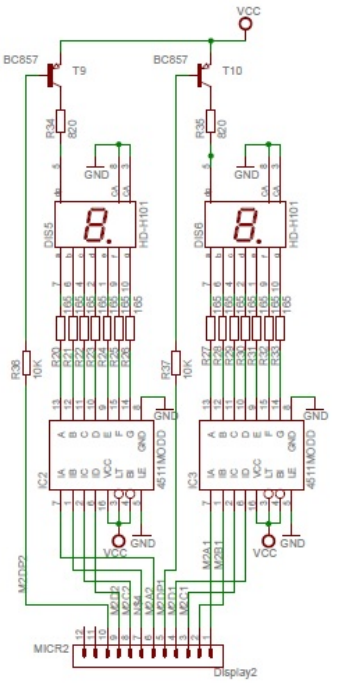


Figura 24. Visualización estática mediante displays en el sistema de desarrollo.

5.1.7 Teclado

El número de puertos IO disponible es mayor, por lo tanto, se dispone de un teclado de 16 teclas controlado por software donde cada línea de control está conectada directamente a un puerto IO de la plataforma Freescale FRDM-KL25Z. El microcontrolador “escribe” en las filas Y, y “lee” las columnas X. Se han incluido sendas resistencias de pull-down de 10KΩ, encapsulado R1206, en las columnas.

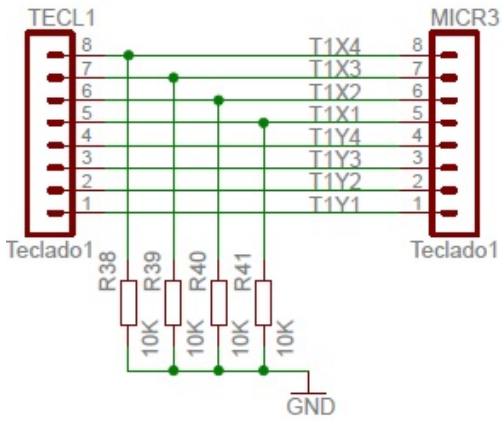


Figura 25. Teclado controlado por software.

Se incluye también, un teclado de 16 teclas controlado por hardware. Para ello se dispone de un codificador de teclado MM74C922N de National Semiconductor disponible en el área de Tecnología Electrónica, se encuentra descatalogado en RS-Amidata, Farnell element14 y Digi-Key.

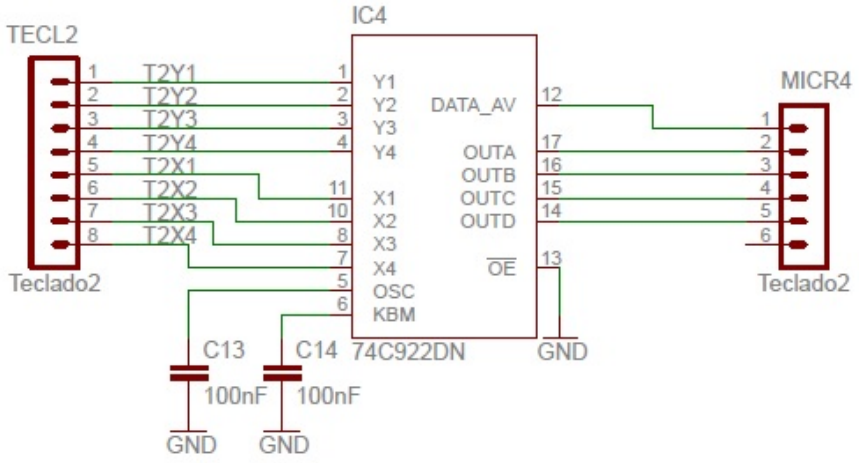


Figura 26. Teclado controlado por hardware.

5.1.8 Módulo LCD

El módulo LCD disponible es alfanumérico de 16 caracteres por 2 filas, modelo DVF16244S1FBL lleva incluido el controlador Hitachi HD44780U. Este driver controlador se puede configurar como 4 bits o como 8 bits.

El controlador consta de dos registros de 8 bits, un registro de instrucciones y otro de datos. Se puede controlar el brillo y el contraste.

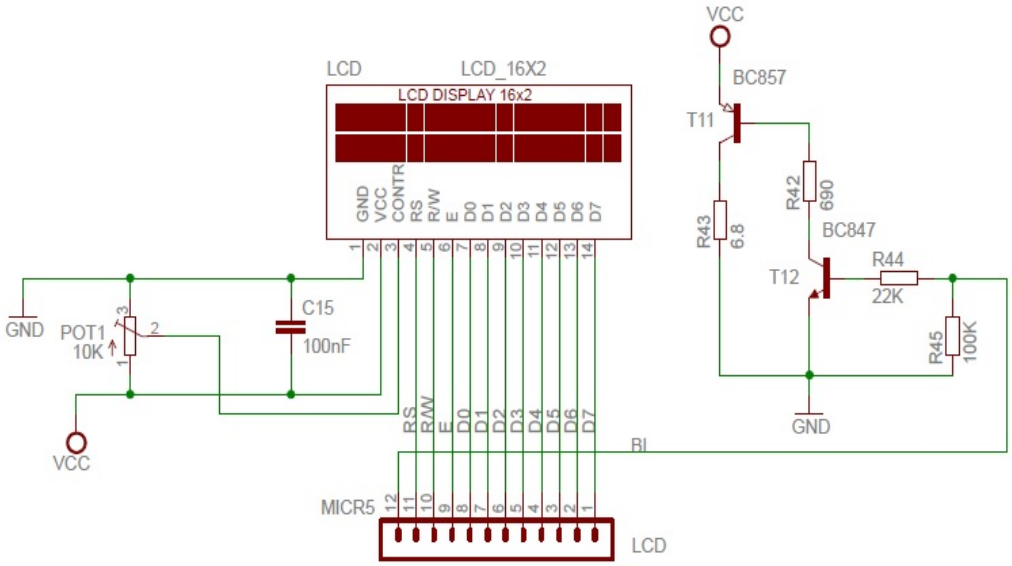


Figura 27. Esquema eléctrico conexión LCD.

5.1.9 Memoria EEPROM serie

Se ha implementado una memoria serie 24AA256SN I2C de Microchip Technology Inc., de 256Kbits (32Kx8) EEPROM con un rango de alimentación entre 1,7 y 5,5V. Ha sido desarrollada para aplicaciones de baja potencia. Se encuentra disponible en RS-Amidata, código 687-9161. Se presenta en encapsulado 8 pines SOIC.

Las líneas de control A0, A1 y A2 permiten hasta 8 dispositivos en el bus I2C. SDA es un pin bidireccional, colector abierto, que permite el tráfico de datos en la memoria. Requiere una resistencia de pull-up de 10KΩ. Para la transferencia de datos, SDA puede cambiar solamente durante el flanco bajo de SCL. SCL es la entrada de sincronismo de reloj.

La memoria 24AA256SN soporta un bus de 2 hilos bidireccional y el protocolo I2C de transmisión de datos. El bus debe ser controlado por un dispositivo maestro que genera la señal del reloj serie (SCL), controla el acceso al bus, genera las condiciones de start y stop mientras la memoria 24AA256SN trabaja como un esclavo. Amo y esclavo pueden funcionar como transmisor o receptor, pero el dispositivo maestro determina el modo activado.

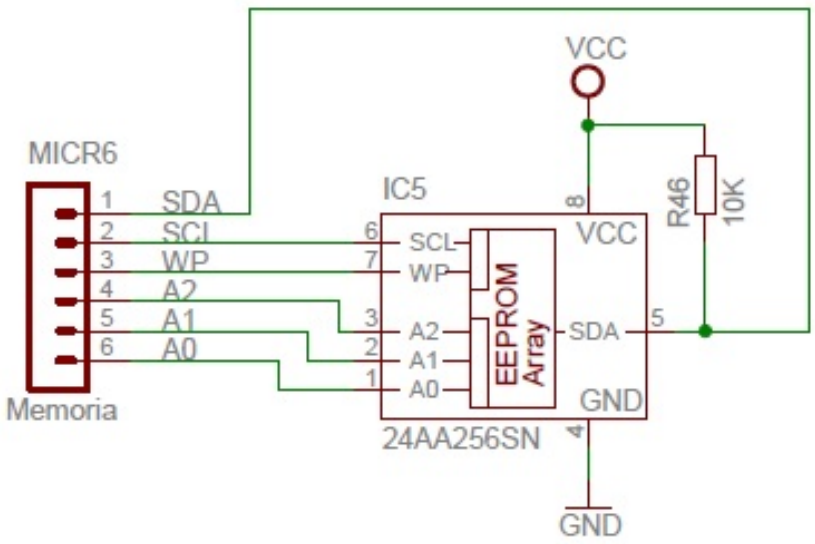


Figura 28. Memoria EEPROM serie.

5.1.10 Conversor analógico-digital serie de 12 bits

El conversor analógico digital AD7992BRMZ es de 12 bits de aproximaciones sucesivas con un interface I2C serie. Tiene un rango de funcionamiento entre 2,7 y 5,5V. Cuenta con un tiempo de conversión de 2 µs, puede trabajar hasta 11MHz de frecuencia. Está fabricado por Analog Devices y se encuentra disponible en RS-Amidata con referencia 758-9976.

El proceso de conversión puede controlarse usando la entrada CONVST. Se activa con el flanco ascendente durante 1µs. En el flanco descendente se inicia la conversión. Se debe permitir un tiempo de puesta en marcha de al menos 1 µs con el pulso a nivel alto CONVST, de lo contrario, el resultado de la conversión no es válido.

Tiene dos entradas de referencia, mediante el conector SV3 y el jumper JP3 se selecciona todas las posibles opciones de entrada. La referencia VIN1 es variable entre 0 y REF_{IN}. VIN2/REF_{IN} puede tomar dos valores, externo o el proporcionado por el potenciometro POT3 incluido en el sistema de desarrollo.

En modo simple, requiere una referencia externa en el rango de 1,2 V a la tensión de alimentación (3,3V). Esto permite una amplia gama de entrada para el conversor. Existen unos registros que pueden ser programados como límites, alto y bajo para el resultado de la conversión, si la salida se encuentra fuera del rango se produce una alerta, pin de salida ALERT/BUSY. Se activa cuando el resultado de la conversión supera los límites programados. Esta salida puede utilizarse como una interrupción.

AS, entrada de selección de la dirección I2C. SDA IO bidireccional, de colector abierto. Se requiere una resistencia de pull-up de 10KΩ. SCL entrada de sincronismo de reloj. Requiere resistencia de pull-up.

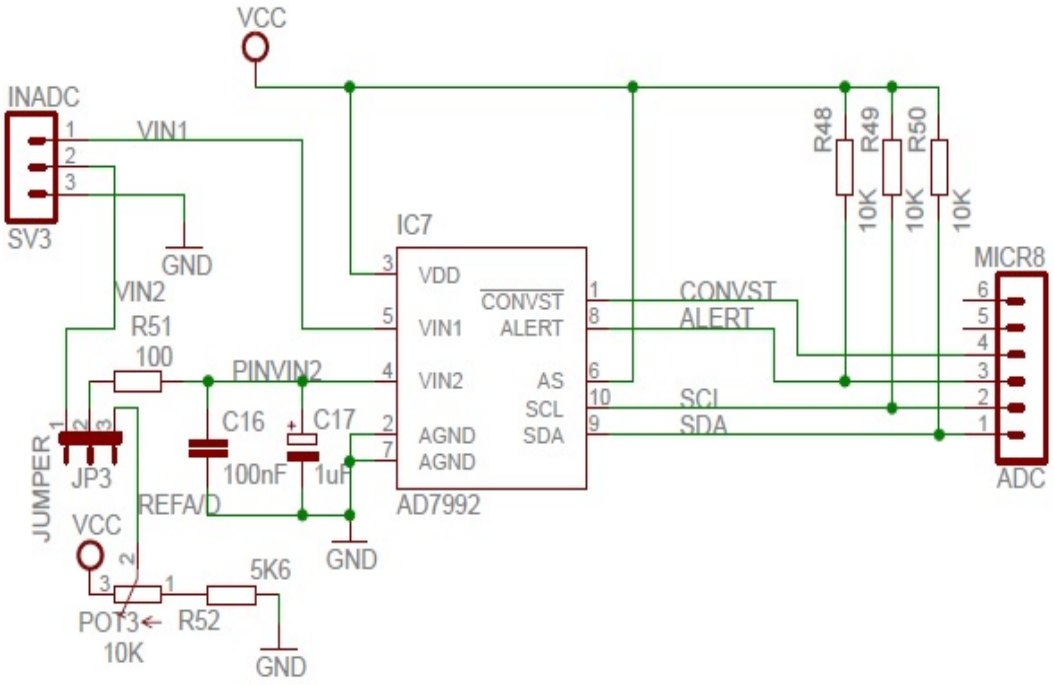


Figura 29. Conversor ADC serie.

5.1.11 Conversor digital-analógico serie

El conversor digital analógico serie LTC2622CMS8 es un conversor doble de 12 bits, de Linear Technology, de tensión de alimentación comprendida entre 2 y 5,5V de salida rail-to-rail, en un encapsulado de 8 pines MSOP. Se encuentra disponible en RS-Amidata y su referencia es 806-1741.

Dado que todos los dispositivos estudiados hasta el momento son I2C, he creído necesario incluir un dispositivo ISP que utiliza un interface serie distinto.

CS/LD, permite seleccionar el integrado correspondiente en el bus ISP, o bien cargar datos. Cuando CS/LD está a nivel bajo, SCK está habilitado para cambiar datos de SDI en el registro. Cuando CSLD está a nivel alto, SCK esta deshabilitado y por tanto se ejecuta el comando especificado.

SCK, entrada de sincronismo de reloj. SDI, entrada de datos de interfaz. REF, entrada de voltaje de referencia. Está comprendida entre 0V ≤ VREF ≤ VCC.

VOUTA y VOUTB, voltaje analógico de salida del conversor. El rango de salida es de 0 hasta VREF.

El jumper JP2 posibilita la elección de una tensión de referencia externa o la suministrada por el potenciómetro POT2 disponible en la placa.

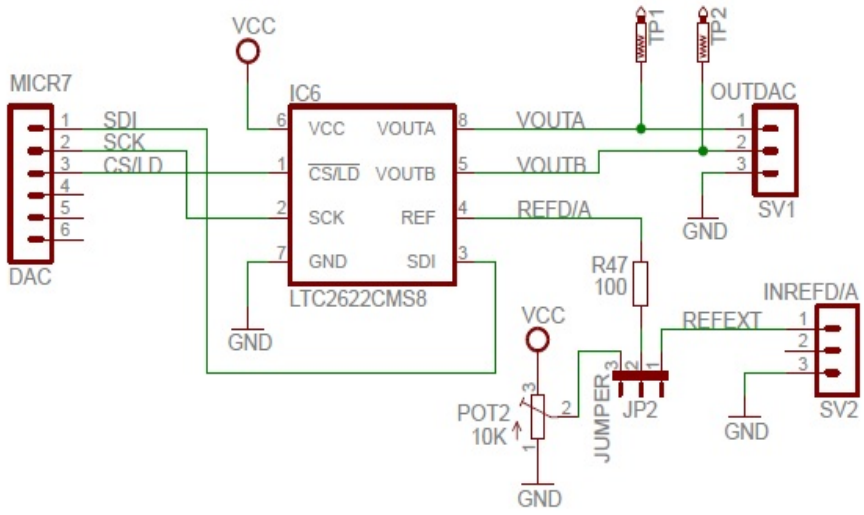


Figura 30. Conversor DAC serie.

5.1.12 Potenciómetro

Se encuentran implementados dos potenciómetros en el sistema de desarrollo, un potenciómetro digital y un potenciómetro analógico.

El potenciómetro digital es de 7 bits, fabricado por Microchip, tiene como referencia MCP454102CMS8, se encuentra en Farnell element14 y su código es 169-8945.

Es de valor de 10KΩ, al tratarse de 7 bits, tiene 128 resistencias, 129 saltos. Soporta el protocolo de comunicación I2C, solo puede operar en modo esclavo (no es capaz de generar la señal de reloj). No necesita resistencias de pull-up.

A0 permite seleccionar el dispositivo en la comunicación I2C. SCL es la entrada del reloj. SDA es la entrada serie de colector abierto. POB, es el terminal B del potenciómetro, corresponde a la dirección 0x00 para el caso de 7 bits. POA, es el terminal A del potenciómetro, corresponde al fondo de escala y en el caso de 7 bits a 0x80. Ninguno de los terminales tiene polaridad entre sí, por tanto, se les puede aplicar corriente positiva o corriente negativa.

POW, corresponde al terminal conectado en el cursor, el voltaje en el terminal cursor está comprendido entre Vcc (3,3V) y GND.

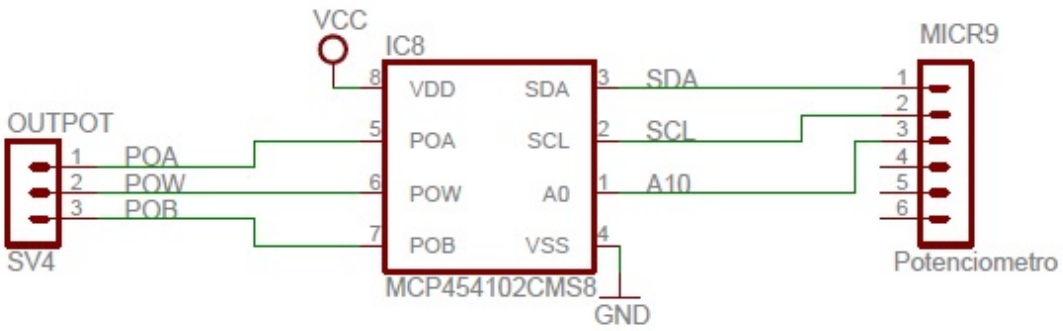


Figura 31. Potenciómetro digital serie.

Se incorpora igualmente un potenciómetro analógico de 10KΩ, se encuentra disponible en RS-Amidata con el código 486-7627. Se trata de un potenciómetro de la serie TS53YJ de Vishay.

Se ha protegido el cursor del potenciómetro colocando una resistencia en serie con dicho terminal de 100Ω.

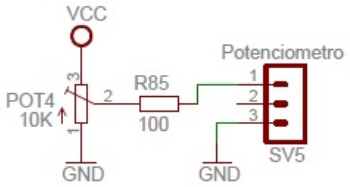


Figura 32. Potenciómetro analógico de 10KΩ.

5.1.13 Acelerómetro analógico

Se trata de un módulo SparkFun Electronics, Triple Axis Accelerometer Breakout basado en el acelerómetro analógico de 3 ejes ADXL335 de Analog Devices. Se trata de uno de los últimos sensores analógicos que ha aparecido en la industria tiene un consumo mínimo, y tiene ultra bajo ruido. Rango de utilización de $\pm 3g$. El módulo se puede alimentar entre 1,8 y 3,6V y tiene comunicación I2C.

Figura 33. Acelerómetro analógico basado en ADXL335.

Se encuentra disponible en Digi-Key con el código 1568-1044-ND.

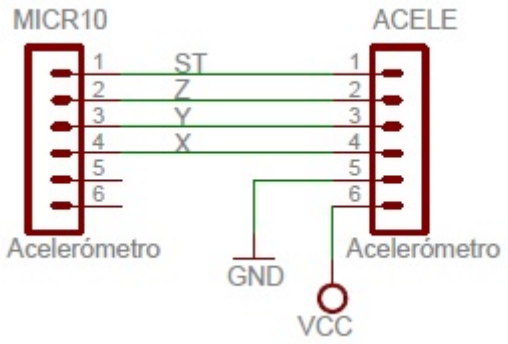


Figura 34. Conexión del módulo acelerómetro.

5.1.14 Sensores

Sensores de temperatura, es un sensor de temperatura basado en el LM35, alimentado a 3,3V el TMP36 de Analog Devices, disponible en RS-Amidata con referencia 427-351.

La serie TMP de sensores de temperatura genera una salida de tensión linealmente proporcional a la temperatura en grados centígrados, no es necesario realizar calibración externa y puede conectarse fácilmente a circuitos de control de temperatura y ADC. Las aplicaciones incluyen gestión y protección térmica, control ambiental, control de procesos industriales, alarmas de incendios y monitores de sistemas de potencia.

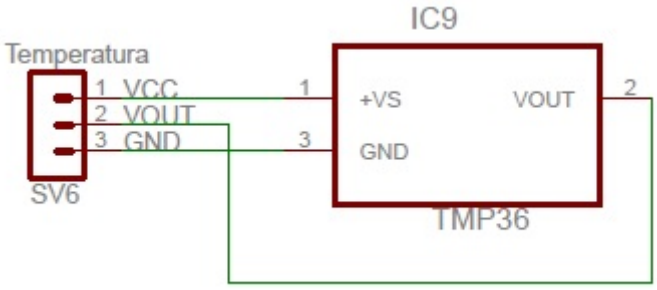


Figura 35. Sensor de temperatura lineal.

Del mismo modo, se incluye un módulo SparkFun Electronics, sensor de temperatura y humedad basado en el dispositivo de Honeywell HumidIcon Digital HIH6130.

Este dispositivo es un sensor de temperatura combinado con un sensor de humedad relativa con salida digital. Se obtiene la medición de la humedad relativa y es compensada con la medida de la temperatura. Proporciona una salida del sensor independiente de la temperatura.

Se encuentra disponible en Digi-Key y su referencia es 1568-1020-ND.

Tensión de funcionamiento entre 2.3 y 5.5V, rango de humedad entre 10-90% RH, rango de temperatura 5-50° C.

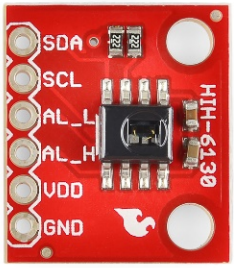


Figura 36. Sensor de humedad relativa y temperatura.

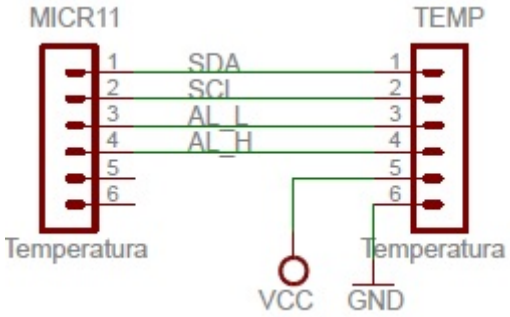


Figura 37. Conexionado del sensor de humedad relativa y temperatura.

Sensor de humedad, se trata del sensor de humedad relativa HCZ-J3-B de Multicomp, presenta una resistencia mayor cuanto menor es la humedad relativa. Se encuentra disponible en Farnell element14 con código 189-1432. Sus aplicaciones van desde medir la humedad relativa, aire acondicionado, humidificadores e higrómetros.

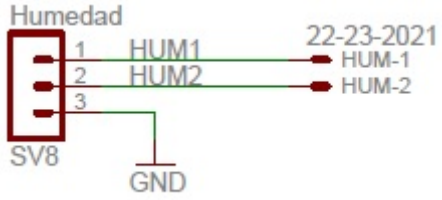


Figura 38. Sensor de humedad relativa.

Sensor de presión, he elegido el dispositivo de Freescale MPX12DP, es un sensor de presión diferencial para aire, alimentado a 3,3v. Es un sensor piezoresistivo, que proporciona una salida de tensión lineal directamente proporcional a la aplicación de presión ejercida.

Ejemplos de aplicaciones son, controles de aire, controles de sistemas, indicador de nivel, instrumentación médica, control industrial, control de sistemas neumáticos, robótica.

Se encuentra disponible en RS-Amidata, con la referencia 719-1024.

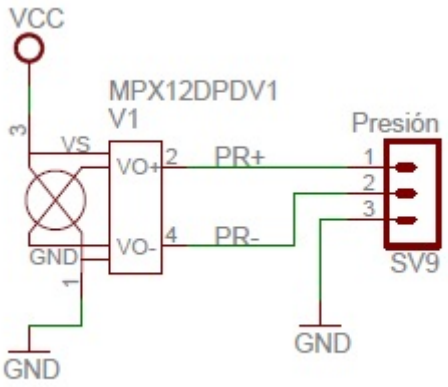


Figura 39. Sensor de presión.

Se incluye igualmente un zumbador piezoeléctrico, modelo KPEG242 de Kingstate de 70dB, continuo, de una frecuencia de 3600 a 4600Hz. Tiene el código 535-8253 y se encuentra disponible en RS-Amidata.

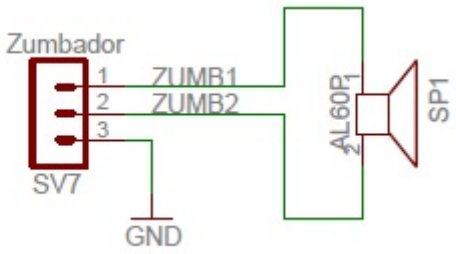


Figura 40. Zumbador piezoeléctrico.

Inicialmente, se pretendía incluir un número mayor de sensores, sensor hall (SS495A), sensor magnético (AA747AHA-LB), célula de carga, galgas extensiométricas (RS N11MA512023), sensor de proximidad, sensor de vibración (AU2402-1), termopares. Se ha desechado incluirlos en este prototipo debido al coste excesivo que suponían.

5.2 PCB del Sistema de Desarrollo

Un circuito impreso o PCB en inglés, es una tarjeta o placa utilizada para realizar el emplazamiento de los distintos elementos que conforman el circuito y las interconexiones eléctricas entre ellos. El circuito impreso se utiliza para conectar eléctricamente a través de las pistas conductoras, y sostener mecánicamente, por medio de la base, un conjunto de componentes electrónicos.

Las pistas son generalmente de cobre mientras que la base se fabrica de fibra de vidrio reforzada. La mayoría de los circuitos impresos están compuestos por entre una y dieciséis capas conductoras, separadas y soportadas por capas de material aislante (sustrato) laminadas (pegadas) entre sí.

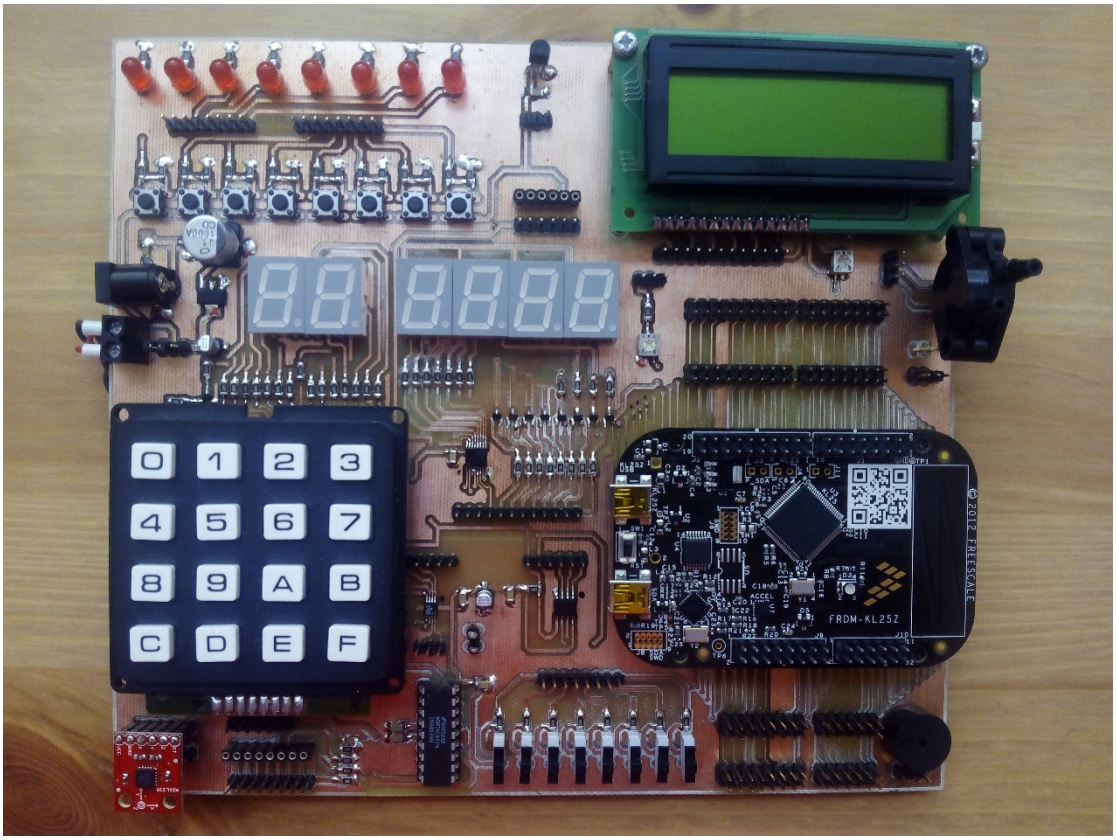


Figura 41. Prototipo PCB.

Se ha desarrollado y construido el prototipo del Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes. Se trata de un primer prototipo, que hay que desarrollar hasta llegar a un sistema real que pueda utilizarse en las prácticas de SEP.

El planteamiento inicial es que la placa fuera versátil y universal, para lo cual se ha distribuido por módulos independientes. Las conexiones de los diversos módulos con el sistema de desarrollo y evaluación FRDM-KL25Z se realizan mediante conectores y a su vez con cables, de forma que el alumno en cada caso elige que puerto utilizar.

En el prototipo siguiente se implementará entre otras cosas la posibilidad de utilizar este sistema de desarrollo con la placa actual DEMO9S08QG8 de Freescale.

Se puede observar, la no coincidencia entre el primer prototipo y la serigrafía de componentes debido a las modificaciones que se han tenido que realizar para su funcionamiento.

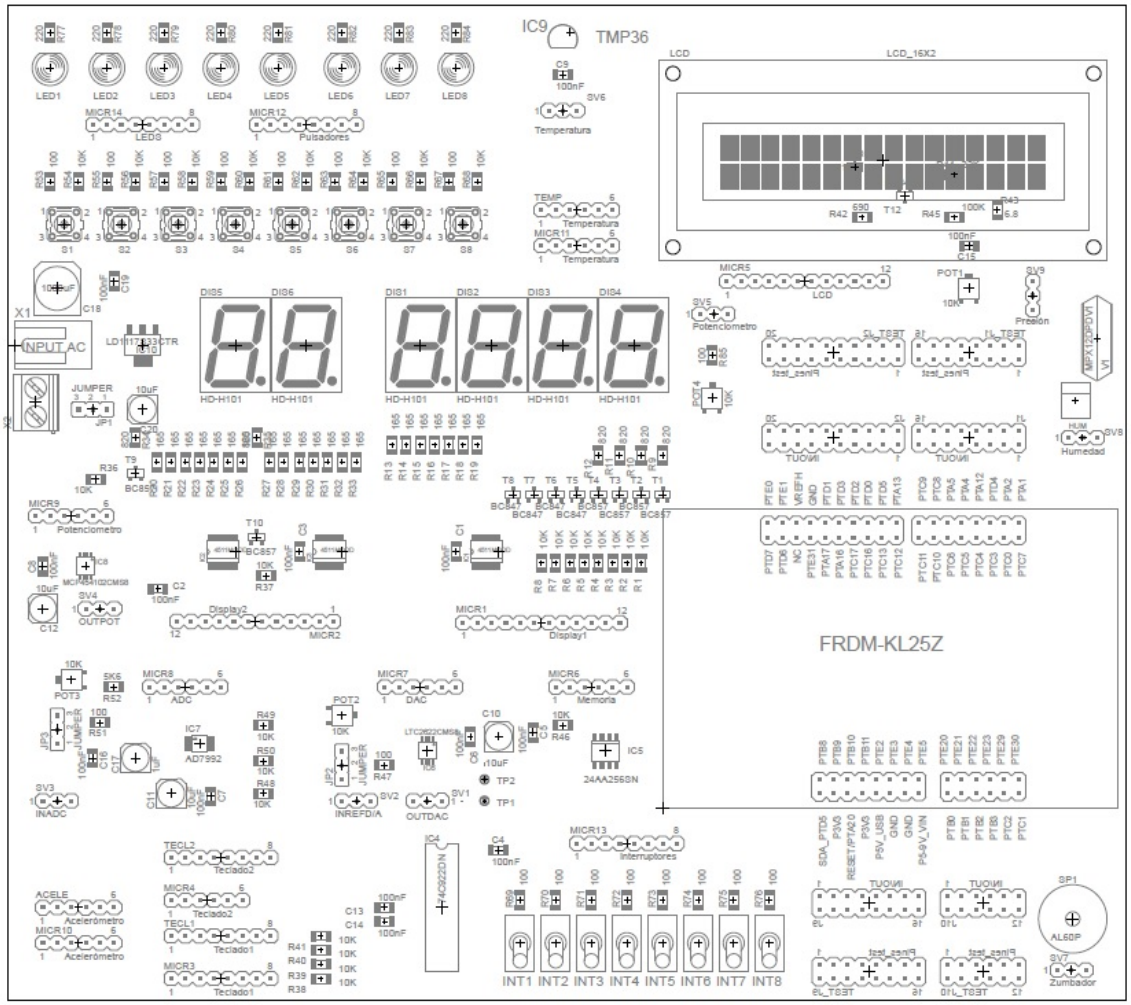


Figura 42. Serigrafía componentes de la PCB.

Los dibujos y texto se pueden imprimir en las superficies exteriores de un circuito impreso a través de la serigrafía. El texto de la serigrafía indica los nombres de los componentes, la configuración de los interruptores, puntos de prueba, y otras características útiles en el ensamblaje, prueba y servicio de la tarjeta.

El emplazamiento y disposición de los componentes en la placa, el lugar que ocupan, su posición orientación, debe guardar una cierta lógica. La mayoría de las tarjetas de circuito impreso están constituidas por dos clases de componentes.

Componentes de inserción THD (Trough Hole Device). En estos componentes las patillas se insertan a través de los agujeros (nodos) para su posterior soldadura.

Componentes de montaje en superficie SMD (Surface Mounted Device). Con esta tecnología se logran tarjetas de muy alta densidad de componentes.

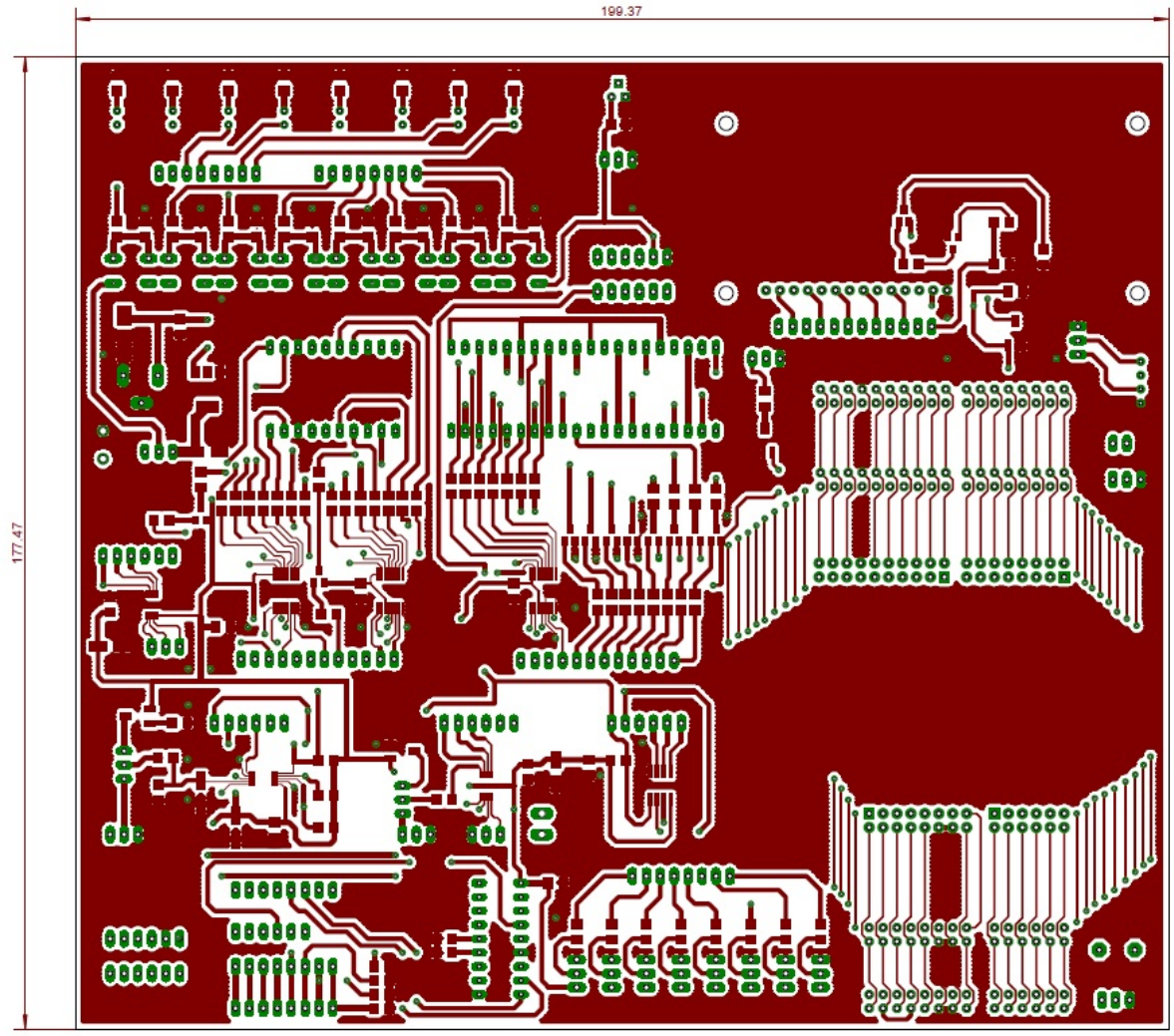


Figura 43. Cara TOP de la PCB.

Se representan en la figura 43 y en la figura 44 las caras TOP y BOTTOM correspondientes al diseño del primer prototipo del Sistema de desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes.

Cuando se va a hacer la distribución y emplazamiento de componentes para diseñar el circuito impreso es necesario conocer cada tipo de componente que se va a emplear, sus terminales de soldadura, tipo de encapsulado, dimensiones.

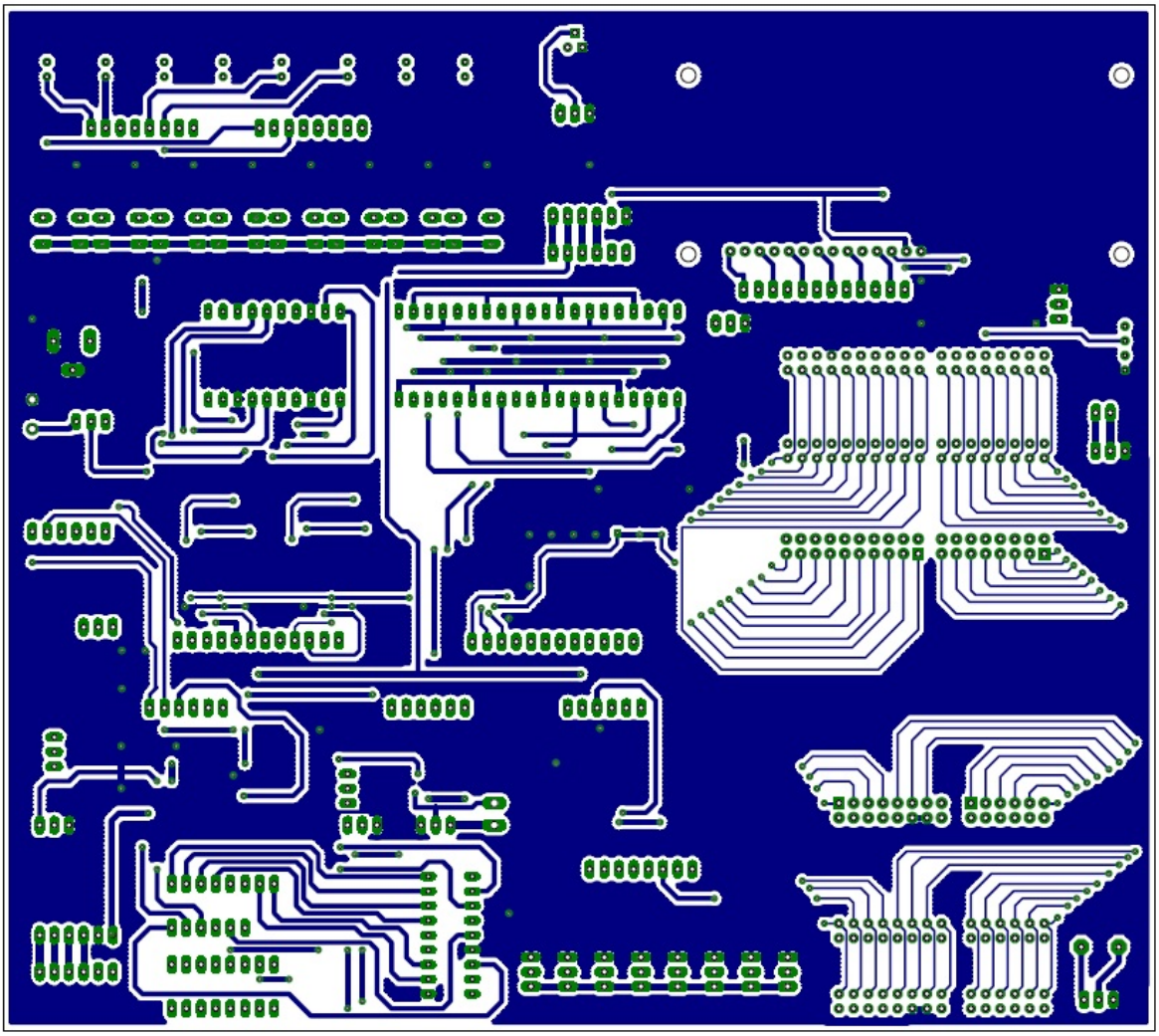


Figura 44. Cara Bottom de la PCB.

Se amplía información sobre la PCB y su diseño en el Anexo A3 Planos PCB del Sistema de Desarrollo, de este documento.

6. Codewarrior Development Studio

Se ha utilizado CodeWarrior Development Studio (v10.6_SE), ya que el entorno de trabajo es similar al empleado actualmente en las prácticas de SEP. Se procede a explicar cómo se instala en el ordenador, del mismo modo existe una ampliación de esta instalación en el Anexo A1, Documento Ayuda para la plataforma Freescale Freedom FRDM-KL25Z.

CodeWarrior trabaja con proyectos, un proyecto es una carpeta en la que se van incluyendo todos los ficheros que se realizan, el programa, funciones, archivos para simulación, depuración, etc. Existe una ampliación de este punto en el Anexo A1, Documento Ayuda.

6.1 Instalación de CodeWarrior Development Studio (v10.6_SE)

CodeWarrior Development Studio (v10.6_SE, http://www.freescale.com) es un potente entorno de desarrollo integrado (IDE, Integrated Development Environment) para los microprocesadores de Freescale (arquitecturas del tipo Coldfire, Coldfire+, DSC, Kinetis, MPC5xxx, RS08, S08 y S12Z), incluyendo los últimos dispositivos y dando soporte a las nuevas versiones.

CodeWarrior está basado en la plataforma Eclipse IDE 4.2.1 (Juno), consta de múltiples módulos. Se va a instalar la versión v10.6_SE, para la familia Kinetis L (plataforma FRDM-KL25Z). Existen distintas ediciones de CodeWarrior, desde la demo hasta la profesional. Se ha elegido, la versión “Special Edition”, que incluye gratuitamente prácticamente todas las funcionalidades, limitando el código C a 64KB.

Como primer paso, hacer doble click en el fichero ejecutable de la aplicación y aceptar los términos de la licencia.

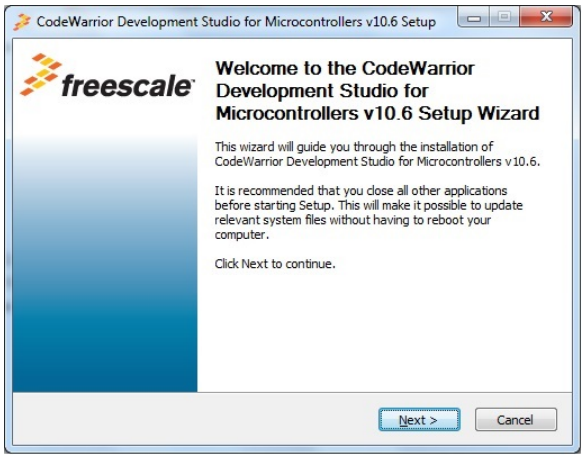


Figura 45. Página de bienvenida a la instalación.

Seleccionar los componentes que queremos instalar, para nuestro caso, solamente seleccionaremos Kinetis, aunque se pueden instalar para todo tipo de familias de dispositivos.

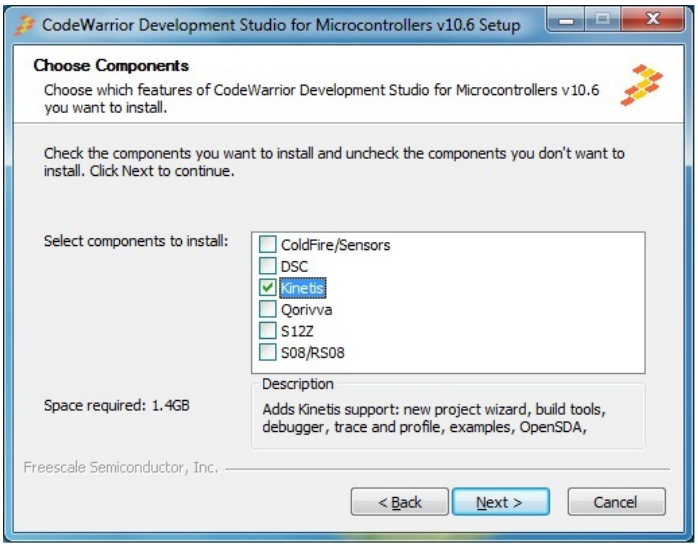


Figura 46. Selección de los componentes a instalar.

A continuación, selecciona la carpeta donde se va a proceder a instalar el programa, la que viene por defecto, u otra que se desee.

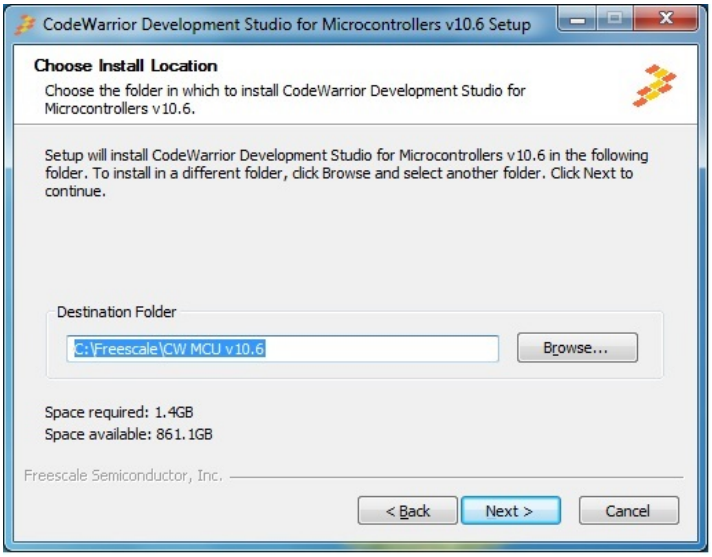


Figura 47. Carpeta donde se instala.

Inicio de la instalación, se conecta a la página web de Freescale para descargar los componentes seleccionados previamente.

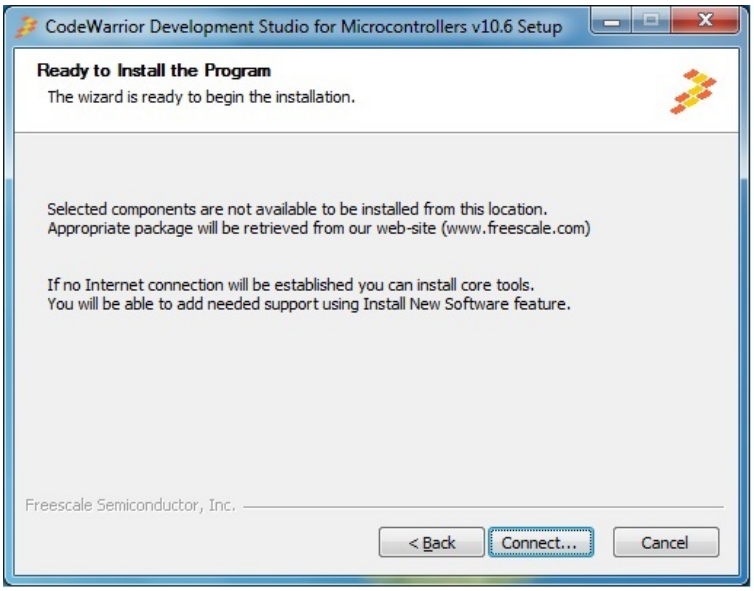


Figura 48. Inicio de la instalación de CodeWarrior.

Posteriormente, hay que instalar distintos software de dispositivos que va apareciendo durante la instalación.

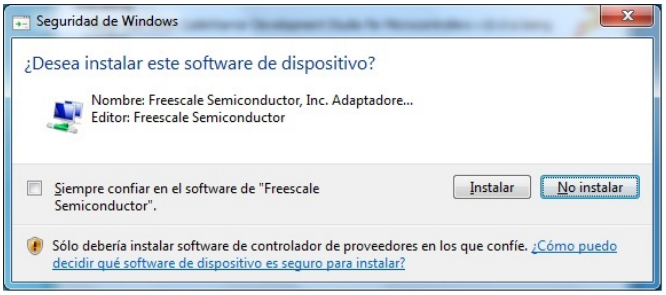


Figura 49. Pantalla tipo de instalación de software de dispositivo.

Se ha completado la instalación de CodeWarrior Development Studio v10.6_SE.

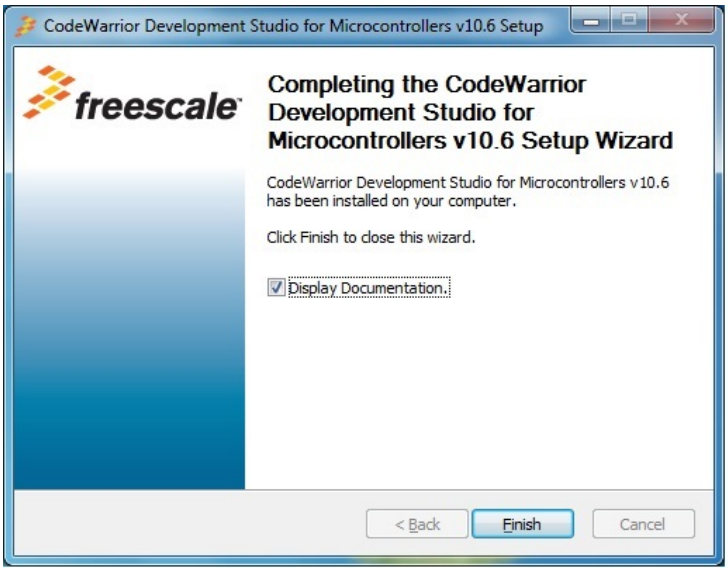


Figura 50. Fin de la instalación de CodeWarrior.

Una vez completada la instalación, aparece una ventana del navegador con la página web de Freescale, donde se pueden descargar diversos manuales entre ellos la guía de uso rápido en formato pdf (***FRDM-KL25Z Quick Start Guide***).

6.2 Como crear un proyecto, trabajando con CodeWarrior

Al arrancar CodeWarrior aparece una pantalla que permite seleccionar la carpeta de trabajo (workspace), donde CodeWarrior guarda los proyectos. Se puede dejar la carpeta que está por defecto u elegir otra que interese.

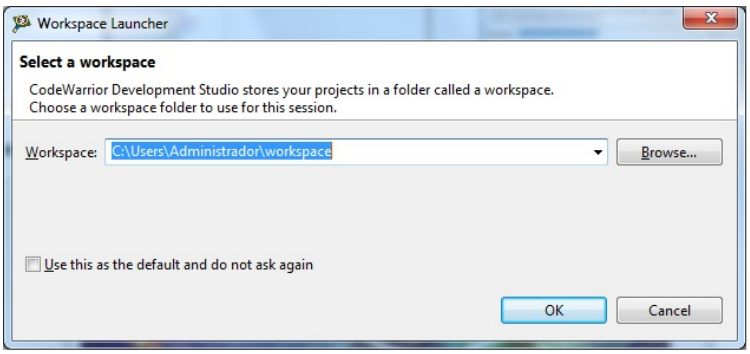


Figura 51. Carpeta donde CodeWarrior guarda los proyectos.

Aparece a continuación el menú de arranque, que nos permite cargar un ejemplo existente, crear un nuevo proyecto, descargar actualizaciones,…

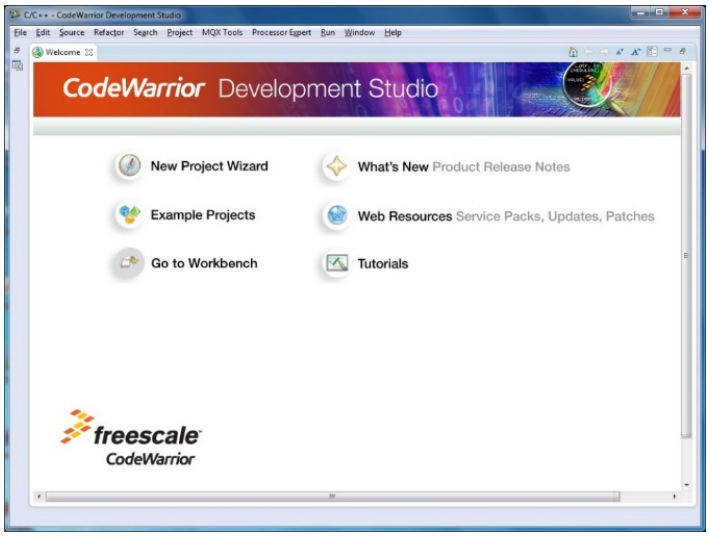


Figura 52. Menú de arranque de CodeWarrior.

Al crear un nuevo MCU Bareboard Project, el programa nos solicita el nombre del proyecto y su ubicación.

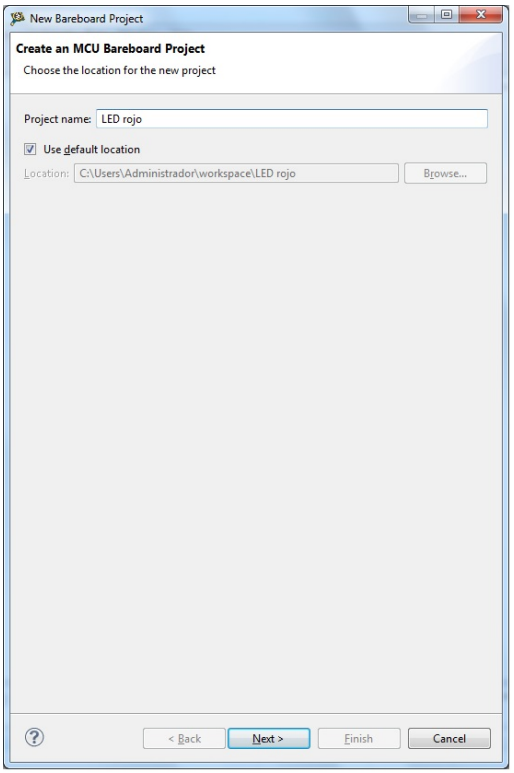


Figura 53. Crear un nuevo MCU Bareboard Project.

Una vez seleccionado el nombre del proyecto se continúa con la selección del dispositivo que se va a utilizar. En nuestro caso, se trata de un microcontrolador de la serie L de Kinetis, de la familia KL2x, el micro MKL25Z128.

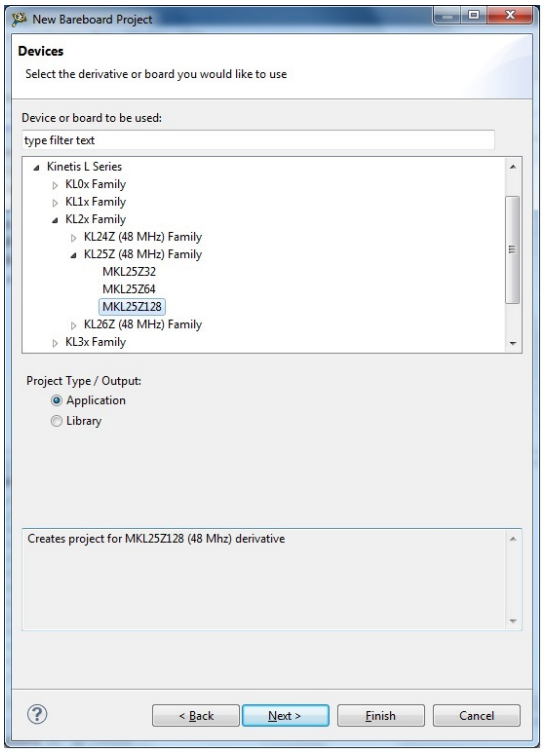


Figura 54. Selección del microcontrolador de Kinetis.

En el siguiente paso, nos pregunta qué tipo de conexión se va a usar entre CodeWarrior y la tarjeta de desarrollo y evaluación. Seleccionamos OpenSDA.

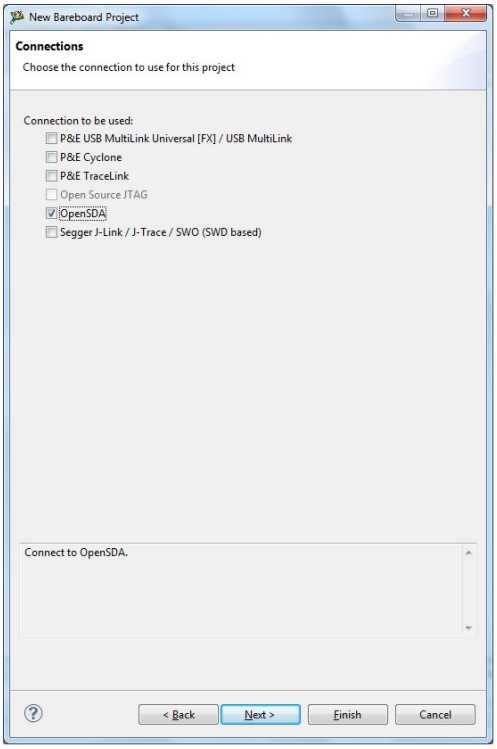


Figura 55. Selección de la conexión OpenSDA.

Posteriormente, la siguiente pantalla nos permite elegir entre lenguaje ensamblador, C y C++. Para nuestro caso, seleccionamos lenguaje C. En cualquier caso depende de nuestro sistema de desarrollo y evaluación.

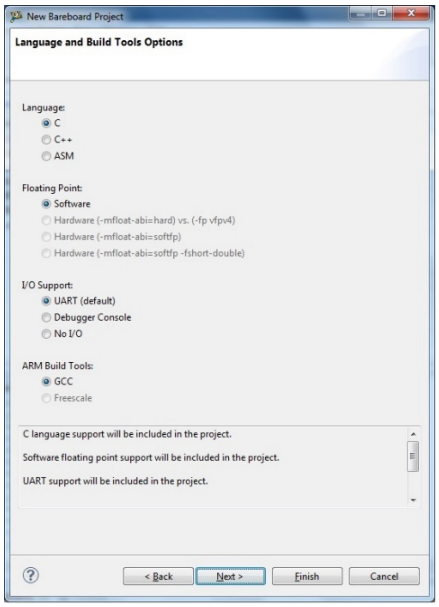


Figura 56. Lenguaje y herramientas de Build.

Por último, la ventana correspondiente a Proccesor Expert, si seleccionamos la opción disponible de Processor Expert se pueden configurar rápidamente dispositivos periféricos en el microcontrolador, o desde una biblioteca de módulos de código que pueden implementar varios servicios de dispositivos tales como temporizador, conversores,…

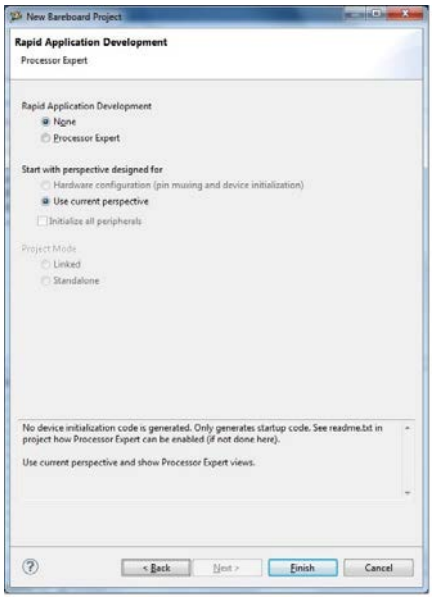


Figura 57. Opciones Processor Expert.

Una vez finalizadas todas las ventanas anteriores ya hemos creado un MCU Bareboard Project, se abre una pantalla, en el lado izquierdo hay un menú desplegable donde se pueden gestionar los proyectos que se encuentran en la carpeta workspace. Con el botón derecho abrimos este menú de gestión.

Justo debajo de este menú, existe una ventana de comandos, se puede activar desde la carpeta Project Settings del menú de proyectos. Nos va a permitir editar la configuración de debugger.

Siguiendo hacia la derecha en la pantalla, existe una pestaña donde podemos programar el fichero main.c, programa principal en lenguaje C. Debajo de la anterior, existe otra ventana que permite comprobar si el programa en C que estamos escribiendo no tiene errores.

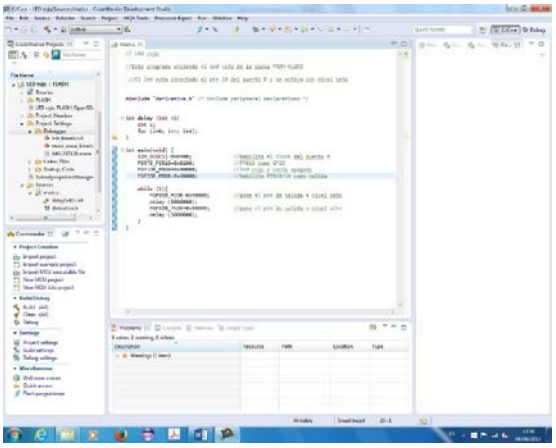
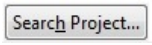


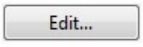
Figura 58. Programación en lenguaje C.

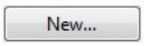
Una vez escrito el programa en la ventana central, podemos editar la configuración de debugger.

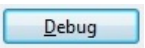
El programa ha creado una configuración de debugger con el nombre del proyecto y el tipo de conexión que hemos elegido (LED rojo_FLASH_OpenSDA).

En la pestaña main, nos permite seleccionar el tipo de sesión de debugger (Download). Indica después que se trata de una aplicación creada mediante lenguaje C/C++. Te permite buscar el proyecto mediante el botón 

Indica el nombre del fichero que vamos a cargar dentro del sistema desarrollo y evaluación (FLASH/LEDrojo.elf). En este momento, se puede seleccionar otro fichero buscándolo en 

En Target settings se puede editar 

la conexión existente y crear una conexión nueva. 

Si procedemos a hacer doble click en 

ejecutamos la aplicación.

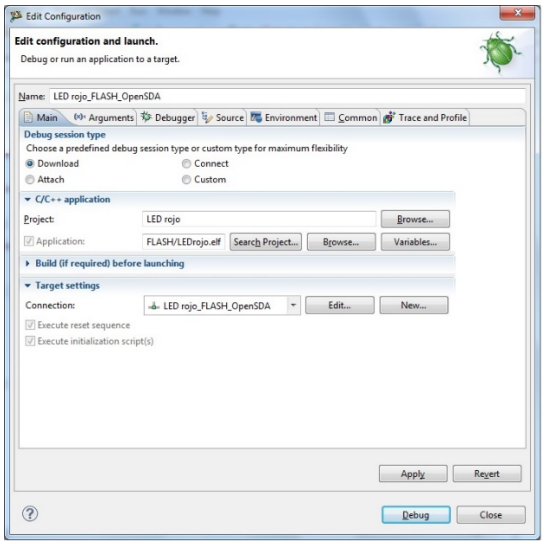


Figura 59. Edición de la configuración de debugger.

Una vez se ha ejecutado la aplicación, estamos en la ventana de debugger. Se pueden intercambiar la pantalla de debugger con la pantalla de programación con las teclas siguientes 

Con la aplicación en marcha, se puede ejecutar paso a  paso o bien de modo continuo  , indistintamente. Estos modos se encuentran disponibles en el panel de control.

Utilizar en modo paso a paso, permite visualizar el estado de los diferentes registros o el valor que en cada caso tienen cada una de las variables (marcado con amarillo en la figura 63).

Una vez que se ha ejecutado en modo continuo, el fichero correspondiente esta grabado en la memoria FLASH del sistema de desarrollo y evaluación FRDM-KL25Z.

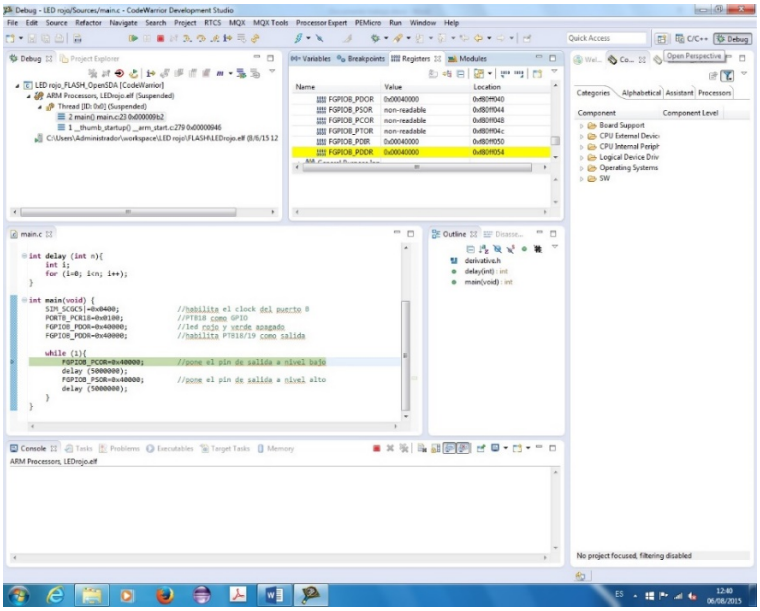


Figura 60. Página de debugger, ejecución paso a paso.

7. Librerías implementadas

Además del diseño de la PCB del Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes, se incluyen varias funciones realizadas con el fin de comunicar el sistema de desarrollo con la placa de desarrollo y evaluación. Estas funciones deben de servir de base para trabajar con microcontroladores de 32 bits. En esta memoria solo se incluyen las funciones, la inicialización y los programas necesarios para su ejecución se encuentran en el Anexo A2, Código implementado en el proyecto.

7.1 Display 7 segmentos con visualización estática

Consiste en visualizar un número comprendido entre 00 y 99 en dos displays. Se pasa a la función el número en 8 bits, los cuatro menos significativos los correspondientes a las unidades y los otros cuatro a las decenas.

La función obtiene por métodos matemáticos cada digito por separado, y posteriormente carga la decena desplazada 4 lugares hacia la izquierda en un registro auxiliar. Para finalizar suma el valor de este registro con el valor de unidades y lo carga en la salida conectada a los decodificadores BCD a 7 segmentos disponibles en el sistema de desarrollo. La función desarrollada es la siguiente:

```
int VisEst (int dato) {

	int numero, auxiliar, unidad, decena;

	numero=dato;
	decena=numero/10;                  //obtengo el dígito decena
	unidad=numero%10;                  //obtengo el dígito unidad
	auxiliar=decena<<4;                //registro auxiliar que desplaza a la
                                           izquierda el digito decena
	FGPIOC_PDOR=auxiliar+unidad;       //salida de ambos displays, 4 bits decenas,
                                           4 bits unidades
	}
```

7.2 Display 7 segmentos con visualización dinámica

La visualización dinámica consiste en visualizar alternativamente cada display, de esta forma, para el ojo humano se visualizan los cuatro dígitos simultáneamente. Para ello se va a representar un número comprendido entre 0000 y 9999.

En un bucle primero encendemos el primer digito, esperar 1ms y después el siguiente. Si ampliamos el tiempo de espera se puede verificar el encendido alternativo de los displays.

Se realiza obteniendo cada una de las cifras por separado con operaciones matemáticas.

A continuación se carga el digito que toca representar y además se activa el display correspondiente. Este tipo de visualización nos permite solamente utilizar 4 bits como salida del número y otros 4 bits como control de activación de los displays. El número de salidas utilizadas es mucho menor que si se realiza una visualización estática, del orden de cuatro veces menos.

Se incluye en la función la subrutina que permite realizar la espera “delay”.

Del mismo modo, tenemos disponibles 4 bits dedicados a visualizar la coma, que se puede implementar como trabajo adicional, mediante una interrupción.

La función implementada se presenta a continuación.

```
int VisDin (int dato, int tiempo) {

	int delay (int n) {
		int i;
		for (i=0; i<n; i++);
	}

int numero, auxiliar, unidad, decena, centena, millar, temp;

numero=dato;

while (temp=tiempo/4) {

unidad=numero%10;				//obtengo unidad
auxiliar=unidad<<2;				//carga y desplazamiento del número
FGPIOE_PDOR=auxiliar+0X100000;			//salida, con número y display unidades
delay (3000);					//espera 1ms

decena=(numero/10)%10;				//obtengo decena
auxiliar=decena<<2;				//carga y desplazamiento del número
FGPIOE_PDOR=auxiliar+0X200000;			//salida, con número y display decenas
delay (3000);					//espera 1ms

centena=(numero/100)%10;			//obtengo centena
auxiliar=centena<<2;				//carga y desplazamiento del número
FGPIOE_PDOR=auxiliar+0X400000;			//salida, con número y display centenas
delay (3000);					//espera 1ms

millar=(numero/1000)%10;			//obtengo millar
auxiliar=millar<<2;				//carga y desplazamiento del número
FGPIOE_PDOR=auxiliar+0X800000;			//salida, con número y display millares
delay (3000);					//espera 1ms
	}
}
```

7.3 LCD formato 4 bits

El LCD disponible puede trabajar en formato 4 bits o en formato 8 bits. Hasta ahora por la limitación de puertos disponibles se ha usado en formato 4 bits. Consta de las siguientes funciones: *void EnviarInstruccion (char instrucc)* enviará al LCD en formato 4 bits (enviar nibble alto-enviar nibble bajo) una instrucción cuyo código se pasa como parámetro de entrada; *void EnviarDato (char dato)* enviará en formato de 4 bits un carácter cuyo código ASCII se pasa como parámetro de entrada. Estas dos funciones, a su vez, deberán llamar a una función de retardo de 5 ms. El estado de las líneas deberá ser en cada caso, RS (*Register Select*: RS=0 envía instrucción; RS=1, envía dato) y E (cuando E cambia de 1 a 0, ejecuta/visualiza).

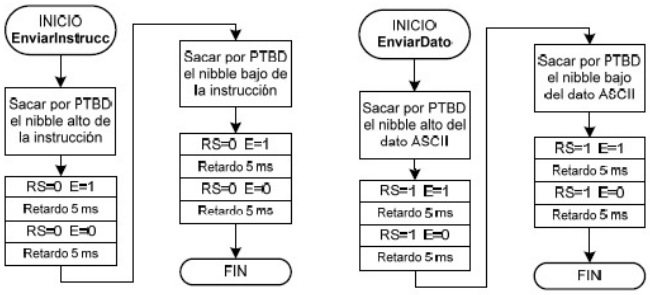


Figura 61. Diagrama de flujo funciones EnviarInstruccion y EnviarDato.

Del mismo modo, hay que inicializar la LCD en formato 4 bits, mediante la función LCDmodo_4bits.

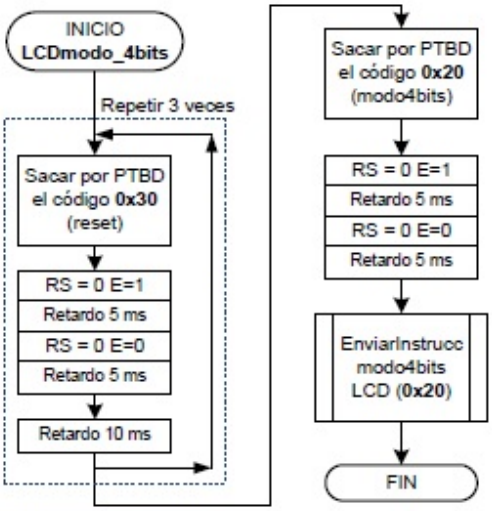


Figura 62. Diagrama de flujo función LCDmodo_4bits.

La función retardo será:

```
int delay (int n) {
              int i;
              for (i=0; i<n; i++);
       }
```

La función EnviarInstrucción es de la forma:

```
void EnviarInstruccion (char instrucc) {

        char auxiliar;
        auxiliar=instrucc;

        FGPIOC_PDOR=auxiliar;              //pasar nibble alto de la instrucción
        FGPIOE_PDOR=0x08;                  //entrada de control RS=0 E=1
        delay (15000);                     //espera 5ms
        FGPIOE_PDOR=0x00;                  //entrada de control RS=0 E=0
        delay (15000);                     //espera 5ms
        FGPIOC_PDOR=auxiliar<<4;           //pasar nibble bajo de la instrucción
        FGPIOE_PDOR=0x08;                  //entrada de control RS=0 E=1
        delay (15000);                     //espera 5ms
        FGPIOE_PDOR=0x00;                  //entrada de control RS=0 E=0
        delay (15000);                     //espera 5ms
        }
```

La función EnviarDato es como sigue:

```
void EnviarDato (char dato) {

        char auxiliar;
        auxiliar=dato;

        FGPIOC_PDOR=auxiliar;              //pasar nibble alto de la instrucción
        FGPIOE_PDOR=0x0C;                  //entrada de control RS=1 E=1
        delay (15000);                     //espera 5ms
        FGPIOE_PDOR=0x04;                  //entrada de control RS=1 E=0
        delay (15000);                     //espera 5ms
        FGPIOC_PDOR=auxiliar<<4;           //pasar nibble bajo de la instrucción
        FGPIOE_PDOR=0x0C;                  //entrada de control RS=1 E=1
        delay (15000);                     //espera 5ms
        FGPIOE_PDOR=0x04;                  //entrada de control RS=1 E=0
        delay (15000);                     //espera 5ms
        }
```

La inicialización del LCDmodo4bits es de la forma:

```
void LCDmodo4bits() {

        int i;
        for (i=0; i<3; i++) {

                FGPIOC_PDOR=0x30;          //pasar instrucción de arranque
                FGPIOE_PDOR=0x08;          //entrada de control RS=0 E=1
                delay (15000);             //espera 5ms
                FGPIOE_PDOR=0x00;          //entrada de control RS=0 E=0
                delay (15000);             //espera 5ms
                delay (30000);             //espera 10ms
                }

        FGPIOC_PDOR=0x20;                  //pasar instrucción de arranque 4 bits
        FGPIOE_PDOR=0x08;                  //entrada de control RS=0 E=1
        delay (15000);                     //espera 5ms
        FGPIOE_PDOR=0x00;                  //entrada de control RS=0 E=0
        delay (15000);                     //espera 5ms
        EnviarInstruccion(0x20);           //llamada función
        }
```

7.4 LCD formato 8 bits

La opción de LCD en formato 8 bits facilita la programación, pero por el contrario es necesaria la existencia de un mayor número de puertos IO.

Las funciones necesarias se basan en las anteriores pero ahora se pasan los 8 bits a la vez, lo que facilita la legibilidad del código.

La nueva función EnviarInstruccion en formato 8 bits es de la siguiente manera:

```
void EnviarInstruccion (char instrucc) {

        char auxiliar;
        auxiliar=instrucc;

        FGPIOC_PDOR=auxiliar;              //pasar la instrucción
        FGPIOE_PDOR=0x08;                  //entrada de control RS=0 E=1
        delay (15000);                     //espera 5ms
        FGPIOE_PDOR=0x00;                  //entrada de control RS=0 E=0
        delay (15000);                     //espera 5ms
        }
```

La nueva función en formato 8 bits EnviarDato es de la forma:

```
void EnviarDato (char dato) {

        char auxiliar;
        auxiliar=dato;

        FGPIOC_PDOR=auxiliar;              //pasar el dato
        FGPIOE_PDOR=0x0C;                  //entrada de control RS=1 E=1
        delay (15000);                     //espera 5ms
        FGPIOE_PDOR=0x04;                  //entrada de control RS=1 E=0
        delay (15000);                     //espera 5ms
        }
```

La nueva inicialización en formato 8 bits es:

```
void LCDmodo8bits() {

        int i;
        for (i=0; i<3; i++) {

                FGPIOC_PDOR=0x30;          //pasar instrucción de arranque
                FGPIOE_PDOR=0x08;          //entrada de control RS=0 E=1
                delay (15000);             //espera 5ms
                FGPIOE_PDOR=0x00;          //entrada de control RS=0 E=0
                delay (15000);             //espera 5ms
                delay (30000);             //espera 10ms
                }
        }
```

7.5 Memoria EEPROM serie

Su estructura está organizada en palabras de 1 byte (8 bits) de longitud, por lo tanto dispondremos en total de 32x8 (256 kbits) para almacenar información.

El protocolo I2C fue desarrollado por la empresa Philips, ahora llamada NXP, por la necesidad en el mercado de establecer una interfaz que comunicara entre si varios dispositivos electrónicos y que ahorrara el máximo número de pines.

El bus de comunicación utiliza dos cables. Uno para transmitir los datos (SDA) y otro para la señal de reloj (SCL). El protocolo de comunicación es del tipo Maestro-Esclavo, el dispositivo que hace de maestro es el que gobierna la comunicación y es el que controla la señal de reloj (SCL). Los dispositivos que hacen de esclavos responden a las peticiones del maestro. Cada dispositivo conectado al bus se identifica por una única dirección.

El procedimiento de escribir en la memoria se realiza de la siguiente manera. En primer lugar, enviaremos el estado de START a la memoria. A continuación enviaremos el byte de control que siempre es '1010' como código del fabricante. Los bits del 3 al 1 contienen la dirección del dispositivo o memoria (A2, A1, A0). En este caso como las hemos puesto a tierra serán '000' y el bit 0 será el bit con el que nos encargaremos de decirle al dispositivo si vamos a leer o escribir en la memoria con un '1' o un '0' respectivamente.

Como queremos escribir este bit será '0'. Después de este byte la memoria nos enviará el estado ACK y continuaremos enviando la parte alta de la dirección en la que queremos escribir. Volveremos a recibir ACK y enviaremos la parte baja de la dirección de la memoria a la cual queremos enviar el dato. Seguiremos enviando el dato que queremos guardar en la memoria y finalmente otra vez ACK. Si después del último ACK no enviamos el estado de STOP seguiremos escribiendo en las direcciones siguientes hasta que enviemos el estado de STOP.

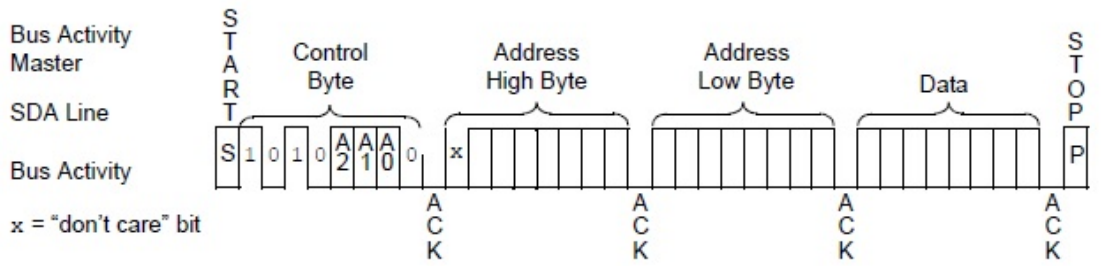


Figura 63. Escribir un byte en la memoria serie I2C.

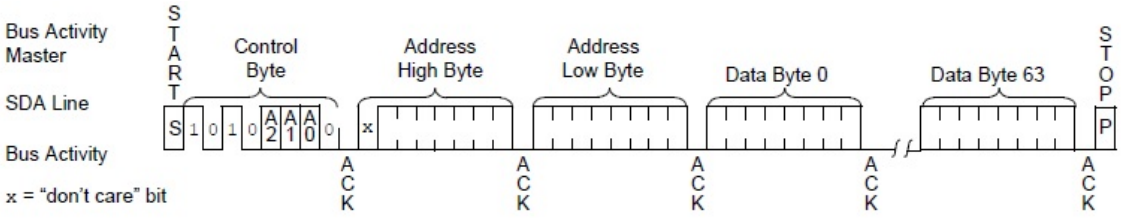


Figura 64. Escribir una página de bytes en la memoria serie I2C.

Para leer un dato de la memoria tendremos que seguir los siguientes pasos. Comenzaremos enviando el estado de START. A continuación el byte de control. Como queremos escribir la dirección de la que queremos leer el bit R/W será '0'. Recibiremos ACK tras enviar el byte de control y proseguiremos enviando la parte alta de la dirección. Volveremos a recibir ACK, continuamos con la parte baja de la dirección y otra vez ACK.

Después de escribir la dirección de la que queremos leer el dato de la memoria procederemos a comenzar desde el principio. Enviamos el estado de START y el byte de control pero esta vez el bit R/W lo pondremos a '1' porque queremos leer el dato de esa dirección. Recibiremos ACK y entonces podremos leer el dato de la dirección de la memoria.

Seguiremos leyendo los datos de las direcciones siguientes hasta que no le enviemos el estado de STOP.

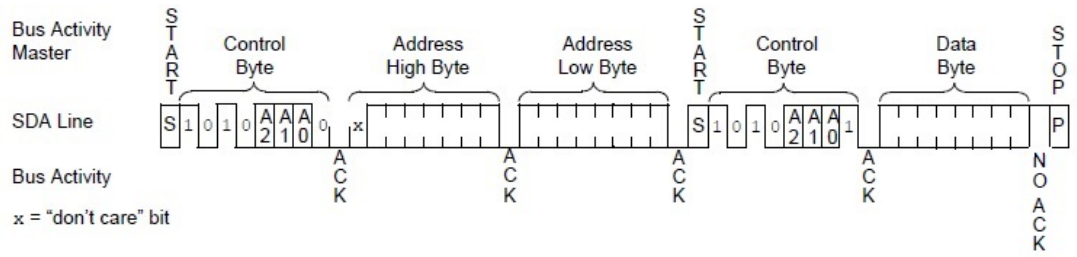


Figura 65. Leer un byte en la memoria serie I2C.

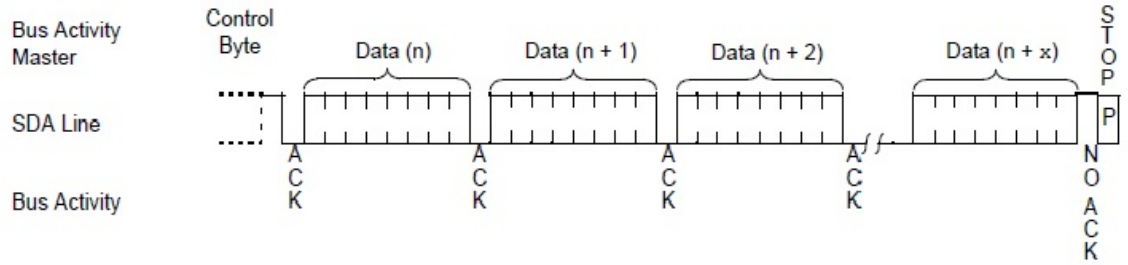


Figura 66. Leer una secuencia de bytes en la memoria serie I2C.

En primer lugar comenzaremos por configurar el bloque I2C. Continuaremos con la configuración de la frecuencia con la que operaremos. La máxima frecuencia a la que puede actuar la memoria es a 400KHz por lo que tiene que ser inferior. Por no estar al límite trabajaremos a 1KHz.

Continuaremos con la función *EscribirMemoria*. Esta función necesita que se pasen como parámetros de entrada las variables dirección y dato. Para escribir en la memoria el microcontrolador tiene que actuar como maestro.

Proseguiremos enviando el byte del código de control. Tenemos que enviar el estado START, código de seguridad, la dirección del dispositivo y el bit R/W, en este caso un '0', '10100000'. Esperaremos a que la trasmisión se haya producido. Para saber si se ha trasmitido el dato observaremos el flag TFC del registro I2C0_S. Si está en '1' significará que ha sido trasmitido completamente y si está a '0' significa que está aún trasmitiendo.

A continuación preguntaremos si hemos recibido el estado ACK de la memoria. En el caso de no haberlo recibido procederemos a enviar el evento STOP y volver otra vez a empezar. En cambio sí hemos recibido el estado ACK se enviará la parte alta de la dirección. Desplazando

ocho veces los bits hacia la derecha. Una vez realizado, esperaremos a que se produzca la trasmisión y continuaremos enviando la parte baja de la dirección.

Después de esto último volveremos a esperar a que se realice la transmisión. A continuación enviaremos el dato que queremos guardar y volveremos a esperar a que se realice la trasmisión. Para acabar esta función procederemos a enviar el evento STOP.

La función EscribirMemoria será de la forma:

```
void EscribirMemoria (int direccion, char dato) {

	int aux_direccion;
	char aux_dato;

	aux_direccion = direccion;
	aux_dato = dato;

	I2C1_C1 |= I2C_C1_TX_MASK;                //habilitar transmisión
	I2C1_C1 |= I2C_C1_MST_MASK;               //habilitar master, enviar START

	I2C1_D = 0XA0;                            //enviar byte control (10100000)
	while(!(I2C1_S & I2C_S_TCF_MASK));        //esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    //comprobar si recepcionado ACK
	{
		I2C1_C1 |= I2C_C1_TXAK_MASK;      //reset flag ACK
		I2C1_D = aux_direccion;           //enviar 8 primeros bits dirección
		while(!(I2C1_S & I2C_S_TCF_MASK)); //esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    //comprobar si recepcionado ACK
	{
		I2C1_C1 |= I2C_C1_TXAK_MASK;      //reset flag ACK
		I2C1_D = aux_direccion<<8;        //enviar segundos 8 bits dirección
		while(!(I2C1_S & I2C_S_TCF_MASK)); //esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    //comprobar si recepcionado ACK
	{
		I2C1_C1 |= I2C_C1_TXAK_MASK;      //reset flag ACK
		I2C1_D = aux_dato;                //enviar dato
		while(!(I2C1_S & I2C_S_TCF_MASK)); //esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    //comprobar si recepcionado ACK
	{
		I2C0_C1 |= I2C_C1_TXAK_MASK;      //reset flag ACK
		I2C0_C1 |= (~I2C_C1_MST_MASK);    //deshabilitar master, enviar STOP
	}
	}
	}
	}

}
```

La función *LeerMemoria* necesita como parámetro de entrada la dirección y devuelve el dato. Como en el caso anterior, tenemos que enviar el byte del código de control con el bit R/W a '0'. Como hemos realizado en la función anterior, esperaremos a que se haya terminado la trasmisión y proseguiremos enviando la parte alta de la dirección seguido de la parte baja de la dirección.

Una vez terminado de enviar la dirección completa repetiremos el código de control pero esta vez el bit R/W estará a '1' de tal manera que así le indicamos a la memoria que queremos leer el dato existente en la dirección de memoria anteriormente enviada.

Cambiaremos el modo de trasmitir al modo recibir para poder obtener el dato de la memoria. Realizamos una lectura en vacío para comenzar la recepción. Después de acabar la recepción de la lectura en vacío, la memoria necesita que le enviemos el estado ACK.

Continuaremos volviendo a leer la memoria y esta vez sí que es el dato el que guardamos en la variable dato. Para acabar la recepción de datos tendremos que enviar el evento STOP. Se ha finalizado la recepción de datos, la función devuelve el dato con return (dato).

La función LeerMemoria será:

```
void LeerMemoria (int direccion) {

	int aux_direccion;
	char aux_dato;

	aux_direccion = direccion;

	I2C1_C1 |= I2C_C1_TX_MASK;                //habilitar transmisión
	I2C1_C1 |= I2C_C1_MST_MASK;               //habilitar master, enviar START

	I2C1_D = 0XA0;                       //enviar byte control (10100000) escribir
	while(!(I2C1_S & I2C_S_TCF_MASK));        //esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    //comprobar si recepcionado ACK
	{
		I2C1_C1 |= I2C_C1_TXAK_MASK;        //reset flag ACK
		I2C1_D = aux_direccion;             //enviar 8 primeros bits dirección
		while(!(I2C1_S & I2C_S_TCF_MASK));  //esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK ) == 0X00)   //comprobar si recepcionado ACK
	{
		I2C1_C1 |= I2C_C1_TXAK_MASK;        //reset flag ACK
		I2C1_D = aux_direccion<<8;          //enviar segundos 8 bits dirección
		while(!(I2C1_S & I2C_S_TCF_MASK));  //esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    //comprobar si recepcionado ACK
	{
		I2C1_C1 |= I2C_C1_TXAK_MASK;        //reset flag ACK
		I2C1_C1 |= I2C_C1_MST_MASK;         //habilitar master, enviar START

		I2C1_D = 0XA1;               //enviar byte control (10100001) leer
		while(!(I2C1_S & I2C_S_TCF_MASK));  //esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    //comprobar si recepcionado ACK
	{
		I2C1_C1|=I2C_C1_TXAK_MASK;          //reset flag ACK
		while(!(I2C1_S & I2C_S_TCF_MASK));  //esperar transferencia finalizada
		aux_dato = I2C1_D;                  //leer dato de la memoria
		I2C1_C1 |= (~I2C_C1_MST_MASK);      //deshabilitar master, enviar STOP
	}
	}
	}
	}
```

8. Conclusiones y líneas de trabajo futuras

Este trabajo representa el primer paso hacia la obtención de un Sistema de Desarrollo para la realización de las prácticas de SEP.

Se han cumplido los objetivos marcados al inicio de este PFC. Una vez realizado el estudio comparativo de las familias de microcontroladores basadas en la arquitectura ARM Cortex M0+, se ha procedido a elegir la placa DEMO Freescale FRDM-KL25Z.

A partir de los componentes seleccionados, se ha realizado un primer prototipo plenamente funcional. Desarrollando y testeando tanto la placa de periféricos realizada como la plataforma seleccionada.

Del mismo modo, existe una documentación que sirve como ayuda para el alumno, y facilita la utilización del Sistema de Desarrollo y de la placa DEMO Freescale FRDM-KL25Z.

Se han incluido varias librerías sobre CodeWarrior Development Studio para este Sistema de Desarrollo.

Una vez realizado el prototipo, en su fase de puesta a punto, ya se ha comprobado la necesidad de mejorar diversas cuestiones que han ido surgiendo.

La primera de ellas se refiere a la propia PCB y a la distribución de las pistas, incluyendo el conector IO de la actual tarjeta QG8 utilizada en prácticas. Siguiendo con otros dispositivos que se puedan implementar en la siguiente fase

Se pueden incluir además los siguientes elementos, desde un giroscopio de 3 ejes y un magnetómetro que junto al acelerómetro ya disponible proporcionen los sensores que actualmente poseen los móviles de última generación. Posibilidad de conectividad inalámbrica.

Un display gráfico de puntos, interface paralelo de 8 bits, SPI, que puede resultar ideal para implementar en instrumentos de medición, equipos industriales, electrodomésticos o electrónica de consumo.

Una LCD directa sin controlador, donde estén disponibles cada uno de los segmentos de los tres displays existentes para su conexión directa al microcontrolador.

Una memoria RAM serie, que pueda ampliar la memoria SRAM que dispone un microcontrolador.

La idea final es la migración de las prácticas de SEP a un microcontrolador Cortex M0+ de 32 bits. En un futuro, se quiere fabricar la PCB del Sistema de Desarrollo en modo profesional, con taladro metalizado y acabado con máscara y serigrafía. Además, dicha placa debe ser un instrumento que el alumno utilice en las prácticas del área de Tecnología Electrónica.

A nivel personal, la realización de este trabajo ha supuesto añadir a los conocimientos ya adquiridos durante estos años en la Universidad, una formación complementaria donde cabe destacar la experiencia adquirida al enfrentarme con los problemas surgidos al desarrollar un prototipo muy específico.

Los principales problemas surgieron en la instalación y puesta a punto del sistema de desarrollo y evaluación Freescale FRDM-KL25Z.

El diseño, la realización del PCB mediante Eagle, y toda la gestión realizada de búsqueda, elección y compra de los componentes que se implementan en el sistema de desarrollo, no llevo ningún problema.

El trabajo menos gratificante fue todo el estudio de mercado necesario para la elección del Sistema de Desarrollo y Evaluación utilizado en el proyecto (Freescale FRDM-KL25Z).

Del mismo modo, la realización de las funciones específicas, mediante CodeWarrior Development Studio, que posibilitan la comunicación entre el sistema de desarrollo FRDM-KL25Z y la PCB han resultado relativamente asequibles, una vez estudiado a fondo el material disponible.

Por último, este trabajo debe servir para facilitar la adquisición de conocimientos prácticos a los futuros alumnos del grado de Ingeniería Electrónica y Automática.

9. Bibliografía

- ***Microcontroladores tendencias.***
  Oscar J. Herrero, Francisco Juarez.
  Arrow Seminario EINA 21/Mayo/2014.

- ***Microcontroladores de 32 bits ARM… o como no temerle al cambio.***
  Marcelo E. Romero, Eduardo Martínez.
  Universidad de Belgrano, Buenos Aires, Argentina.

- ***Comparativa de CPUs de 32 bits.***
  Ignacio J. Zaradnik.
  SASE 2011 Electrocomponentes SA.

- ***Sistemas basados en microcontroladores con fines docentes.***
  Javier Casado Zardoya.
  PFC Electrónica. Dtor. Antonio Bono Nuez.

- ***Design Center.***
  FARNELL element14.
  http://www.element14.com/community/community/designcenter

- ***FARNELL element14.***
  http://es.farnell.com/

- ***RS-Amidata.***
  http://es.rs-online.com/web/

- ***Digi-Key.***
  http://www.digikey.es/

- ***Freescale Semiconductor.***
  http://www.freescale.com/

- ***FRDM-KL25Z User's Manual.***
  Freescale Semiconductor.
  http://www.freescale.com/FRDM-KL25Z

- ***FRDM-KL25Z Quick Start Guide.***
  Freescale Semiconductor.
  http://www.freescale.com/FRDM-KL25Z

- ***FRDM-KL25Z Pinouts.***
  Freescale Semiconductor.
  http://www.freescale.com/FRDM-KL25Z

- ***OpenSDA User's Guide.***
  Freescale Semiconductor.
  http://www.freescale.com/FRDM-KL25Z

- ***KL25 Sub-Family Reference Manual.***
  Freescale Semiconductor.
  http://cache.freescale.com/files/32bit/doc/ref_manual/KL25P80M48SF0RM.pdf
- ***Kinetis KL25 Sub-Family:***
  48 MHz Cortex-M0+ Based Microcontroller with USB.
  Freescale Semiconductor.
  http://cache.freescale.com/files/32bit/doc/data_sheet/KL25P80M48SF0.pdf

- ***KL25 Sub-Family Data Sheet.***
  Freescale Semiconductor.
  http://www.element14.com/community/docs/DOC-50512/l/freescale-kinetis-kl25-sub-family-data-sheet

- ***Kinetis L Peripheral Module Quick Reference.***
  Freescale Semiconductor.
  http://cache.freescale.com/files/32bit/doc/quick_ref_guide/KLQRUG.pdf

- ***CodeWarrior Development Studio for Microcontrollers V10.x Quick Start.***
  Freescale Semiconductor.
  http://cache.freescale.com/files/soft_dev_tools/doc/quick_ref_guide/MCU_QS.pdf?fsrch=1

- ***CodeWarrior Development Studio for Micros V10.x. Profiling and Analysis Quick Start***.
  Freescale Semiconductor.
  http://cache.freescale.com/files/soft_dev_tools/doc/user_guide/PROFILING_AND_ANALYSIS_MCU_QS.pdf

- ***CodeWarrior Development Studio for Microcontrollers V10.x Targeting Manual.***
  Freescale Semiconductor.
  http://cache.freescale.com/files/soft_dev_tools/doc/user_guide/CWMCUDBGUG.pdf

- ***CodeWarrior Development Studio Common Features Guide.***
  Freescale Semiconductor.
  http://cache.freescale.com/files/soft_dev_tools/doc/user_guide/CWCFUG.pdf

Anexos

A. ANEXOS

A.1 Documento Ayuda para la Plataforma Freescale FRDM-KL25Z

Se incluyen en este anexo los documentos que se pueden proporcionar al alumno para la realización de las prácticas de SEP.

- Instalación Codewarrior Development Studio (v10.6_SE).
- Instalación Plataforma Freescale FRDM-KL25Z.
- Como crear un proyecto, trabajar con CodeWarrior.

A.1.1 Instalación Codewarrior Development Studio (v10.6_SE)

CodeWarrior Development Studio (v10.6_SE, **http://www.freescale.com**) es un potente entorno de desarrollo integrado (IDE, Integrated Development Environment) para los microprocesadores de Freescale (arquitecturas del tipo Coldfire, Coldfire+, DSC, Kinetis, MPC5xxx, RS08, S08 y S12Z), incluyendo los últimos dispositivos y dando soporte a las nuevas versiones.

CodeWarrior está basado en la plataforma Eclipse IDE 4.2.1 (Juno), consta de múltiples módulos: editor, compilador de ensamblador, compilador en lenguaje C/C++, simulador y depurador de hardware real, programación mediante flash, componentes y software Process Expert, interface OpenSDA, Open Source BDM, Open Source JTAG, CodeWarrior USB TAP, Cyclone MAX (P&E Microcomputer Systems), USB BDM Multilink (P&E Microcomputer Systems), USB Multilink Universal [FX] (P&E Microcomputer Systems).

Se va a instalar la versión v10.6_SE, para la familia Kinetis L (plataforma FRDM-KL25Z). Existen distintas ediciones de CodeWarrior, desde la demo hasta la profesional. Se ha elegido, dentro de las diferentes versiones, la versión “Special Edition”, que incluye gratuitamente prácticamente todas las funcionalidades, limitando el código C a 64KB.

Los requerimientos del hardware del sistema son: 1,8GHz Pentium compatible como mínimo, 2GB RAM, 20GB de espacio libre en el disco duro, DVD drive, puerto USB, conexión a internet. Microsoft Windows XP, Windows 7 o superior, 32 bits o 64 bits.

Como primer paso, hacer doble click en el fichero ejecutable de la aplicación.

Comenzar la instalación de Codewarrior Development Studio v10.6_SE y aceptar los términos de la licencia.

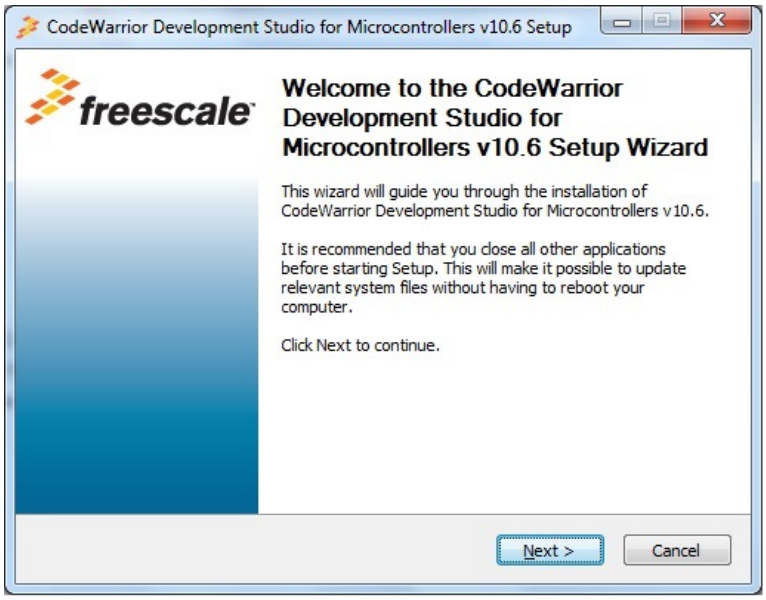


Figura 1. Página de bienvenida a la instalación.

Seleccionar los componentes que queremos instalar, en nuestro caso, solamente seleccionaremos Kinetis, aunque se pueden instalar un mayor número de familias.

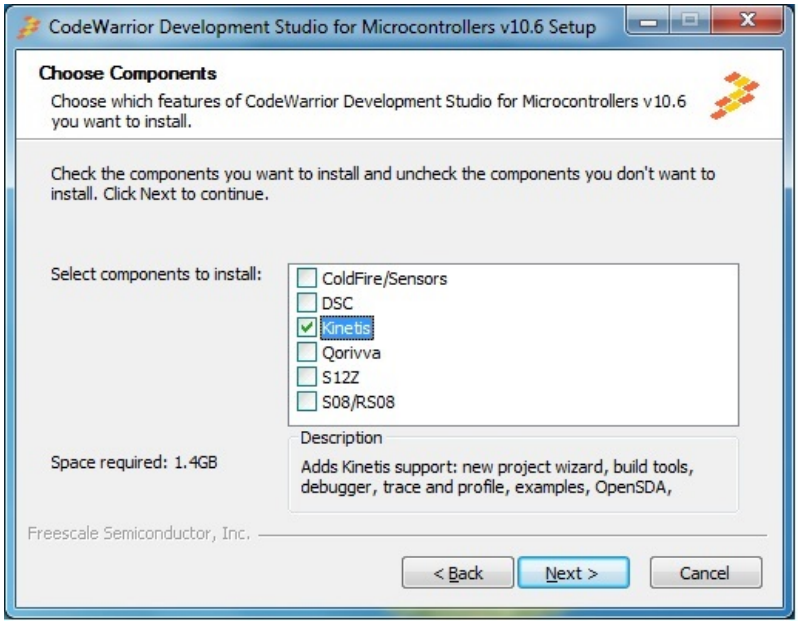


Figura 2. Selección de los componentes a instalar.

A continuación, selecciona la carpeta donde se va a proceder a instalar el programa, la que viene por defecto, u otra que se desee.

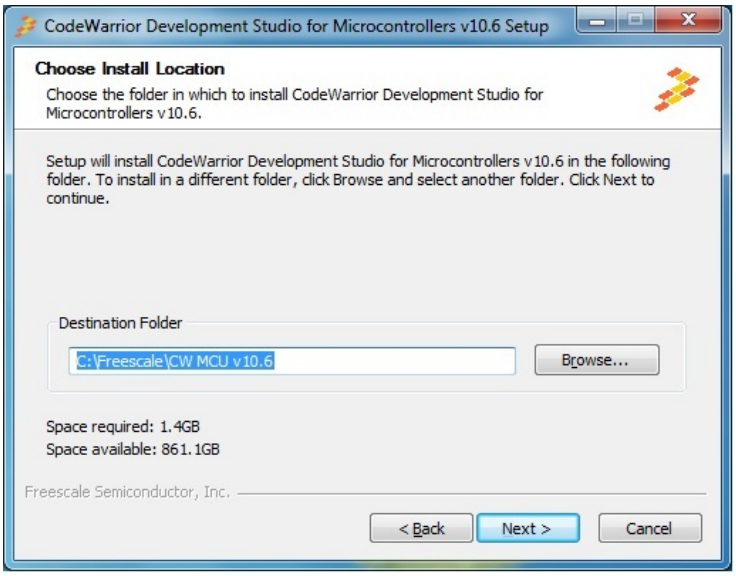


Figura 3. Carpeta donde se instala CodeWarrior.

El programa de instalación, se conecta a la página web de Freescale para descargar los componentes seleccionados previamente.

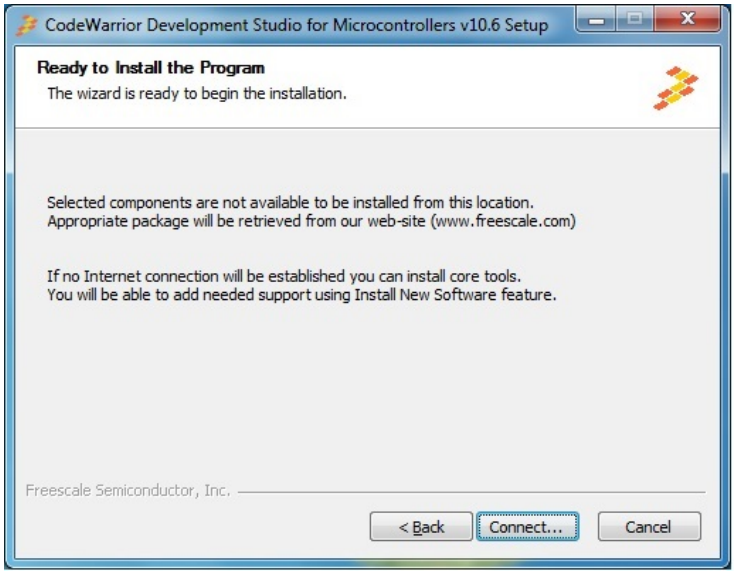


Figura 4. Inicio de la instalación de CodeWarrior.

Posteriormente, hay que instalar distintos software de dispositivos que van apareciendo durante la instalación.

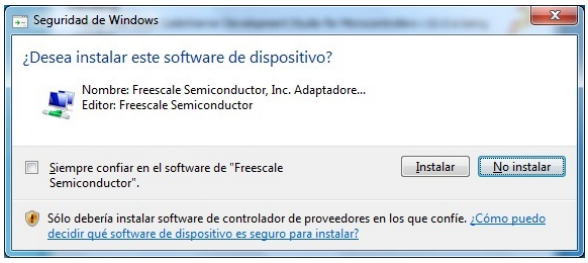


Figura 5. Pantalla tipo de instalación de software de dispositivo.

Se ha completado la instalación de CodeWarrior Development Studio v10.6_SE.

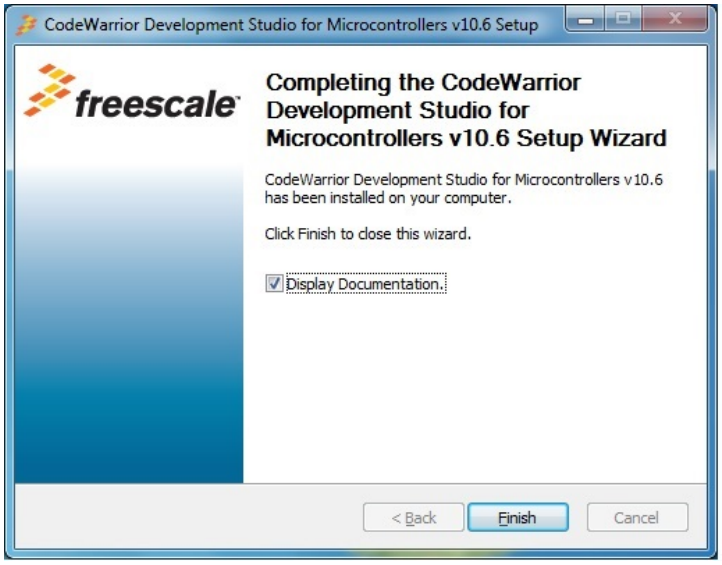


Figura 6. Fin de la instalación de CodeWarrior.

Una vez completada la instalación, aparece una ventana del navegador con la página web de Freescale, donde se pueden descargar diversos manuales entre ellos la guía de uso rápido en formato pdf (***FRDM-KL25Z Quick Start Guide***).

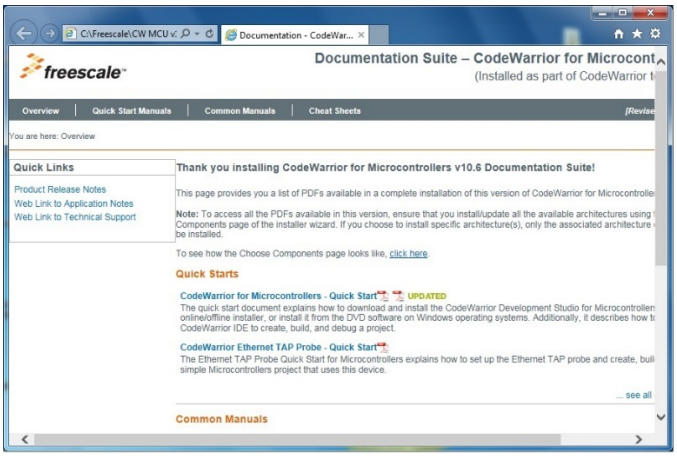


Figura 7. Página web de Freescale.

A.1.2 Instalación Plataforma Freescale FRDM-KL25Z

La plataforma de desarrollo Freescale Freedom es un conjunto de herramientas de hardware y software para la evaluación y desarrollo. Es ideal para prototipado rápido de aplicaciones basadas en microcontroladores.

La tarjeta Freescale Freedom KL25Z, modelo FRDM-KL25Z, es un diseño rentable y capaz con un microcontrolador Kinetis serie L, uno de los primeros microcontroladores de la industria construido sobre el núcleo ARM ® Cortex ™ M0+.

FRDM-KL25Z puede utilizarse para evaluar dispositivos de la serie Kinetis L. Cuenta con un KL25Z128VLK, un microcontrolador con una máxima frecuencia de operación de 48MHz, 128KB de memoria flash, un controlador USB, periféricos analógicos y digitales. La placa FRDM-KL25Z es compatible con el diseño de Arduino ™ R3, proporcionando una amplia gama de expansión. En la tarjeta se incluyen un LED RGB, un acelerómetro digital 3 ejes y una zona táctil capacitiva.

FRDM-KL25Z es la primera plataforma de hardware de Freescale con sistema abierto integrado estándar serie y adaptador OpenSDA. Este circuito ofrece varias opciones para comunicaciones series, programación flash y control de ejecución de depuración.

Existe una amplia lista de documentos de referencia de la placa de Freescale Freedom FRDM-KL25Z.

Todos estos documentos están disponibles en la página web de Freescale **http://www.freescale.com/FRDM-KL25Z.**

Las características de la tarjeta FRDM-KL25Z incluyen:

• MKL25Z128VLK4 en un encapsulado LQFP 80.

• Zona táctil capacitiva.

• Acelerómetro MMA8451Q.

• LED RGB.

• Varias opciones de alimentación: USB, batería de botón, fuente externa.

• Fácil acceso al conector de entradas/salidas, IO MCU compatible Arduino ™ R3.

• Inteface programable OpenSDA con múltiples aplicaciones disponibles, incluyendo:

Interfaz de programación flash del dispositivo de almacenamiento masivo.

Interfaz de depuración PE, proporciona control de ejecución y compatibilidad con herramientas IDE.

Interfaz de DAP CMSIS: nuevo estándar para la interfaz de depuración.

La figura 8 muestra un diagrama de bloques de la placa Freescale Freedom FRDM-KL25Z.

Los principales componentes y su ubicación en la placa se representan en la figura 9.

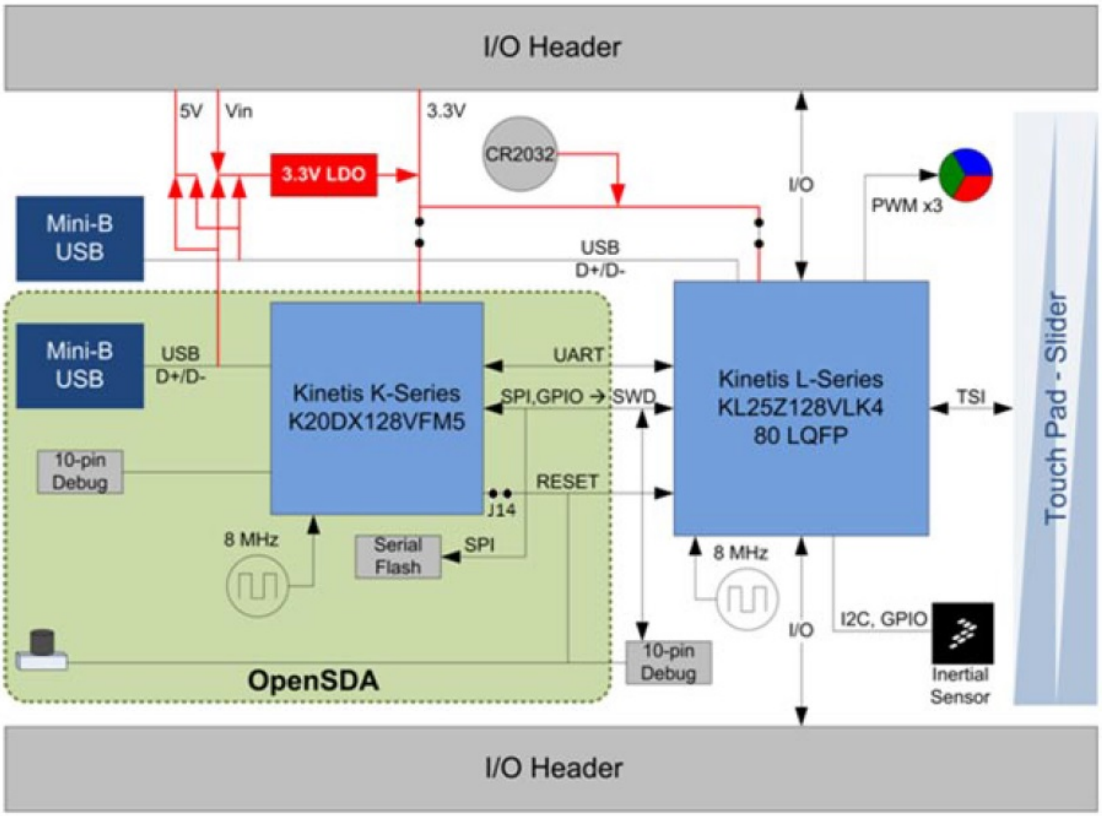


Figura 8. Diagrama de bloques de la placa FRDM-KL25Z de Freescale.

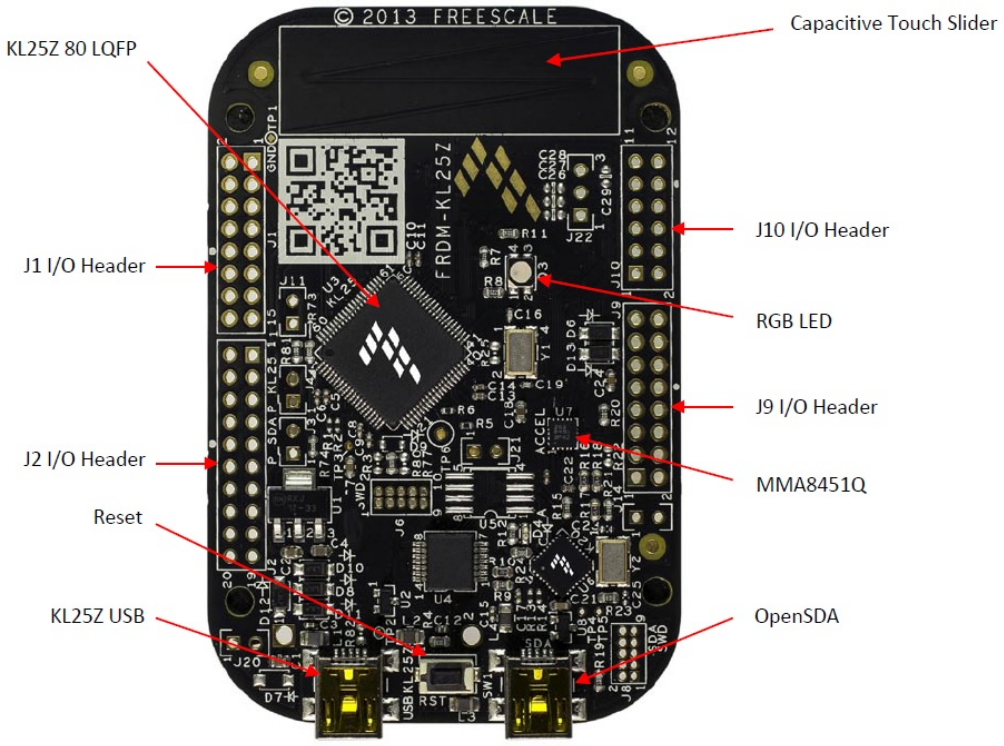


Figura 9. Principales componentes y su ubicación en la placa FRDM-KL25Z de Freescale.

El microcontrolador KL25Z128VLK4 está disponible en un encapsulado 80-pin LQFP. Unos pines se utilizan internamente en la placa de desarrollo, pero muchos están conectados directamente a una de los cuatro conectores de IO.

Los pines en el microcontrolador KL25Z llevan el nombre de su función de entrada-salida. Por ejemplo, el pin 1 en el puerto A se denomina PTA1. En su caso, los nombres de los contactos del conector de IO reciben el mismo nombre que el pin de KL25Z conectado a él.

Los conectores se denominan, J1 el primero de 16 pines a la izquierda, J2 el conector inferior de 20 pines, J9 el de 16 pines inferior derecha y por último J10 el de 12 pines.

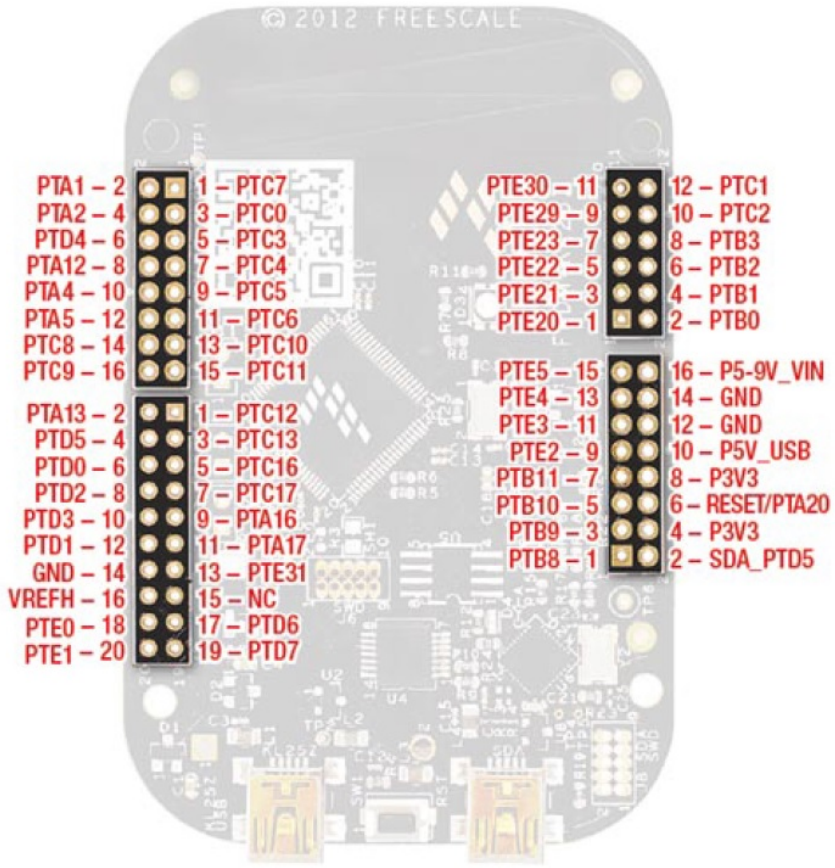


Figura 10. Conectores IO de la placa FRDM-KL25Z.

-

Los conectores de IO en el FRDM-KL25Z se han dispuesto para permitir la compatibilidad con placas de periféricos del microcontrolador Arduino. Las filas exteriores de pines (los pines pares) en los conectores comparten el mismo espacio mecánico y colocación con los conectores de IO en el Arduino revisión 3 (R3) estándar.

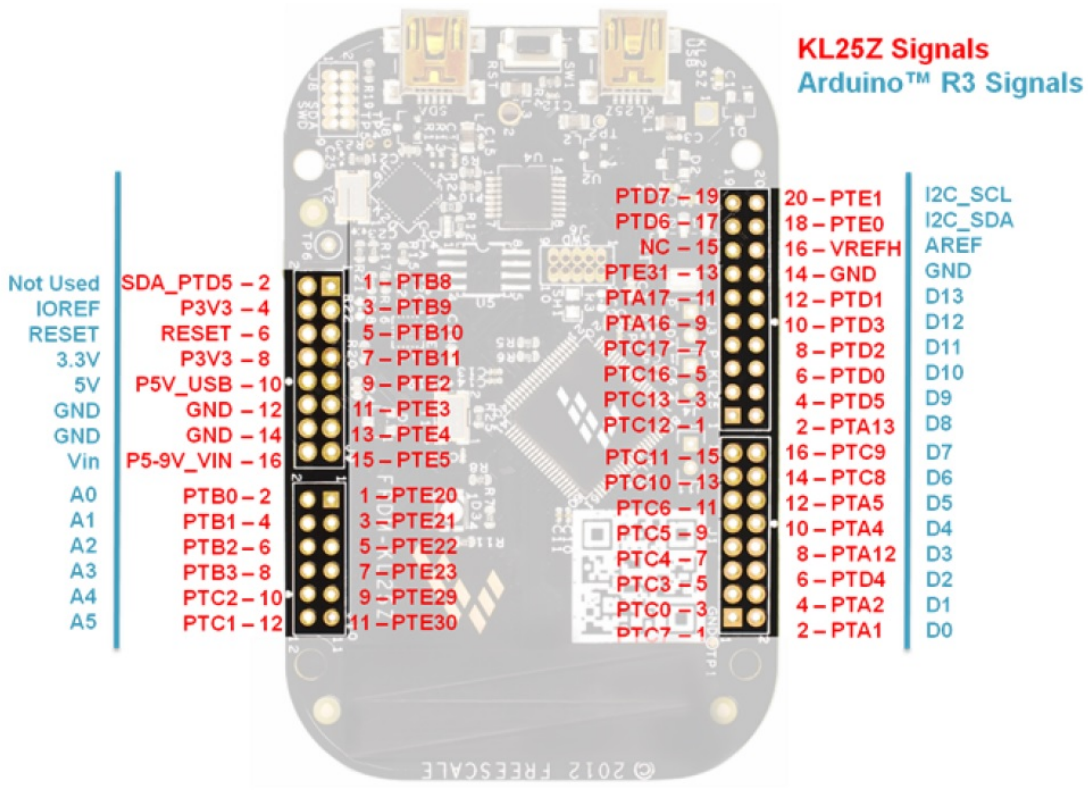


Figura 11. Conectores IO de la placa FRDM-KL25Z compatibles con Arduino.

El procedimiento de instalación de la Plataforma Freescale Freedom FRDM-KL25Z está descrito en ***FRDM-KL25Z Quick Start Guide***, un resumen de este documento es el siguiente:

Si la tarjeta es nueva, al conectarse el cable USB en el conector mini USB OpenSDA, se enciende un LED verde. En el ordenador, la placa es un dispositivo de almacenaje mass-storage device, un volumen llamado FRDM-KL25Z.

Tiene preinstalada una demo con la activación del LED RGB de la placa, el color del LED cambia mediante los datos que le proporciona el acelerómetro interno. Deslizando el dedo por la zona capacitiva de la placa podemos modificar el brillo del LED.

Una vez instalado CodeWarrior v10.6_SE, conectar el cable USB y desde propiedades de Equipo abrir el Administrador de dispositivos y comprobar que la placa se ha instalado correctamente. Para ello, en Jungo existe un dispositivo “PEMicro OpenSDA Debug Drive” y en los puertos (COM y LPT) aparece conectado un “OpenSDA-CDC Serial Port” en el puerto COM5 en este caso.

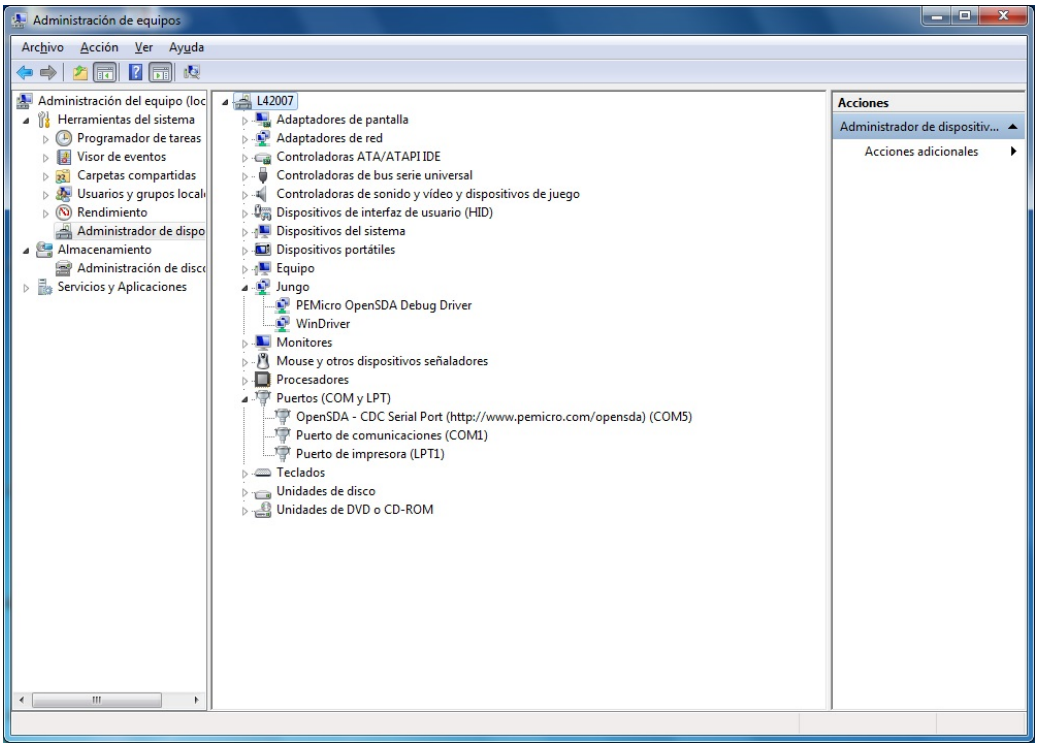


Figura 10. Administrador de dispositivos.

Si por cualquier problema no se ha instalado correctamente, descargar e instalar los drives "**P&E OpenSDA USB drivers**" disponibles en la página web http://www.pemicro.com/openSDA.

Para instalar este driver, una vez en el Administrador de dispositivos en OpenSDA Serial Port, con el botón derecho seleccionar "Actualizar software del controlador", "Buscar el driver en el equipo", el que acabamos de descargar y completar la instalación.

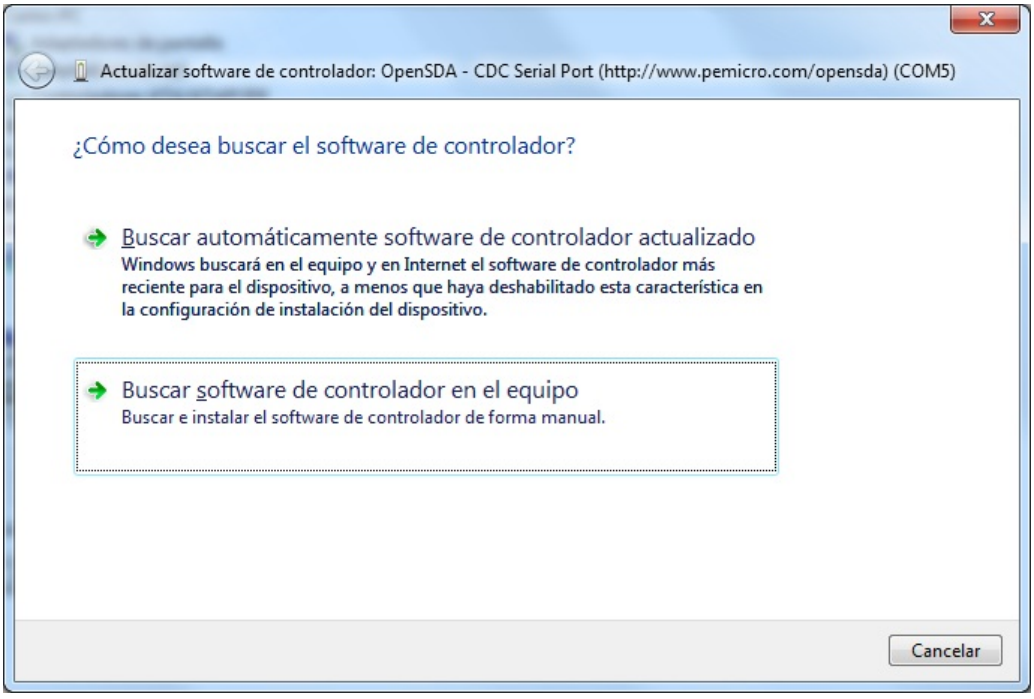


Figura 11. Actualizar software de dispositivo desde el equipo.

Introducción al modo OpenSDA

OpenSDA es una serie de estándares abiertos y un adaptador debuger. El circuito se basa en un microcontrolador (MCU) de Freescale Kinetis K20 con 128 KB de flash incorporado y un controlador USB integrado.

OpenSDA cuenta con un dispositivo de almacenamiento masivo (MSD) gestor de arranque, que proporciona un mecanismo rápido y fácil para cargar distintas aplicaciones OpenSDA como programadores flash, interfaces de control de ejecución de depuración, convertidores de serial a USB y más.

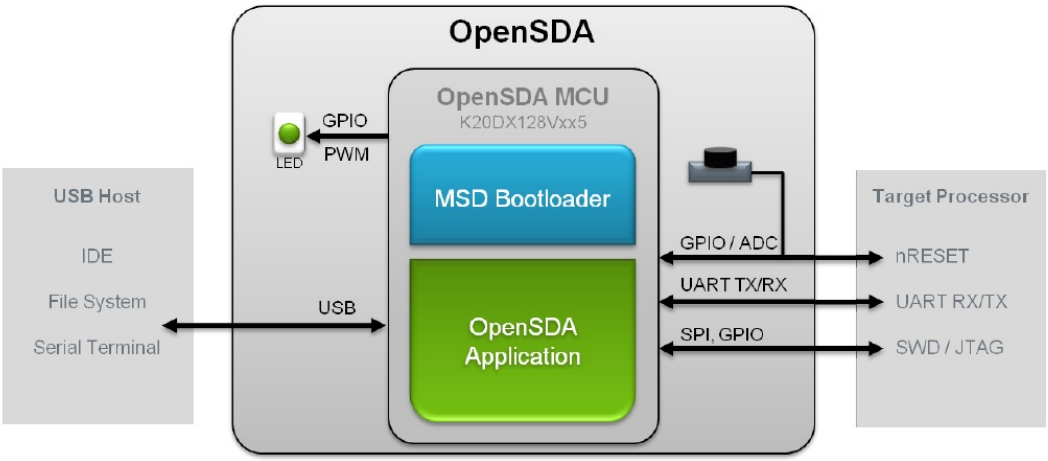


Figura 12. Diagrama de bloques OpenSDA.

Modo OpenSDA Bootloader

Para entrar en modo OpenSDA Bootloader, se realiza la siguiente operación:

- Desconectar el cable USB del equipo.
- Presionar y mantener pulsado el botón de Reset (SW1).
- Conectar el cable USB entre el ordenador y el conector mini USB OpenSDA de la tarjeta.
- Soltar el botón de Reset.

El LED de la placa parpadea cada 0,5 segundos, el dispositivo de almacenaje ahora tiene el nombre de BOOTLOADER.

Figura 13. Dispositivo de almacenaje Bootloader.

Si existe un problema al intentar cargar una aplicación en la tarjeta, puede aparecer el mensaje de que no encuentra el dispositivo OpenSDA, si esto ocurre verificar la versión Bootloader. Para ello, en modo Bootloader se ejecuta el fichero SDA_INFO.HTML.

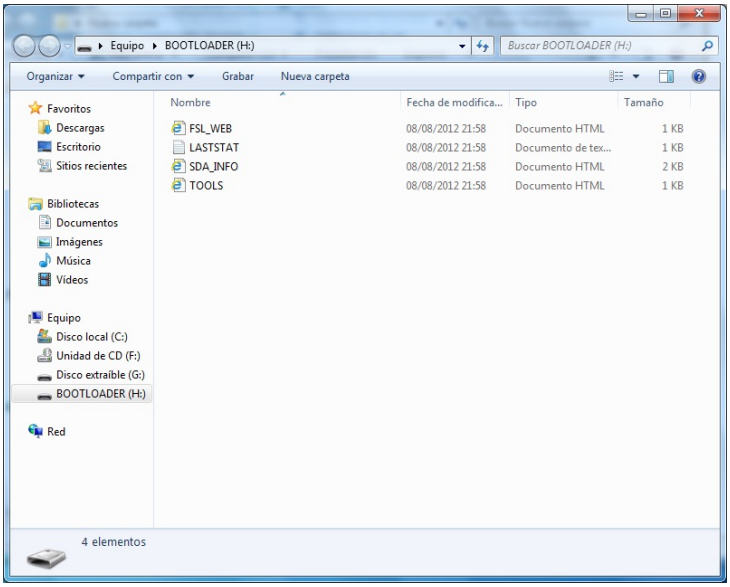


Figura 14. Fichero SDA_INFO.

Se abre una ventana del navegador donde se puede verificar la versión de Bootloader, la versión debe ser 1.11 o superior, si no es el caso descargar de http://www.pemicro.com/openSDA.

Figura 15. Comprobación versión Bootloader.

Cargar una aplicación OpenSDA

• Localizar las aplicaciones OpenSDA en la carpeta FRDM-KL25Z Quick Start Package.

• Copiar y pegar la aplicación en el drive Bootloader.

• Desconectar el cable para salir del modo Bootloader y conectarlo de nuevo.

La nueva aplicación OpenSDA está cargada en la plataforma FRDM-KL25Z.

Si es necesario instalar una nueva versión del MSD Flash Programmer, utiliza el mismo procedimiento que para cargar cualquier aplicación OpenSDA.

A.1.3 Como crear un proyecto, trabajar con CodeWarrior

CodeWarrior trabaja con proyectos, un proyecto es una carpeta en la que se van incluyendo todos los ficheros que se realizan, el programa, funciones, archivos para simulación, depuración, etc. Siempre existe un fichero de proyecto .cproject con la información relativa al proyecto, el programa principal será main.c. A modo de ejemplo vamos a crear un proyecto que hace parpadear el LED RGB incluido en la plataforma Freescale Freedom FRDM-KL25Z.

Al arrancar CodeWarrior aparece una pantalla que permite seleccionar la carpeta de trabajo donde CodeWarrior guarda los proyectos. Se puede dejar la carpeta que está por defecto u elegir otra que interese.

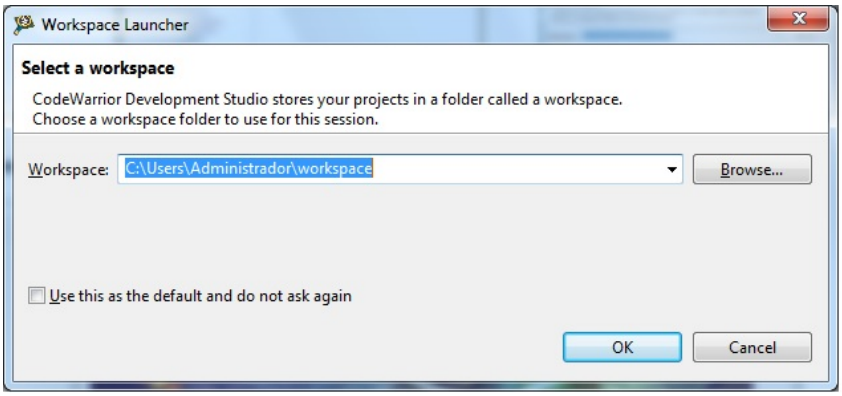


Figura 16. Carpeta donde CodeWarrior guarda los proyectos.

Aparece a continuación el menú de arranque, que nos permite cargar un ejemplo existente, crear un nuevo proyecto, descargar actualizaciones,…

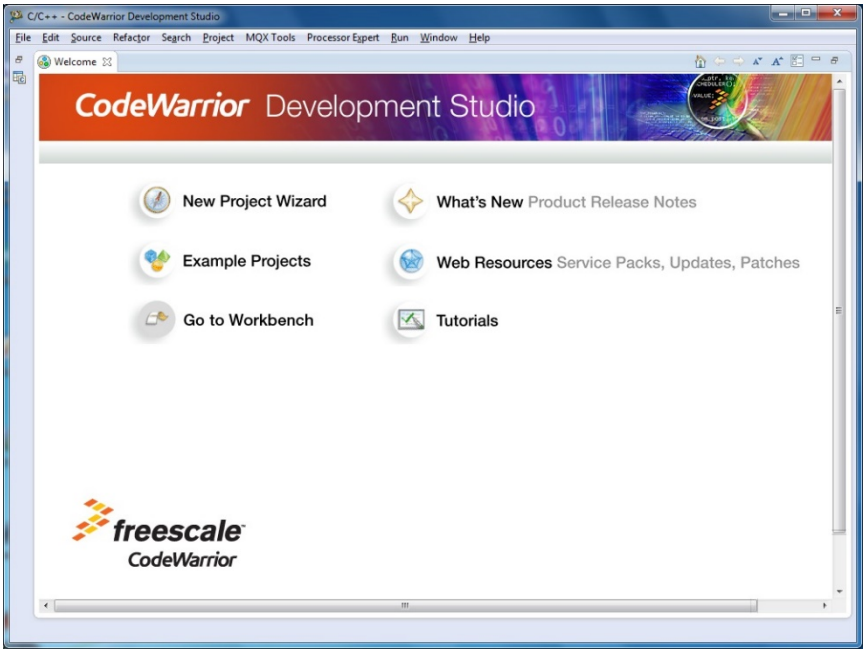


Figura 17. Menú de arranque de CodeWarrior.

Al crear un nuevo MCU Bareboard Project, el programa nos solicita el nombre del proyecto y su ubicación.

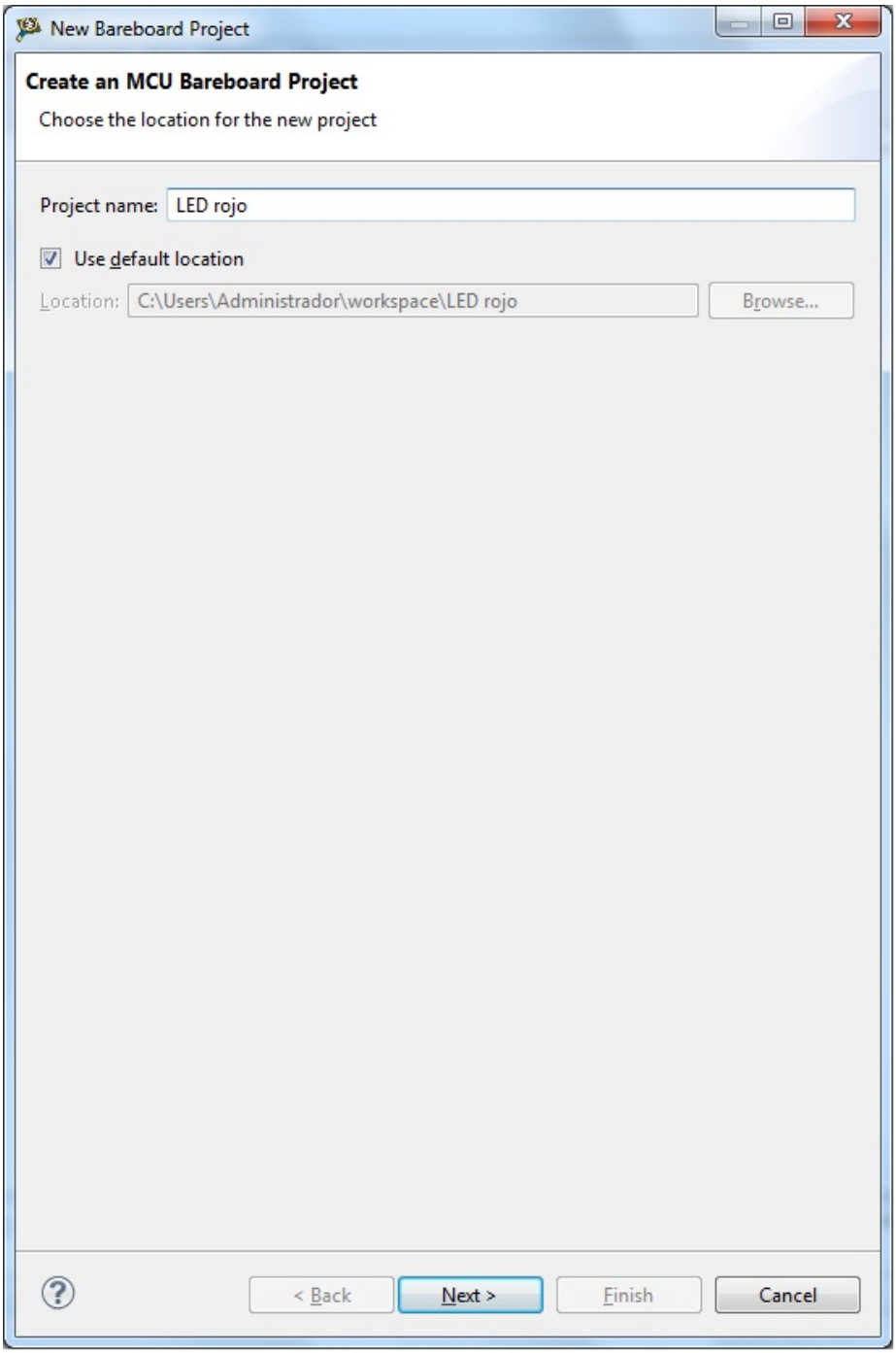


Figura 18. Crear un nuevo MCU Bareboard Project.

Una vez seleccionado el nombre del proyecto se continúa con la selección del dispositivo que se va a utilizar. En nuestro caso, se trata de un microcontrolador de la serie L de Kinetis, de la familia KL2x, el micro MKL25Z128.

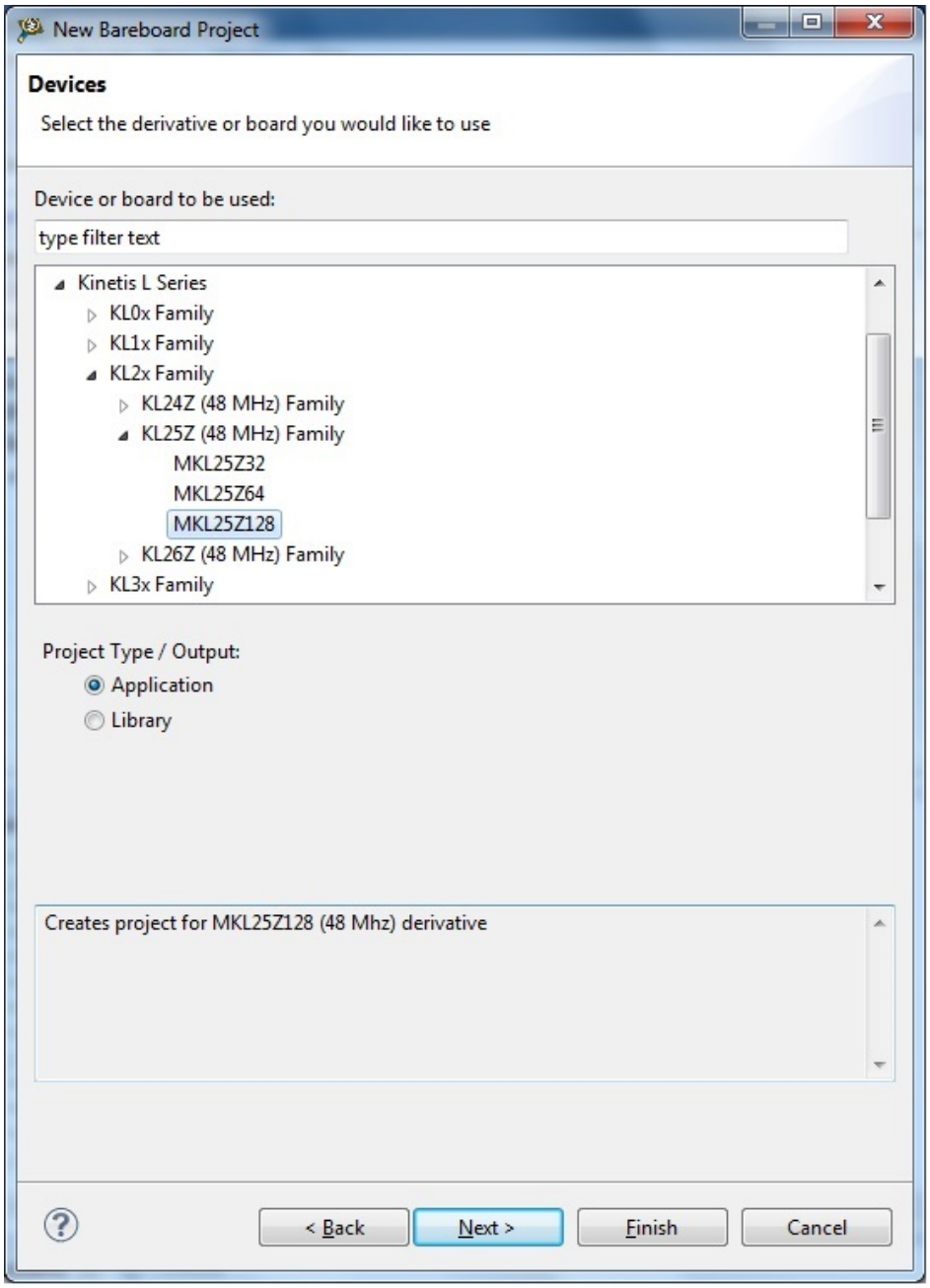


Figura 19. Selección del microcontrolador de Kinetis.

En el siguiente paso, nos pregunta qué tipo de conexión se va a usar entre CodeWarrior y la tarjeta de desarrollo y evaluación. Seleccionamos OpenSDA, en cualquier caso aunque seleccionemos más tipos de conexiones, en el momento de ejecutar la opción debugger el programa nos pregunta que conexión queremos utilizar y en ese momento se puede seleccionar OpenSDA.

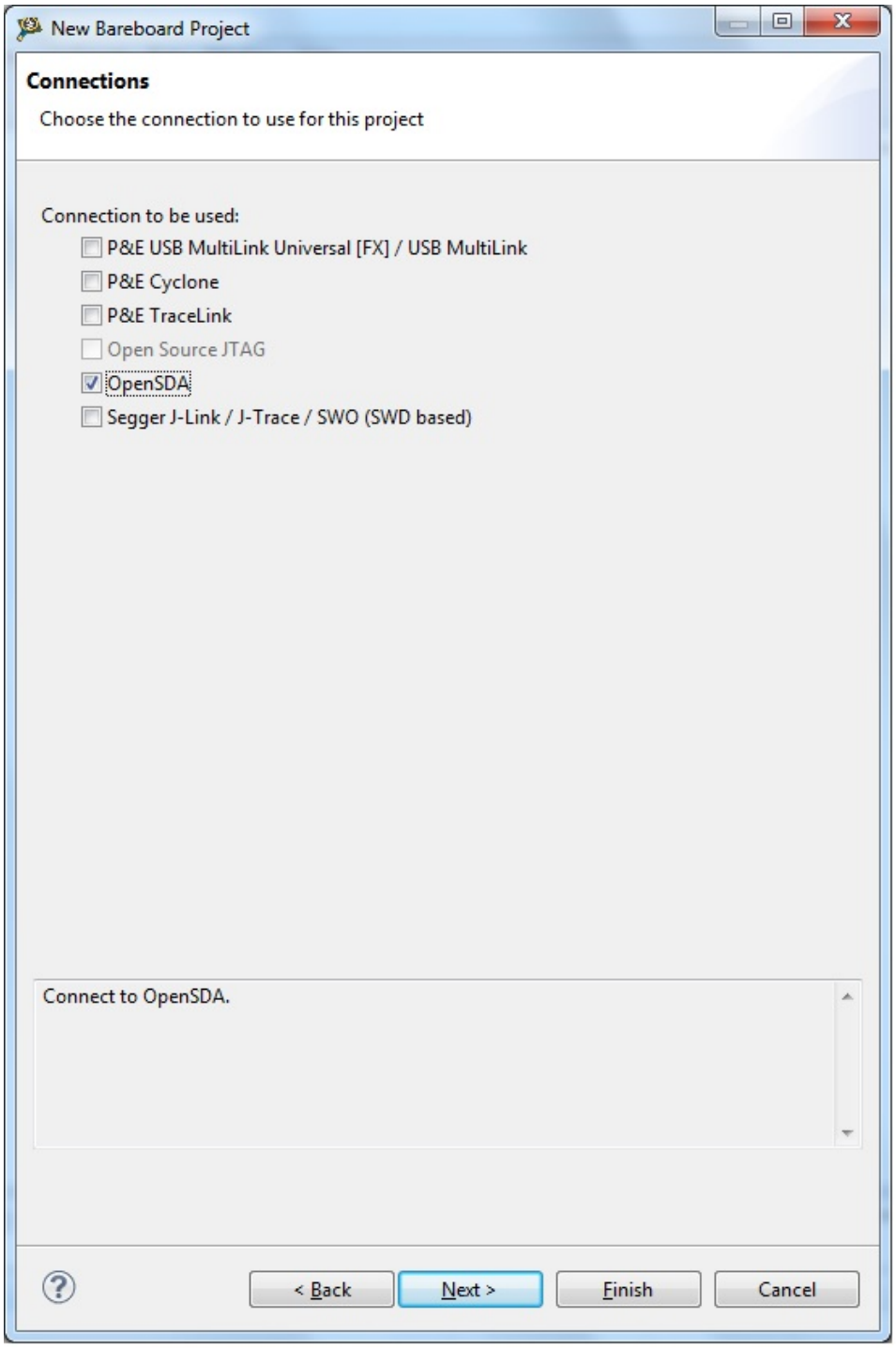


Figura 20. Selección de la conexión OpenSDA.

Posteriormente, la siguiente pantalla nos permite elegir entre lenguaje ensamblador, C y C++. Para nuestro caso, seleccionamos lenguaje C. Floating point en modo software por defecto, lo mismo que I/O Support UART. ARM Build Tools elegimos GCC. En cualquier caso depende de nuestro sistema de desarrollo y evaluación.

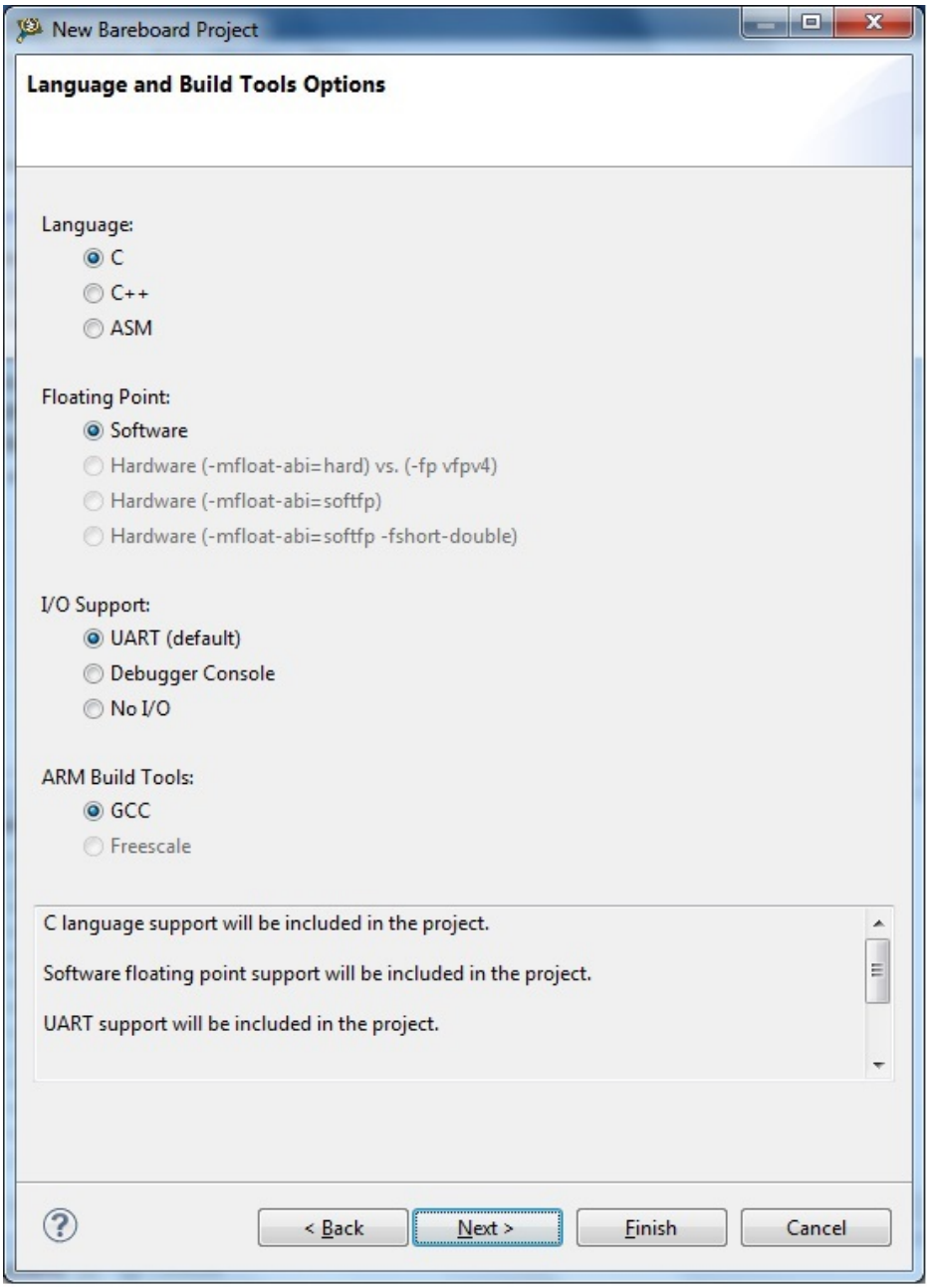


Figura 21. Lenguaje y herramientas de Build.

Por último, la ventana correspondiente a Proccesor Expert, si seleccionamos la opción disponible de Processor Expert se pueden configurar rápidamente dispositivos periféricos en el microcontrolador, o desde una biblioteca de módulos de código que pueden implementar varios servicios de dispositivos tales como temporizador, conversores,…

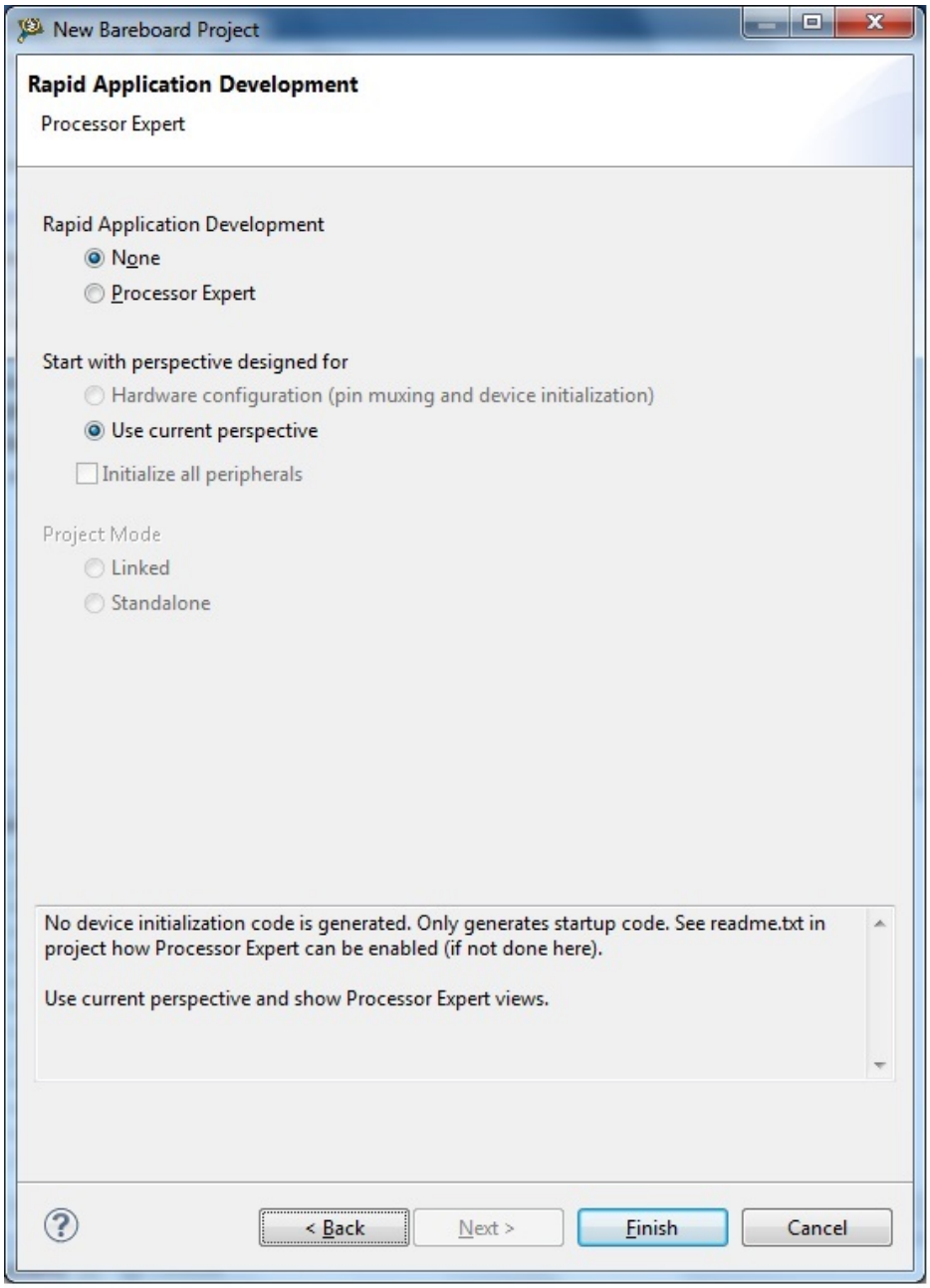


Figura 22. Opciones Processor Expert.

Una vez finalizadas todas las ventanas anteriores ya hemos creado un MCU Bareboard Project, se abre una pantalla, en el lado izquierdo hay un menú desplegable donde se pueden gestionar los proyectos que se encuentran en la carpeta workspace. Con el botón derecho abrimos este menú de gestión.

Justo debajo de este menú, existe una ventana de comandos, se puede activar desde la carpeta Project Settings del menú de proyectos. Nos va a permitir editar la configuración de debugger.

Siguiendo hacia la derecha en la pantalla, existe una pestaña donde podemos programar el fichero main.c, programa principal en lenguaje C. En la figura 23, en la ventana central "main.c" se muestra un ejemplo de programa en C de parpadeo de LEDS. Debajo de la anterior,

existe otra ventana que permite comprobar si el programa en C que estamos escribiendo no tiene errores.

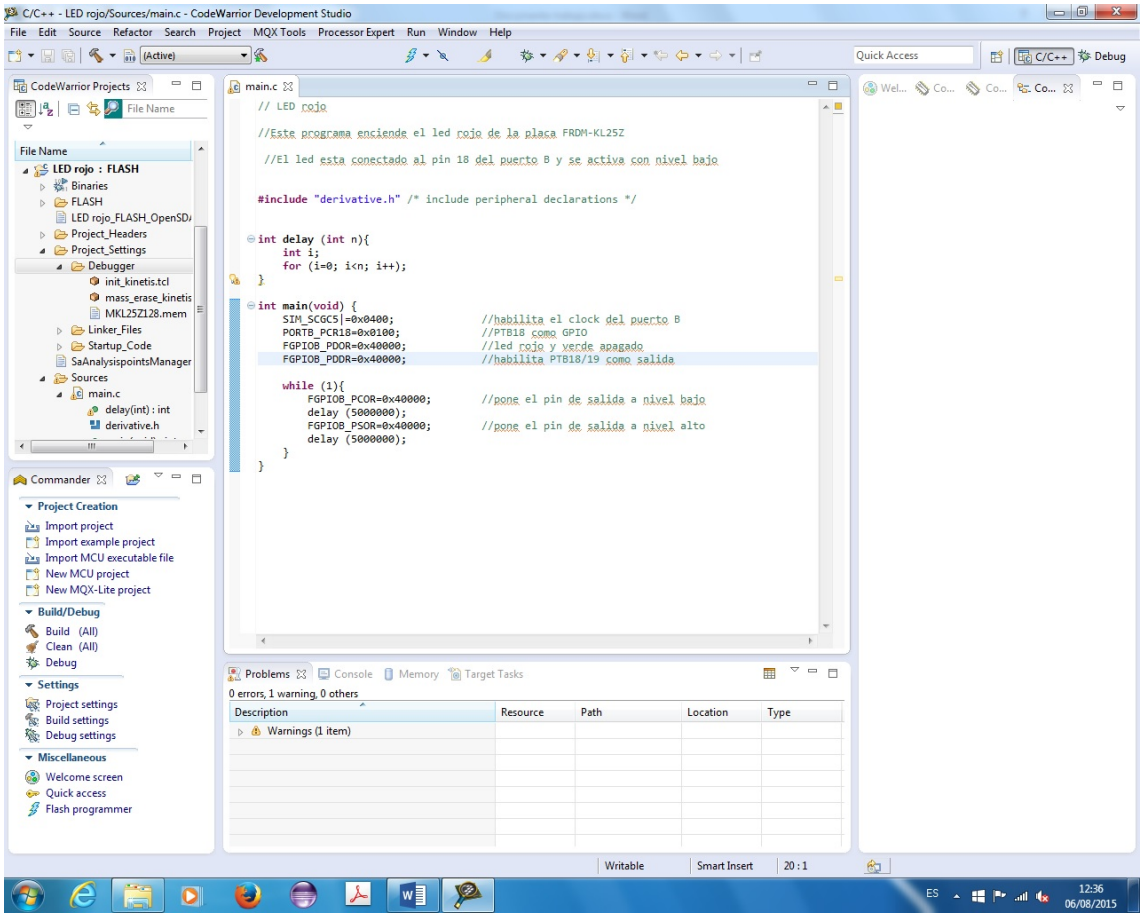
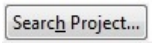


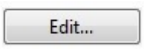
Figura 23. Programación en lenguaje C.

Una vez escrito el programa en la ventana central, podemos editar la configuración de debugger. Sería en este momento cuando el programa nos preguntaría cual es la conexión con la placa de desarrollo y evaluación que seleccionamos. Si solamente se ha elegido una el programa va directamente a la pantalla de configuración del debugger.

El programa ha creado una configuración de debugger con el nombre del proyecto y el tipo de conexión que hemos elegido (LED rojo_FLASH_OpenSDA).

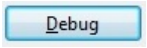
En la pestaña main, nos permite seleccionar el tipo de sesión de debugger (Download). Indica después que se trata de una aplicación creada mediante lenguaje C/C++. Te permite buscar el proyecto mediante el botón 

Indica el nombre del fichero que vamos a cargar dentro del sistema desarrollo y evaluación (FLASH/LEDrojo.elf). En este momento, se puede seleccionar otro fichero buscándolo en 

En Target settings se puede editar la 

conexión existente y crear una 

conexión nueva.

Si procedemos a hacer doble click  en ejecutamos la aplicación.

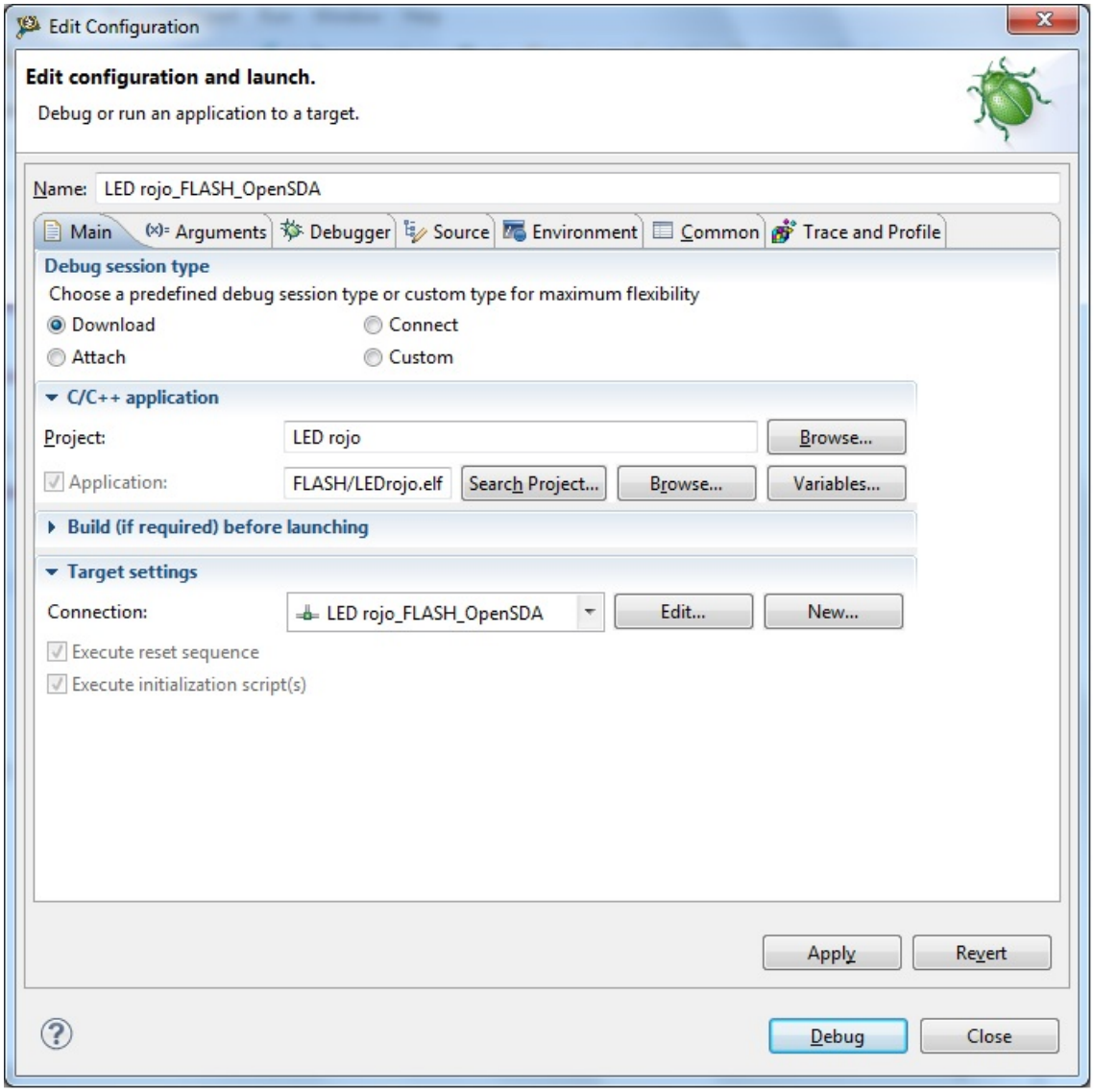


Figura 24. Edición de la configuración de debugger.

Una vez se ha ejecutado la aplicación, estamos en la ventana de debugger. Se pueden intercambiar la pantalla de debugger con la pantalla de programación con las teclas siguientes 

Con la aplicación en marcha, se puede ejecutar paso a  paso o bien de modo continuo  , indistintamente. Estos modos se encuentran disponibles en el panel de control.

Utilizar en modo paso a paso, permite visualizar el estado de los diferentes registros o el valor que en cada caso tienen cada una de las variables (marcado con amarillo en la figura 24).

Una vez que se ha ejecutado en modo continuo, el fichero correspondiente esta grabado en la memoria FLASH del sistema de desarrollo y evaluación FRDM-KL25Z.

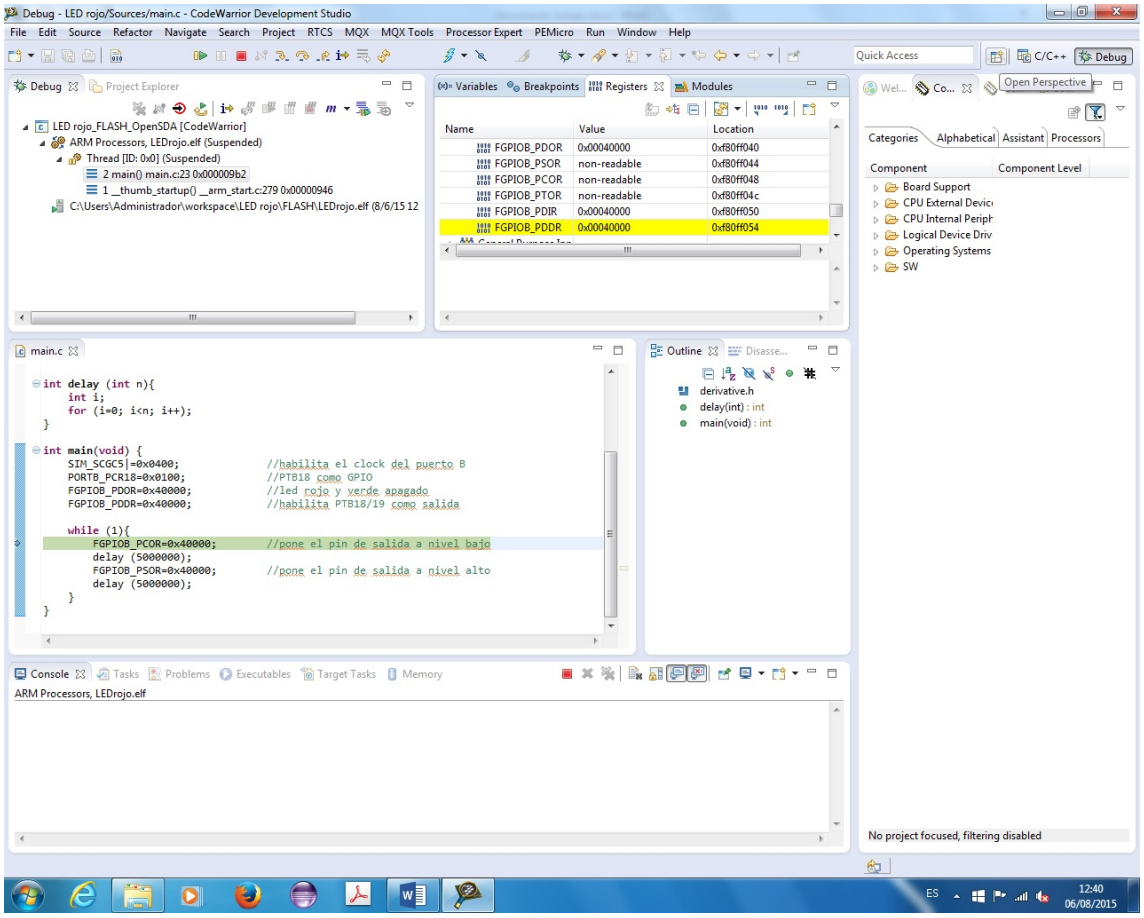


Figura 25. Página de debugger, ejecución paso a paso.

A.2 Código implementado en el proyecto

Se incluyen en este Anexo los programas necesarios para probar las funciones realizadas en este proyecto. Se incluye además el programa LED RGB el cual nos ha servido para comunicar la placa con el ordenador, y comprobar su funcionamiento.

Como norma general, hay que documentar bien los programas. Tiene que ser fácil para los posibles usuarios posteriores la utilización de las funciones aquí implementadas. Todo programa consta de varias partes.

Un encabezamiento, son comentarios que facilitan su uso. Una línea #include que declara funciones generales utilizables para la placa FRDM-KL25Z. Una declaración de variables y de constantes que en este ejemplo inicial no existe. El programa principal, el cual incluye unas primeras órdenes que permiten la inicialización de puertos, habilitación de entradas o salidas, inicio de clock. Posteriormente el programa principal.

A.2.1 LED RGB

La función consiste en encender el color rojo del led RGB disponible en el sistema de desarrollo y evaluación FRDM-KL25Z. Se realiza un bucle con el encendido y apagado cada 0,5 segundos del led rojo.

```
//LED ROJO

//Este programa enciende el led rojo del sistema de desarrollo FRDM-KL25Z

//El led está conectado al pin 18 del puerto B y se activa con nivel bajo

#include "derivative.h"                    //declaración de funciones y variables


int delay (int n){
       int i;
       for (i=0; i<n; i++);
}

int main(void) {
       SIM_SCGC5|=0x0400;                  //habilita el clock del puerto B
       SIM_COPC=0X0000;                    //disabled cop watchdog

       PORTB_PCR18=0x0100;                 //PTB18 como GPIO
       FGPIOB_PDOR=0x40000;                //led rojo azul y verde apagado
       FGPIOB_PDDR=0x40000;                //habilita PTB18/19 como salida

       while (1){
              FGPIOB_PCOR=0x40000;         //pone el pin de salida a nivel bajo
              delay (5000000);             //llamada a función retardo 0,5s
              FGPIOB_PSOR=0x40000;         //pone el pin de salida a nivel alto
              delay (5000000);             //llamada a función retardo 0,5s

       }
}
```

A.2.2 Display 7 segmentos visualización estática

```
//Display Estático

//Visualización estática de un número entre 00 y 99

#include "derivative.h"                    //declaración de funciones y variables

int dato;                                  //declaración de variables de 16 bits


int VisEst (int dato) {

        int numero, auxiliar, unidad, decena;

        numero=dato;
        decena=numero/10;                  //obtengo el dígito decena
        unidad=numero%10;                  //obtengo el dígito unidad
        auxiliar=decena<<4;                //registro auxiliar que desplaza a la
                                           izquierda el digito decena
        FGPIOC_PDOR=auxiliar+unidad;       //salida de ambos displays, 4 bits decenas,
                                           4 bits unidades
        }

int main(void) {

        SIM_SCGC5|=0X0800;                 //habilito el clock del puerto C
        SIM_COPC=0X0000;                   //disabled cop watchdog

        PORTC_PCR0=0X0100;                 //PTC0 como GPIO
        PORTC_PCR1=0X0100;                 //PTC1 como GPIO
        PORTC_PCR2=0X0100;                 //PTC2 como GPIO
        PORTC_PCR3=0X0100;                 //PTC3 como GPIO
        PORTC_PCR4=0X0100;                 //PTC4 como GPIO
        PORTC_PCR5=0X0100;                 //PTC5 como GPIO
        PORTC_PCR6=0X0100;                 //PTC6 como GPIO
        PORTC_PCR7=0X0100;                 //PTC7 como GPIO

        FGPIOC_PDDR=0X00FF;                //habilito las salidas del puerto C
        FGPIOC_PCOR=0x00FF;                //inicializo el registro a 0

        while (1) {

                VisEst (27);               //llamada a función
        }
}
```

A.2.3 Display 7 segmentos visualización dinámica

```
//Display Dinámico
//Visualización dinámica de un número entre 0000 y 9999

#include "derivative.h"                    //declaración de funciones y variables
int dato=1234;                             //declaración de constantes
int tiempo;                                //declaración de variables

int VisDin (int dato, int tiempo) {

       int delay (int n) {
              int i;
              for (i=0; i<n; i++);
       }

int numero, auxiliar, unidad, decena, centena, millar, temp;
numero=dato;

while (temp=tiempo/4) {

unidad=numero%10;                          //obtengo unidad
auxiliar=unidad<<2;                        //carga y desplazamiento del número
FGPIOE_PDOR=auxiliar+0X100000;             //salida, con número y display unidades
delay (3000);                              //espera 1ms

decena=(numero/10)%10;                     //obtengo decena
auxiliar=decena<<2;                        //carga y desplazamiento del número
FGPIOE_PDOR=auxiliar+0X200000;             //salida, con número y display decenas
delay (3000);                              //espera 1ms

centena=(numero/100)%10;                   //obtengo centena
auxiliar=centena<<2;                       //carga y desplazamiento del número
FGPIOE_PDOR=auxiliar+0X400000;             //salida, con número y display centenas
delay (3000);                              //espera 1ms

millar=(numero/1000)%10;                   //obtengo millar
auxiliar=millar<<2;                        //carga y desplazamiento del número
FGPIOE_PDOR=auxiliar+0X800000;             //salida, con número y display millares
delay (3000);                              //espera 1ms
       }
}

int main(void) {

       SIM_SCGC5|=0X2000;                  //habilito el clock del puerto E
       SIM_COPC=0X0000;                    //disabled cop watchdog
       PORTE_PCR2=0X0100;                  //PTE2 como GPIO
       PORTE_PCR3=0X0100;                  //PTE3 como GPIO
       PORTE_PCR4=0X0100;                  //PTE4 como GPIO
       PORTE_PCR5=0X0100;                  //PTE5 como GPIO
       PORTE_PCR20=0X0100;                 //PTE20 como GPIO
       PORTE_PCR21=0X0100;                 //PTE21 como GPIO
       PORTE_PCR22=0X0100;                 //PTE22 como GPIO
       PORTE_PCR23=0X0100;                 //PTE23 como GPIO
       FGPIOE_PCOR=0xF0003C;               //inicializo el registro a 0
       FGPIOE_PDDR=0XF0003C;               //habilito las salidas del puerto E

       while(1) {

              int VisDin (int dato, int tiempo);  //llamada a función
       }
}
```

A.2.4 LCD 4 bits

Se presenta el diagrama de flujo del programa principal con las llamadas a las diversas funciones implementadas.

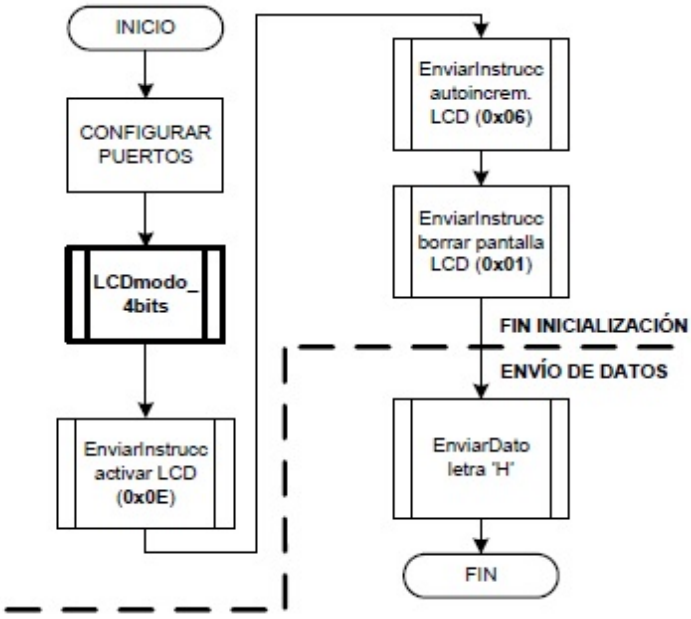


Figura 26. Diagrama de flujo programa principal LCD formato 4 bits.

```
//LCD 4 bits

#include "derivative.h"                    //declaración de funciones y variables

char instrucc, dato;

int delay (int n) {
              int i;
              for (i=0; i<n; i++);
       }

void EnviarInstruccion (char instrucc) {

       char auxiliar;
       auxiliar=instrucc;

       FGPIOC_PDOR=auxiliar;               //pasar nibble alto de la instrucción
       FGPIOE_PDOR=0x08;                   //entrada de control RS=0 E=1
       delay (15000);                      //espera 5ms
       FGPIOE_PDOR=0x00;                   //entrada de control RS=0 E=0
       delay (15000);                      //espera 5ms
       FGPIOC_PDOR=auxiliar<<4;            //pasar nibble bajo de la instrucción
       FGPIOE_PDOR=0x08;                   //entrada de control RS=0 E=1
       delay (15000);                      //espera 5ms
       FGPIOE_PDOR=0x00;                   //entrada de control RS=0 E=0
       delay (15000);                      //espera 5ms
       }
```

```
void EnviarDato (char dato) {

	char auxiliar;
	auxiliar=dato;

	FGPIOC_PDOR=auxiliar;		//pasar nibble alto de la instrucción
	FGPIOE_PDOR=0x0C;		//entrada de control RS=1 E=1
	delay (15000);			//espera 5ms
	FGPIOE_PDOR=0x04;		//entrada de control RS=1 E=0
	delay (15000);			//espera 5ms
	FGPIOC_PDOR=auxiliar<<4;	//pasar nibble bajo de la instrucción
	FGPIOE_PDOR=0x0C;		//entrada de control RS=1 E=1
	delay (15000);			//espera 5ms
	FGPIOE_PDOR=0x04;		//entrada de control RS=1 E=0
	delay (15000);			//espera 5ms
	}

void LCDmodo4bits() {

	int i;
	for (i=0; i<3; i++) {

		FGPIOC_PDOR=0x30;	//pasar instrucción de arranque
		FGPIOE_PDOR=0x08;	//entrada de control RS=0 E=1
		delay (15000);		//espera 5ms
		FGPIOE_PDOR=0x00;	//entrada de control RS=0 E=0
		delay (15000);		//espera 5ms
		delay (30000);		//espera 10ms
		}

	FGPIOC_PDOR=0x20;		//pasar instrucción de arranque 4 bits
	FGPIOE_PDOR=0x08;		//entrada de control RS=0 E=1
	delay (15000);			//espera 5ms
	FGPIOE_PDOR=0x00;		//entrada de control RS=0 E=0
	delay (15000);			//espera 5ms
	EnviarInstruccion(0x20);	//llamada función
	}


int main(void) {

	SIM_SCGC5|=0X2800;		//habilito el clock del puerto C y E
	SIM_COPC=0X0000;		//disabled cop watchdog

	PORTC_PCR4=0X0100;		//PTC4 como GPIO
	PORTC_PCR5=0X0100;		//PTC5 como GPIO
	PORTC_PCR6=0X0100;		//PTC6 como GPIO
	PORTC_PCR7=0X0100;		//PTC7 como GPIO
	PORTE_PCR2=0X0100;		//PTE2 como GPIO
	PORTE_PCR3=0X0100;		//PTE3 como GPIO

	FGPIOC_PDDR=0X00F0;		//habilito como salidas del puerto C
	FGPIOE_PDDR=0X000C;		//habilito como salidas del puerto E
	FGPIOC_PCOR=0X00FF;		//inicializo el registro C a 0
	FGPIOE_PCOR=0X00FF;		//inicializo el registro E a 0


	LCDmodo4bits();			//llamada a función
	EnviarInstruccion(0x0E);	//llamada a función
	EnviarInstruccion(0x06);	//llamada a función
	EnviarInstruccion(0x01);	//llamada a función
	EnviarDato(0x48);		//llamada a función
}
```

A.2.5 LCD 8 bits

```
//LCD 8 bits

#include "derivative.h"                    //declaración de funciones y variables
char instrucc, dato;

int delay (int n) {
              int i;
              for (i=0; i<n; i++);
       }

void EnviarInstruccion (char instrucc) {

       char auxiliar;
       auxiliar=instrucc;

       FGPIOC_PDOR=auxiliar;               //pasar la instrucción
       FGPIOE_PDOR=0x08;                   //entrada de control RS=0 E=1
       delay (15000);                      //espera 5ms
       FGPIOE_PDOR=0x00;                   //entrada de control RS=0 E=0
       delay (15000);                      //espera 5ms
       }

void EnviarDato (char dato) {

       char auxiliar;
       auxiliar=dato;

       FGPIOC_PDOR=auxiliar;               //pasar el dato
       FGPIOE_PDOR=0x0C;                   //entrada de control RS=1 E=1
       delay (15000);                      //espera 5ms
       FGPIOE_PDOR=0x04;                   //entrada de control RS=1 E=0
       delay (15000);                      //espera 5ms
       }

void LCDmodo8bits() {

       int i;
       for (i=0; i<3; i++) {

              FGPIOC_PDOR=0x30;            //pasar instrucción de arranque
              FGPIOE_PDOR=0x08;            //entrada de control RS=0 E=1
              delay (15000);               //espera 5ms
              FGPIOE_PDOR=0x00;            //entrada de control RS=0 E=0
              delay (15000);               //espera 5ms
              delay (30000);               //espera 10ms
              }
       }

int main(void) {

       SIM_SCGC5|=0X2800;                  //habilito el clock del puerto C y E
       SIM_COPC=0X0000;                    //disabled cop watchdog
       PORTC_PCR0=0X0100;                  //PTC0 como GPIO
       PORTC_PCR1=0X0100;                  //PTC1 como GPIO
       PORTC_PCR2=0X0100;                  //PTC2 como GPIO
       PORTC_PCR3=0X0100;                  //PTC3 como GPIO
       PORTC_PCR4=0X0100;                  //PTC4 como GPIO
       PORTC_PCR5=0X0100;                  //PTC5 como GPIO
       PORTC_PCR6=0X0100;                  //PTC6 como GPIO
       PORTC_PCR7=0X0100;                  //PTC7 como GPIO
       PORTE_PCR2=0X0100;                  //PTE2 como GPIO
```

```
        PORTE_PCR3=0X0100;                  //PTE3 como GPIO
        FGPIOC_PDDR=0X00FF;                 //habilito como salidas del puerto C
        FGPIOE_PDDR=0X000C;                 //habilito como salidas del puerto E
        FGPIOC_PCOR=0X00FF;                 //inicializo el registro C a 0
        FGPIOE_PCOR=0X00FF;                 //inicializo el registro E a 0

        LCDmodo8bits();                     //llamada a función
        EnviarInstruccion(0x0E);            //llamada a función
        EnviarInstruccion(0x06);            //llamada a función
        EnviarInstruccion(0x01);            //llamada a función
        EnviarDato(0x48);                   //llamada a función
}
```

A.2.6 Memoria EEPROM

El clock del microcontrolador está controlado por los registros del Módulo de Generación del Reloj (MCG en inglés). Se compone de dos relojes de referencia interna IRC, uno de 32KHz y otro de 4MHz, que puede dividirse mediante el registro SIM_CLKDIV1.

Existen dos modos de trabajo:

- Modo FLL, mediante el reloj lento de 32KHz o una fuente externa como reloj de referencia. Se puede seleccionar , habilitar y controlar este modo mediante los registros SIM_SOPT1 y SIM_SOPT2.

- Modo PLL, con una fuente externa como reloj de referencia (no disponible en todos los dispositivos).

Existen del mismo modo, varios tipos de clock:

- MCGOUTCLK, reloj del sistema principal, núcleo, bus y memoria. Diversas formas de generarse.

- MCGFLLCLK, se produce con la salida FLL seleccionada, disponible en el momento que FLL está habilitado.

- MCGPLLCLK, salida PLL seleccionada, disponible en el momento que PLL está habilitado.

- MCGIRCLK, salida IRC seleccionada, disponible en el modo que se selecciona este reloj.

- OSCERCLK, proporcionado por el sistema a partir de una fuente de reloj externa de onda cuadrada.

- ERCLK32K, la salida del oscilador del sistema está configurado en baja potencia, modo de 32KHz, RTCCLKIN o el oscilador LPO.

- LPO, salida de baja frecuencia disponible en todos los modos de trabajo.

MCG puede configurarse en varios modos de trabajo, FEI, FEE, FBI, FBE, PBE, PEE, BLPI, BLPE, y modo Stop. Una vez realizados cambios en la selección del reloj hay que comprobar los bits de estado para continuar.

Configuración del reloj RTC, no tiene un reloj dedicado, pero puede seleccionarse de entre: reloj del sistema a 32KHz (OSC32KCLK), el pin RTC_CLKIN, LPO. La función RTCCLKOUT solo será correcta si se utiliza a 32,768KHz.

En el MCG se dispone de varios registros de control de 8 bits. MCG_C1 en donde se puede seleccionar que tipo de referencia interna elegimos y su habilitación. MCG_C2 y sucesivos con distintos tipos de control, selección del rango de frecuencia, selección de referencia externa, potencia y diversos parámetros que podemos controlar, todos ellos disponibles en el documento pdf ***KL25 Sub-Family Reference Manual.***

Del mismo modo, para inicializar el módulo I2C se utilizan diversos registros SIM_SCGC4 para habilitar el reloj del módulo I2C. Los registros de control disponibles son de 8 bits de igual forma que los registros del módulo MCG.

I2C1_C1 y sucesivos registros de control permiten la habilitación del módulo I2C, las interrupciones, el modo maestro/esclavo, el modo de transmisión, la señal de ACK de comprobación de transmisión o recepción y también se encargan de generar las señales de START y STOP.

```
//Memoria serie I2C


#include "derivative.h" /* include peripheral declarations */

int direccion;
char dato;


void configuracion (void) {

	SIM_SCGC5 |= SIM_SCGC5_PORTE_MASK |
	             SIM_SCGC5_PORTD_MASK |
	             SIM_SCGC5_PORTC_MASK |
	             SIM_SCGC5_PORTB_MASK |
	             SIM_SCGC5_PORTA_MASK;         //habilitación reloj de los puertos

	SIM_CLKDIV1 = 0x00U;                       //divisor del reloj para 1
	SIM_SOPT2 &= ~SIM_SOPT2_PLLFLLSEL_MASK;
	                    //PLLFLLSEL=0 selecciona FLL para diversos periféricos
	SIM_SOPT1 |=  SIM_SOPT1_OSC32KSEL(0X03);   //LPO 1KHz
	SIM_SOPT2 = (SIM_SOPT2&~SIM_SOPT2_TPMSRC(0x02))|SIM_SOPT2_TPMSRC(0X01);
	                    //selecciona MCGFLLCLK
	MCG_C1 = (MCG_C1_IREFS_MASK|MCG_C1_IRCLKEN_MASK); //IREFS=1,IRCLKEN=1
	MCG_C2 = 0x00U;                            //borrado registro de control
	MCG_C4 &= ~(MCG_C4_DMX32_MASK|MCG_C4_DRST_DRS(0x03)); //DMX32=0,DRST_DRS=0

	OSC0_CR = OSC_CR_ERCLKEN_MASK;             //ERCLKEN=1
	MCG_C5 = 0x00U;                            //borrado registro de control
	MCG_C6 = 0x00U;                            //borrado registro de control

	while((MCG_S&MCG_S_IREFST_MASK) == 0x00U);
	                    //esperar hasta que el reloj FLL es el reloj utilizado
	while((MCG_S & 0x0CU) != 0x00U);    // esperar hasta que FLL es seleccionado
}


void I2Cinicializacion (void) {

	SIM_SCGC4 |= SIM_SCGC4_I2C1_MASK;          //habilitación reloj I2C1
	I2C1_C1 = 0x00U;                           //borrado registro de control
	I2C1_FLT = I2C_FLT_STOPF_MASK;             //borrado flag estado del bus
	I2C1_S = I2C_S_IICIF_MASK;                 //borrado flag de interrupciones

	PORTE_PCR0 &= ~(PORT_PCR_MUX_MASK);
	PORTE_PCR0 |= (PORT_PCR_MUX(6));
	PORTE_PCR1 &= ~(PORT_PCR_MUX_MASK);
	PORTE_PCR1 |= (PORT_PCR_MUX(6));

	I2C1_C2 = 0x00U;                           //borrado registro de control
	I2C1_FLT = 0x00U;                          //borrado glitch registro
	I2C1_SMB = I2C_SMB_SLTF_MASK;              //SLTF=1
	I2C1_F = 0x00U;                            //inicializar registro
	I2C1_C1 |=(I2C_C1_IICEN_MASK|I2C_C1_IICIE_MASK|I2C_C1_TXAK_MASK);
	              /*IICEN=1 habilitación I2C, IICIE=1 habilitación interrupciones
	        I2C, TXAK=1 habilitación transmisión ACK*/
}
```

```
void EscribirMemoria (int direccion, char dato) {

	int aux_direccion;
	char aux_dato;

	aux_direccion = direccion;
	aux_dato = dato;

	I2C1_C1 |= I2C_C1_TX_MASK;                //habilitar transmisión
	I2C1_C1 |= I2C_C1_MST_MASK;               //habilitar master, enviar START

	I2C1_D = 0XA0;                            //enviar byte control (10100000)
	while(!(I2C1_S & I2C_S_TCF_MASK));        //esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    /comprobar si recepcionado ACK
	{
		I2C1_C1 |= I2C_C1_TXAK_MASK;      //reset flag ACK
		I2C1_D = aux_direccion;           //enviar 8 primeros bits dirección
		while(!(I2C1_S & I2C_S_TCF_MASK));//esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    //comprobar si recepcionado ACK
	{
		I2C1_C1 |= I2C_C1_TXAK_MASK;      //reset flag ACK
		I2C1_D = aux_direccion<<8;        //enviar segundos 8 bits dirección
		while(!(I2C1_S & I2C_S_TCF_MASK));//esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    //comprobar si recepcionado ACK
	{
		I2C1_C1 |= I2C_C1_TXAK_MASK;      //reset flag ACK
		I2C1_D = aux_dato;                //enviar dato
		while(!(I2C1_S & I2C_S_TCF_MASK));//esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    //comprobar si recepcionado ACK
	{
		I2C0_C1 |= I2C_C1_TXAK_MASK;      //reset flag ACK
		I2C0_C1 |= (~I2C_C1_MST_MASK);    //deshabilitar master, enviar STOP
	}
	}
	}
	}
}


void LeerMemoria (int direccion) {

	int aux_direccion;
	char aux_dato;

	aux_direccion = direccion;

	I2C1_C1 |= I2C_C1_TX_MASK;                //habilitar transmisión
	I2C1_C1 |= I2C_C1_MST_MASK;               //habilitar master, enviar START

	I2C1_D = 0XA0;                      //enviar byte control (10100000) escribir
	while(!(I2C1_S & I2C_S_TCF_MASK));        //esperar transferencia finalizada

	if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)    //comprobar si recepcionado ACK
	{
		I2C1_C1 |= I2C_C1_TXAK_MASK;      //reset flag ACK
		I2C1_D = aux_direccion;           //enviar 8 primeros bits dirección
		while(!(I2C1_S & I2C_S_TCF_MASK));//esperar transferencia finalizada
```

```
        if((I2C1_S & I2C_S_RXAK_MASK ) == 0X00)    //comprobar si recepcionado ACK
        {
                I2C1_C1 |= I2C_C1_TXAK_MASK;       //reset flag ACK
                I2C1_D = aux_direccion<<8;         //enviar segundos 8 bits dirección
                while(!(I2C1_S & I2C_S_TCF_MASK)); //esperar transferencia finalizada

        if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)     //comprobar si recepcionado ACK
        {
                I2C1_C1 |= I2C_C1_TXAK_MASK;       //reset flag ACK
                I2C1_C1 |= I2C_C1_MST_MASK;        //habilitar master, enviar START

                I2C1_D = 0XA1;              //enviar byte control (10100001) leer
                while(!(I2C1_S & I2C_S_TCF_MASK)); //esperar transferencia finalizada

        if((I2C1_S & I2C_S_RXAK_MASK) == 0X00)     //comprobar si recepcionado ACK
        {
                I2C1_C1|=I2C_C1_TXAK_MASK;         //reset flag ACK
                while(!(I2C1_S & I2C_S_TCF_MASK)); //esperar transferencia finalizada
                aux_dato = I2C1_D;                 //leer dato de la memoria
                I2C1_C1 |= (~I2C_C1_MST_MASK);     //deshabilitar master, enviar STOP
        }
        }
        }
        }

}


int main(void) {

        SIM_COPC = 0X00U;                          //deshabilitar cop watchdog

        configuracion();                           //llamada a función
        I2Cinicializacion();                       //llamada a función

        EscribirMemoria(1100,34);                  //llamada a función
        LeerMemoria(0100);                         //llamada a función

        }
```

A.3 Planos PCB del Sistema de Desarrollo

Se ha desarrollado y construido el prototipo del Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes. Se trata de un primer prototipo, que hay que desarrollar hasta llegar a un sistema real que pueda utilizarse en las prácticas de SEP.

El planteamiento inicial es que la placa fuera versátil y universal, para lo cual se ha distribuido por módulos independientes. Las conexiones de los diversos módulos con el sistema de desarrollo y evaluación FRDM-KL25Z se realizan mediante conectores y a su vez con cables, de forma que el alumno en cada caso elige que puerto utilizar.

En el prototipo siguiente se implementará entre otras cosas la posibilidad de utilizar este sistema de desarrollo con la placa actual DEMO9S08QG8 de Freescale.

Se puede observar, la no coincidencia entre el primer prototipo y la serigrafía de componentes debido a las modificaciones que se han tenido que realizar para su funcionamiento.

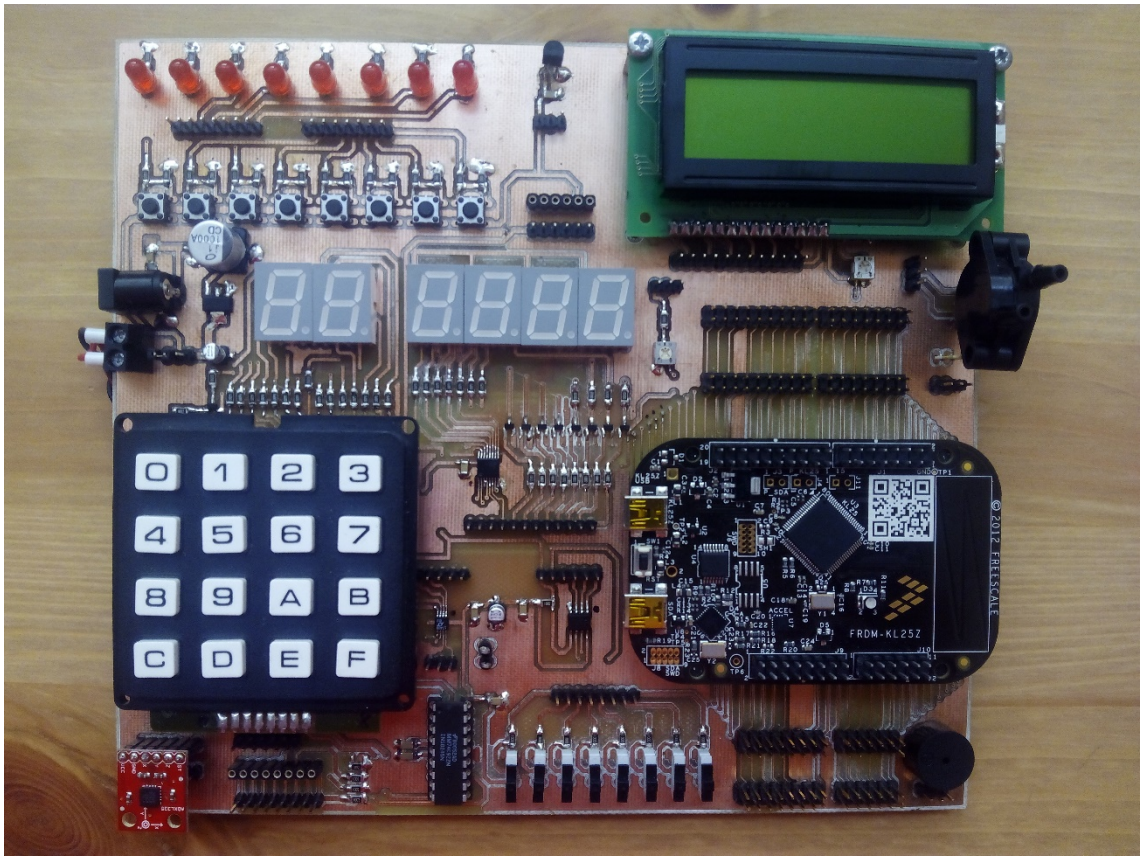
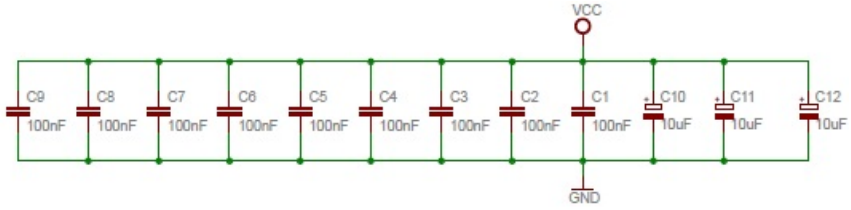
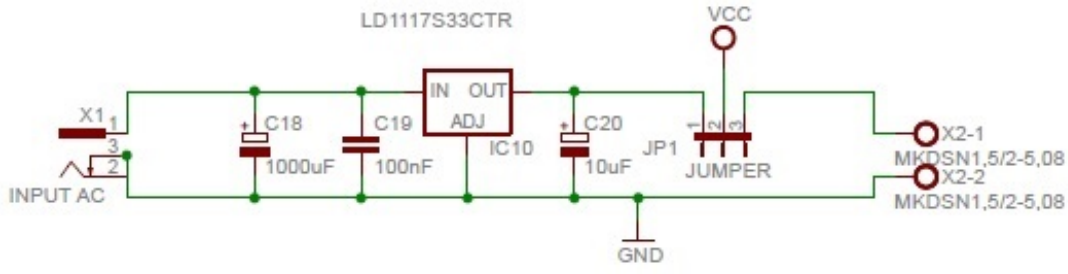
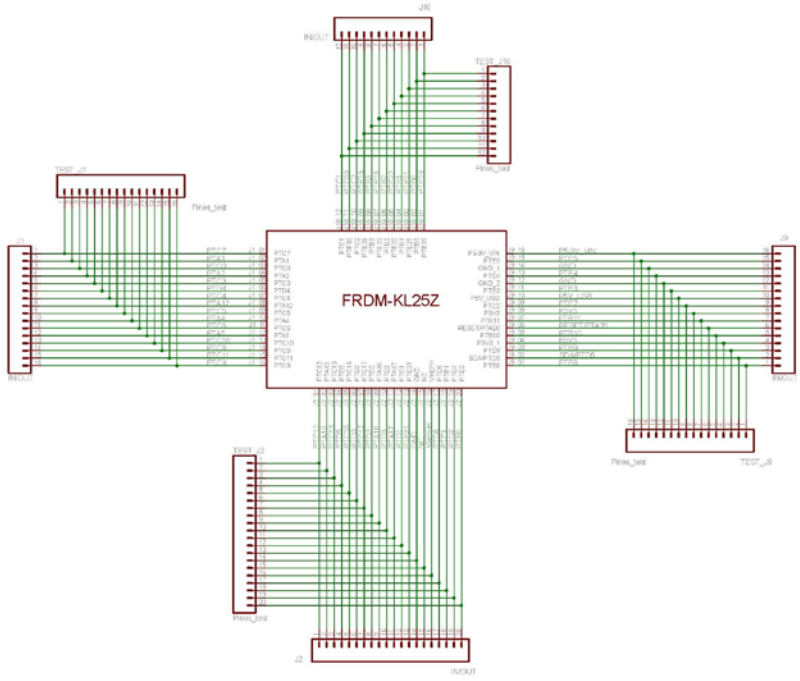
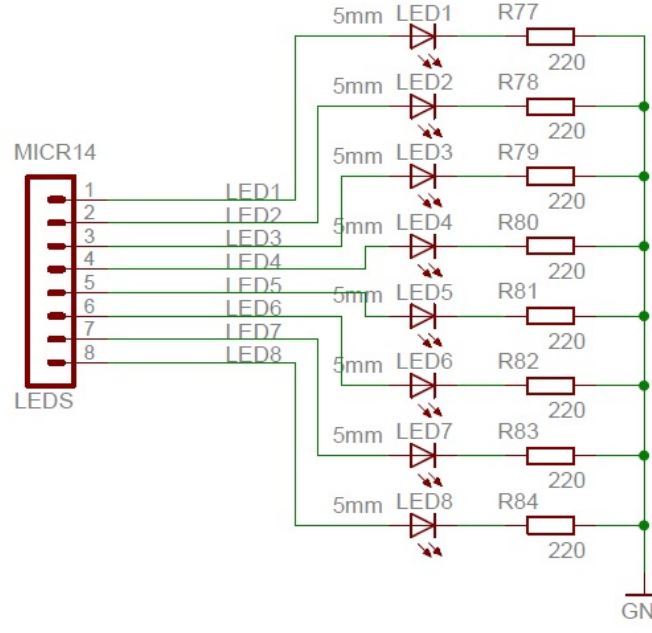


Figura 27. Prototipo PCB.

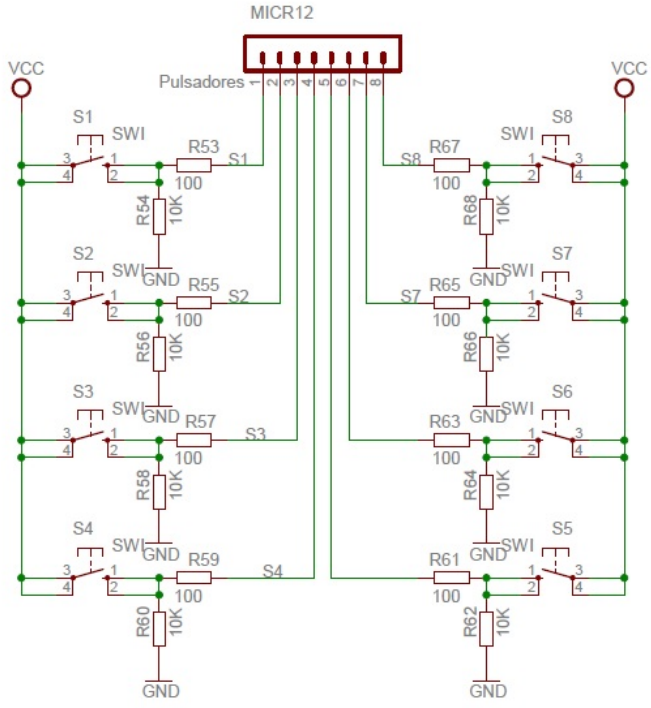


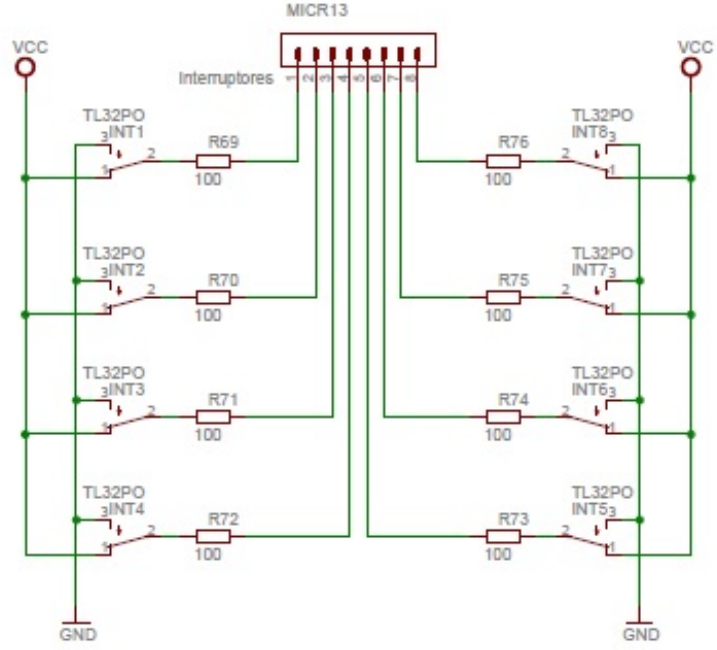


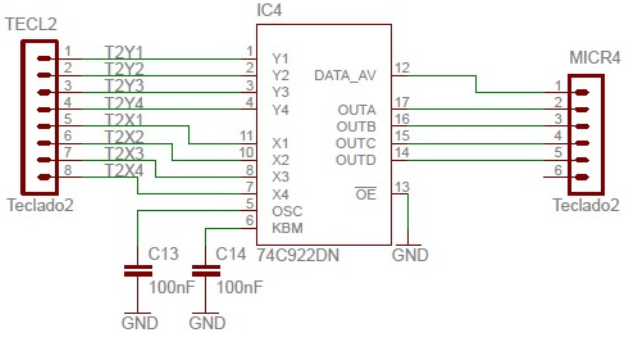


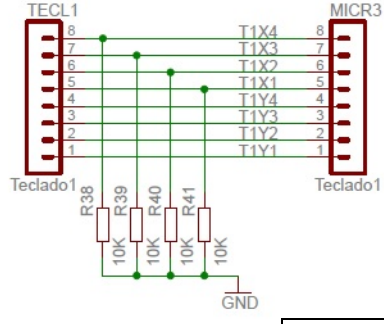


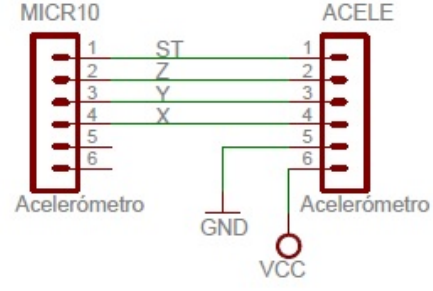
	Fecha	Nombre	Firma:	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
Dibujado	31/8/2015	Carlos E. Jarauta		
Comprobado				
Escala:	Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes Esquema eléctrico		Plano Nº: 1.1	Universidad Zaragoza
			Nº del alumno 386528	
			Curso PFC	



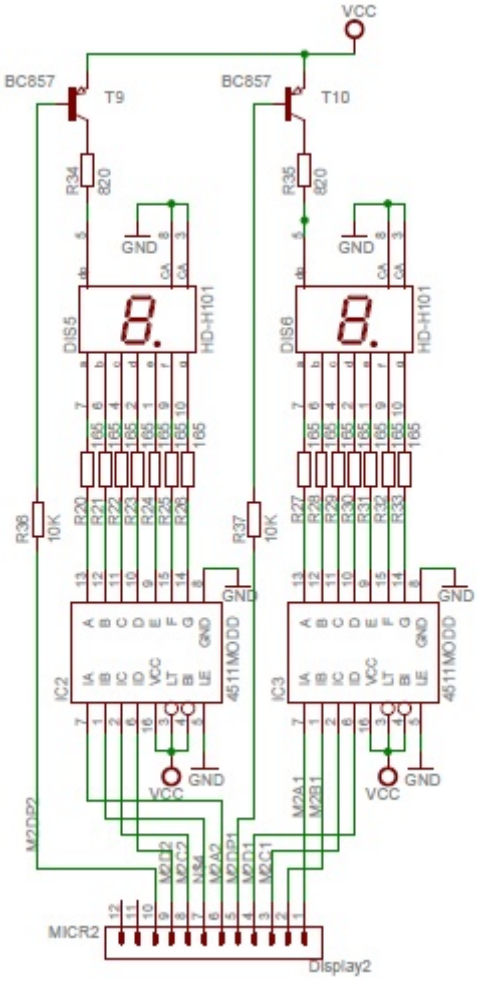


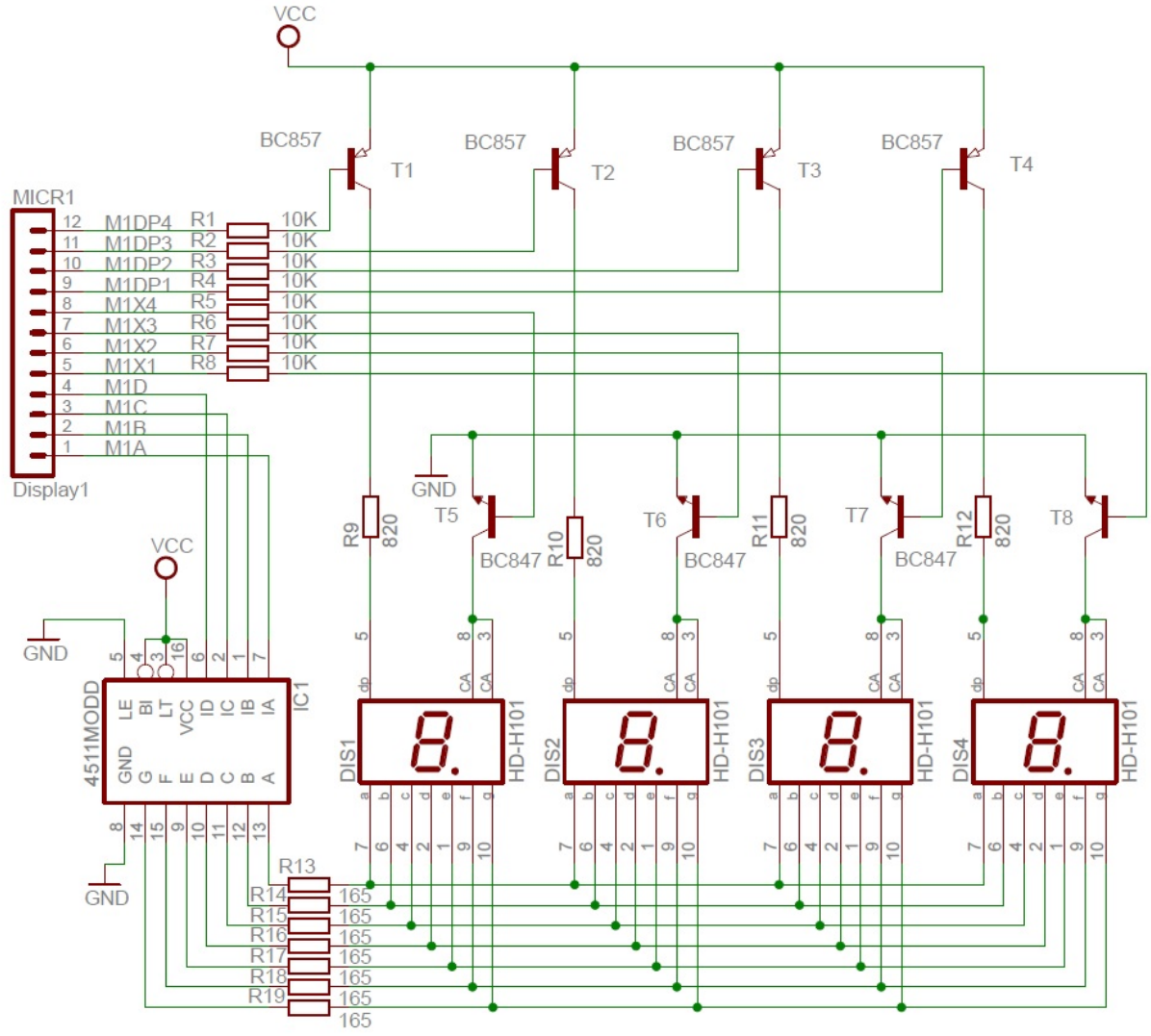




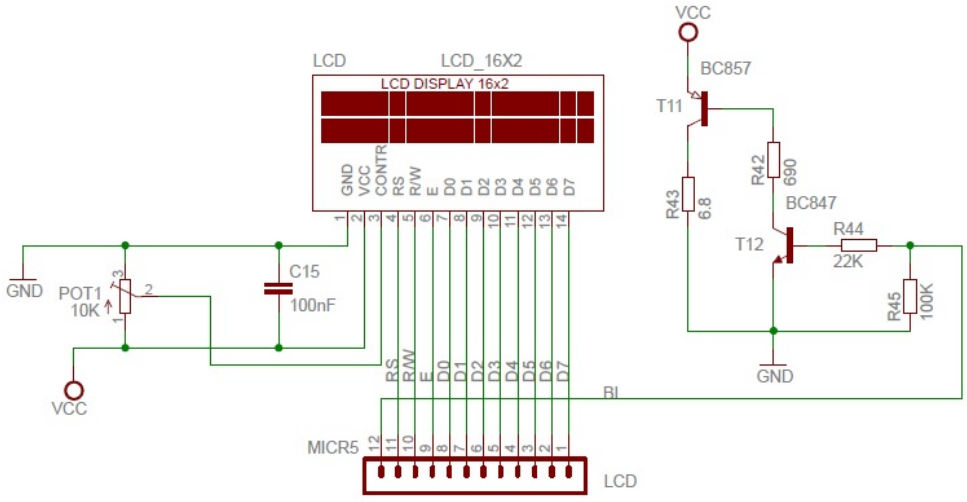


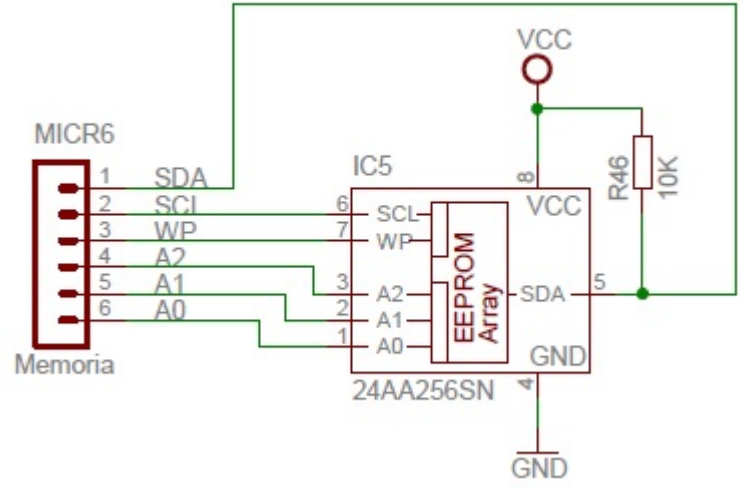
	Fecha	Nombre	Firma:	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
Dibujado	31/8/2015	Carlos E. Jarauta		
Comprobado				
Escala:	Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes Esquema eléctrico		Plano Nº: 1.2	Universidad Zaragoza
			Nº del alumno 386528	
			Curso PFC	

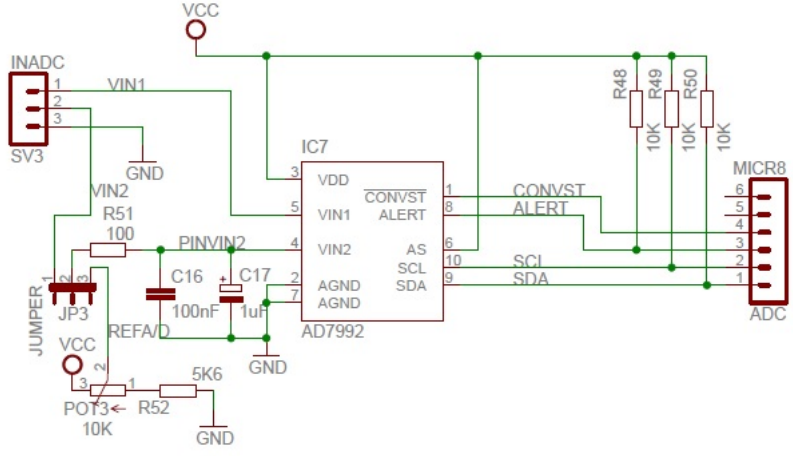


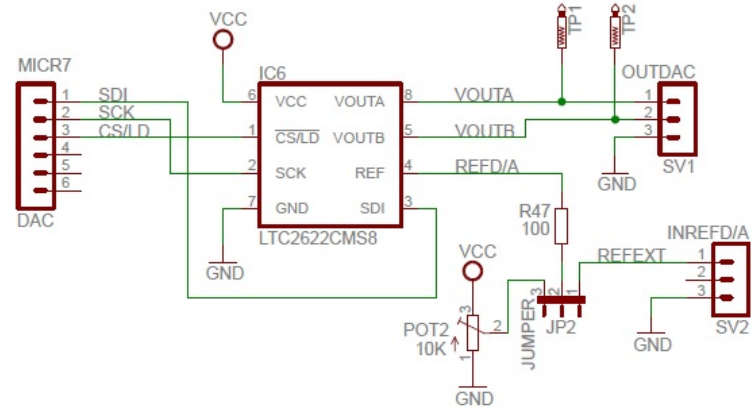


	Fecha	Nombre	Firma:	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
Dibujado	31/8/2015	Carlos E. Jarauta		
Comprobado				
Escala:	Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes Esquema eléctrico		Plano Nº: 1.3	Universidad Zaragoza
			Nº del alumno 386528	
			Curso PFC	

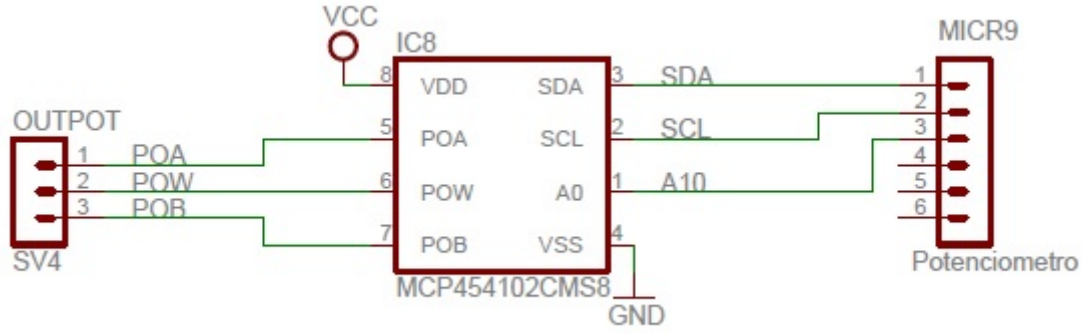


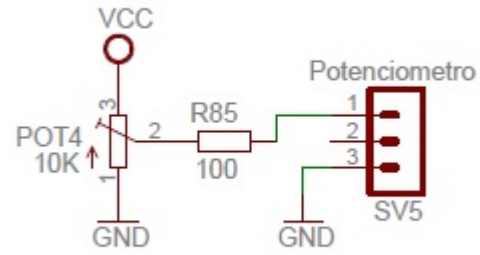


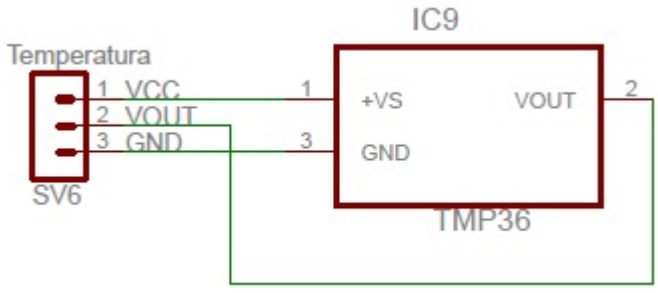


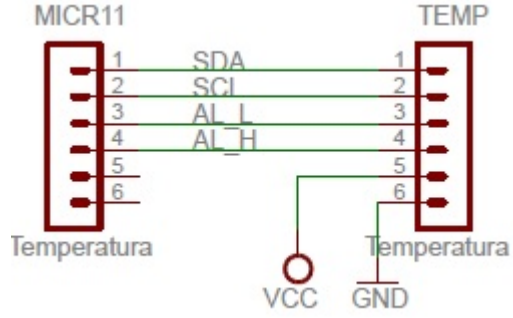


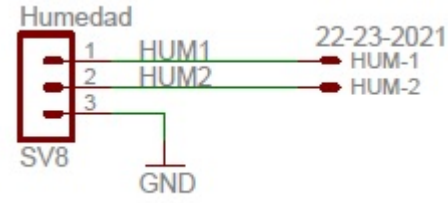
	Fecha	Nombre	Firma:	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
Dibujado	31/8/2015	Carlos E. Jarauta		
Comprobado				
Escala:	Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes Esquema eléctrico		Plano Nº: 1.4	Universidad Zaragoza
			Nº del alumno 386528	
			Curso PFC	

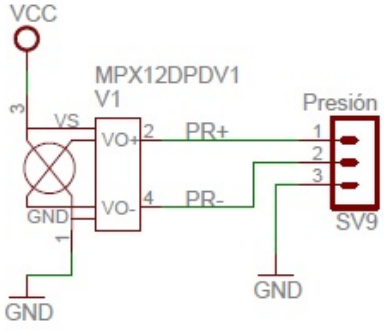


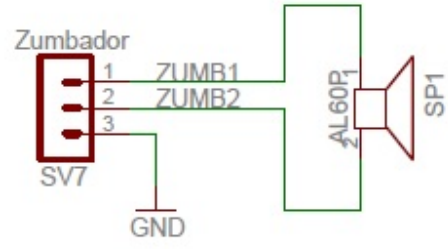




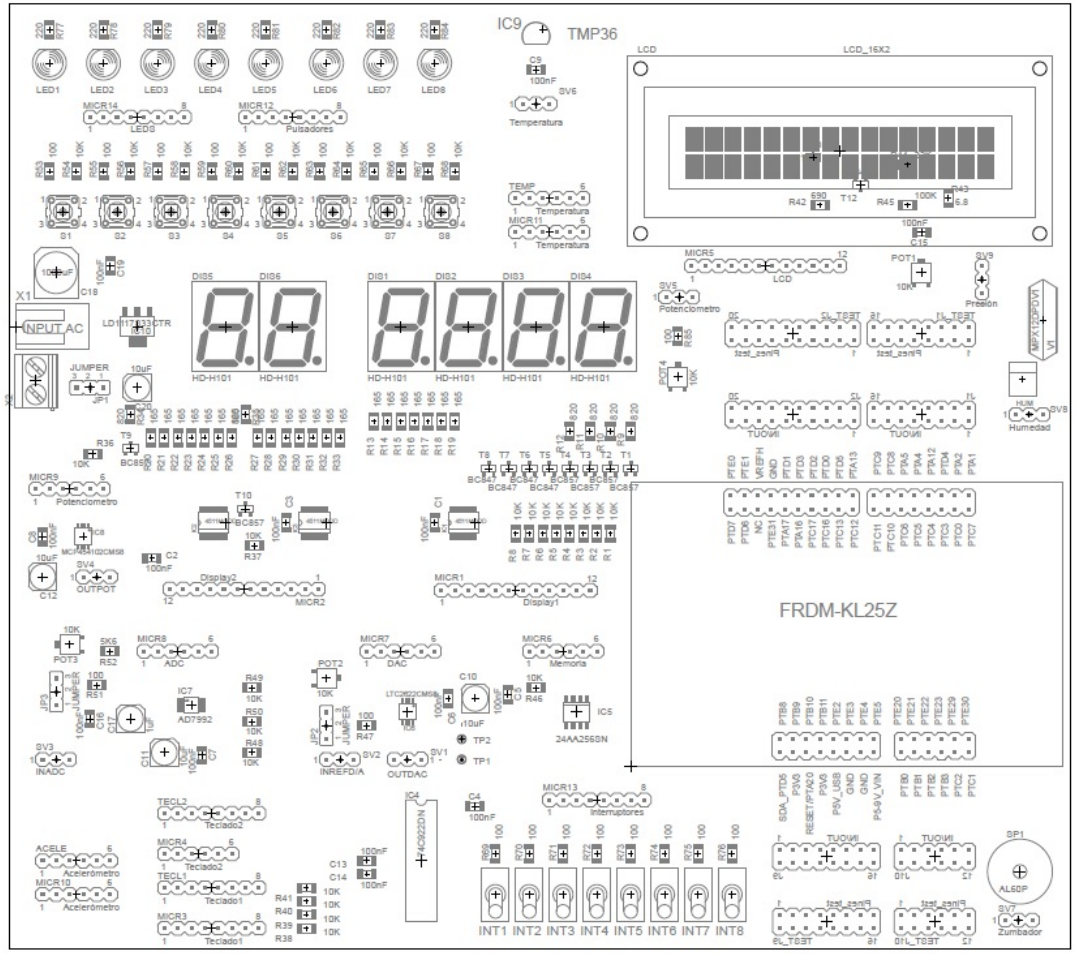




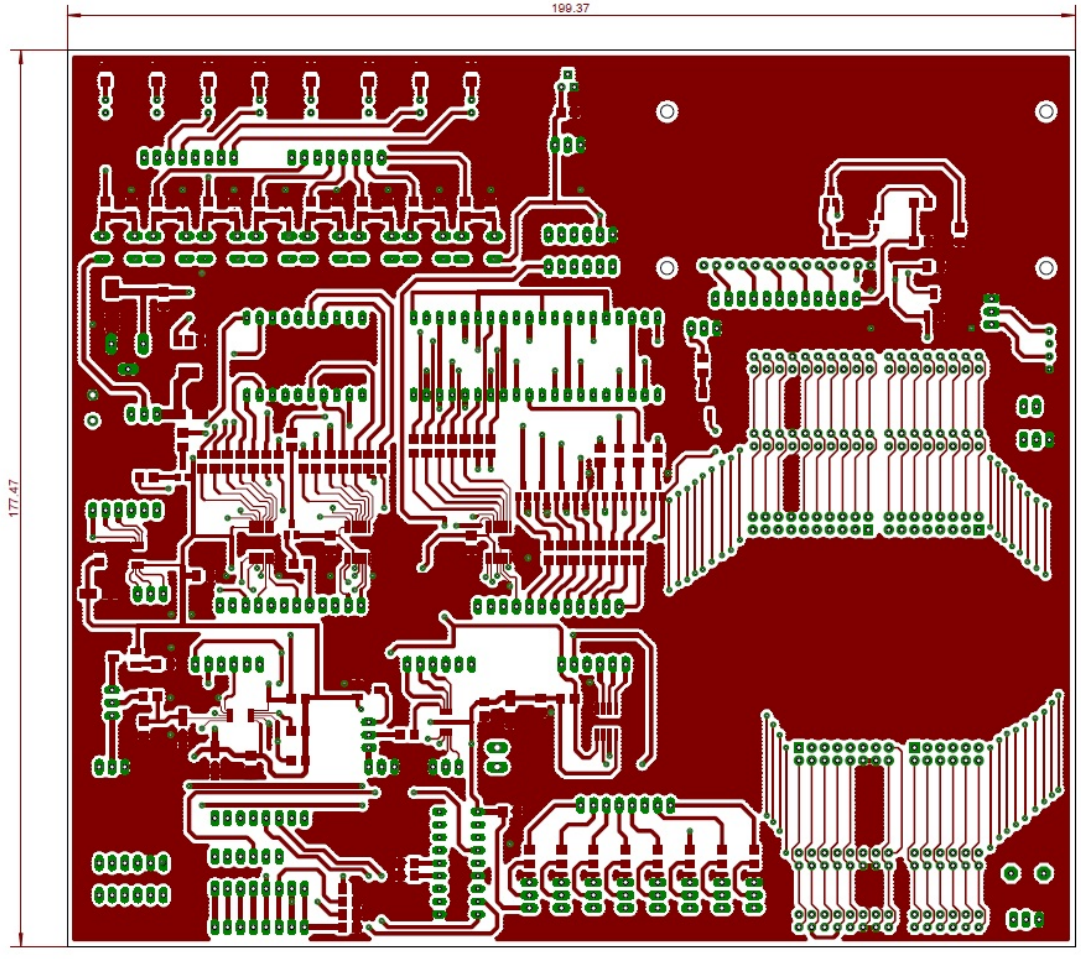




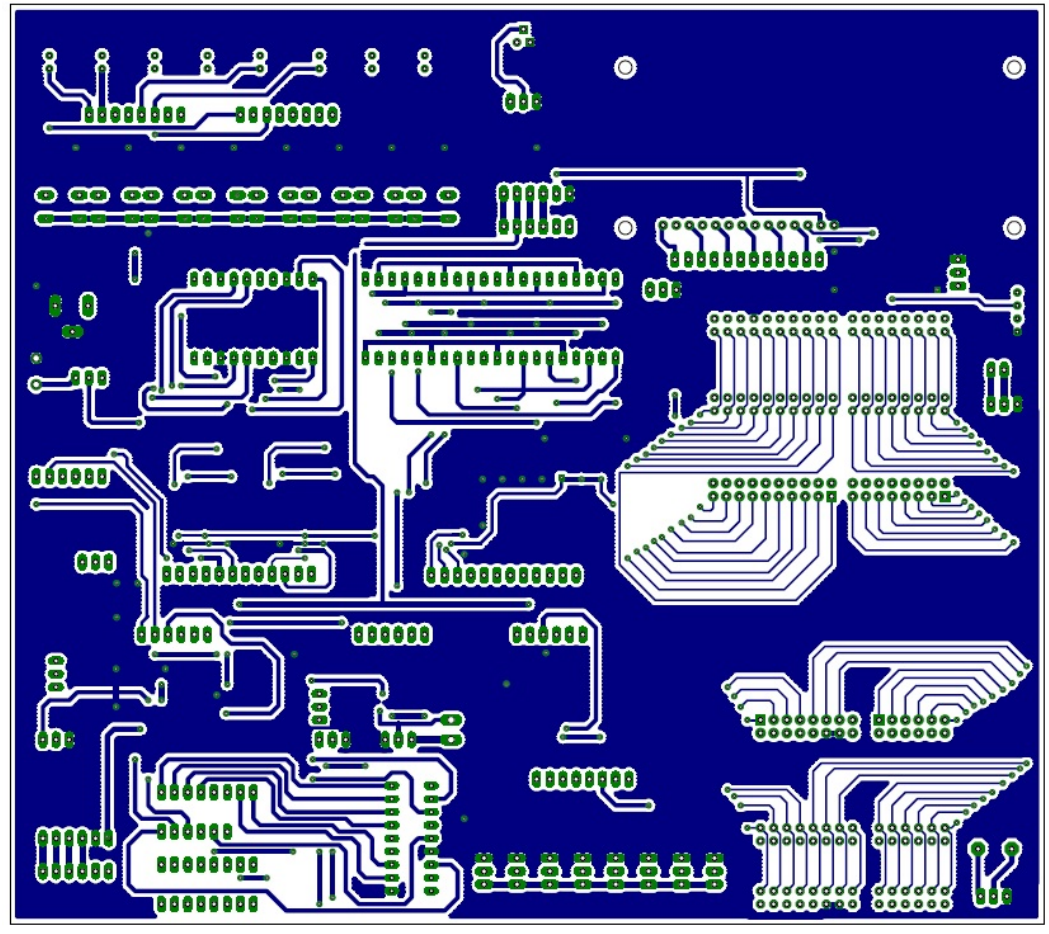
	Fecha	Nombre	Firma:	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
Dibujado	31/8/2015	Carlos E. Jarauta		
Comprobado				
Escala:	Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes Esquema eléctrico		Plano Nº: 1.5	Universidad Zaragoza
			Nº del alumno 386528	
			Curso PFC	



	Fecha	Nombre	Firma:	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
Dibujado	31/8/2015	Carlos E. Jarauta		
Comprobado				
Escala:	Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes Serigrafía de componentes		Plano Nº: 2	Universidad Zaragoza
			Nº del alumno 386528	
			Curso PFC	



	Fecha	Nombre	Firma:	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
Dibujado	31/8/2015	Carlos E. Jarauta		
Comprobado				
Escala:	Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes PCB cara TOP		Plano Nº: 3	Universidad Zaragoza
			Nº del alumno 386528	
			Curso PFC	



	Fecha	Nombre	Firma:	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
Dibujado	31/8/2015	Carlos E. Jarauta		
Comprobado				
Escala:	Sistema de Desarrollo basado en Freescale Freedom FRDM-KL25Z con fines docentes PCB cara BOTTOM		Plano Nº: 4	Universidad Zaragoza
			Nº del alumno 386528	
			Curso PFC	

Partlist

Exported from Definitiva.brd at 27/07/2015 12:56:00
EAGLE Version 7.2.0 Copyright (c) 1988-2014 CadSoft

Part	*Value*	*Package*	*Library*	*Part*	*Value*	*Package*	*Library*
ACELE	Acelerómetro	MA06-1	con-lstb	HUM	22-23-2021	22-23-2021	con-mole
C1	100nF	C1206	resistor	C2	100nF	C1206	resistor
C3	100nF	C1206	resistor	C4	100nF	C1206	resistor
C5	100nF	C1206	resistor	C6	100nF	C1206	resistor
C7	100nF	C1206	resistor	C8	100nF	C1206	resistor
C9	100nF	C1206	resistor	C10	10uF	153CLV-0505	rcl
C11	10uF	153CLV-0505	rcl	C12	10uF	153CLV-0505	rcl
C13	100nF	C1206	resistor	C14	100nF	C1206	resistor
C15	100nF	C1206	resistor	C16	100nF	C1206	resistor
C17	1uF	153CLV-0505	rcl	C18	1000uF	150CLZ-0810	rcl
C19	100nF	C1206	resistor	C20	10uF	153CLV-0505	rcl
DIS1	HD-H101	HDSP-M	displ-hp	DIS2	HD-H101	HDSP-M	displ-hp
DIS3	HD-H101	HDSP-M	displ-hp	DIS4	HD-H101	HDSP-M	displ-hp
DIS5	HD-H101	HDSP-M	displ-hp	DIS6	HD-H101	HDSP-M	displ-hp
IC1	4511MODD	TSOP	BCD7seg	IC2	4511MODD	TSOP	BCD7seg
IC3	4511MODD	TSOP	BCD7seg	IC4	74C922DN	DIL18	ana
IC5	24AA256SN	SO-08	microchip	IC6	LTC2622CM	MSOP8	linear-tec
IC7	AD7992	MSOP10	analog-devic	IC8	MCP454102	MSOP8	potmio
IC9	TMP36	TO127P	temper	IC10	LD1117S33	SOT223	regnew
INT1	TL32PO	TL3XPO	switch	INT2	TL32PO	TL3XPO	switch
INT3	TL32PO	TL3XPO	switch	INT4	TL32PO	TL3XPO	switch
INT5	TL32PO	TL3XPO	switch	INT6	TL32PO	TL3XPO	switch
INT7	TL32PO	TL3XPO	switch	INT8	TL32PO	TL3XPO	switch
J1	IN/OUT	MA08-2	tira16	J2	IN/OUT	MA10-2	tira20
J9	IN/OUT	MA08-2	tira16	J10	IN/OUT	MA06-2	tira12
JP1	JUMPER	JP2	jumper	JP2	JUMPER	JP2	jumper
JP3	JUMPER	JP2	jumper	LCD	LCD_16X2	TUXGR_16X2_R2	LCD
LED1	5mm	LED5MM	led	LED2	5mm	LED5MM	led
LED3	5mm	LED5MM	led	LED4	5mm	LED5MM	led
LED5	5mm	LED5MM	led	LED6	5mm	LED5MM	led
LED7	5mm	LED5MM	led	LED8	5mm	LED5MM	led
MICR1	Display1	MA12-1	con-lstb	MICR2	Display2	MA12-1	con-lstb
MICR3	Teclado1	MA08-1	con-lstb	MICR4	Teclado2	MA06-1	con-lstb
MICR5	LCD	MA12-1	con-lstb	MICR6	Memoria	MA06-1	con-lstb
MICR7	DAC	MA06-1	con-lstb	MICR8	ADC	MA06-1	con-lstb
MICR9	Potenciometro	MA06-1	con-lstb	MICR10	Acelerómetr	MA06-1	con-lstb
MICR11	Temperatura	MA06-1	con-lstb	MICR12	Pulsadores	MA08-1	con-lstb
MICR13	Interruptores	MA08-1	con-lstb	MICR14	LEDS	MA08-1	con-lstb
PCB2	FRDM-KL25Z	FRDM-KL25Z	micromod				
POT1	10K	3223G	pot	POT2	10K	3223G	pot
POT3	10K	3223G	pot	POT4	10K	3223G	pot
R1	10K	R1206	resistor	R2	10K	R1206	resistor
R3	10K	R1206	resistor	R4	10K	R1206	resistor
R5	10K	R1206	resistor	R6	10K	R1206	resistor
R7	10K	R1206	resistor	R8	10K	R1206	resistor
R9	820	R1206	resistor	R10	820	R1206	resistor
R11	820	R1206	resistor	R12	820	R1206	resistor
R13	165	R1206	resistor	R14	165	R1206	resistor
R15	165	R1206	resistor	R16	165	R1206	resistor
R17	165	R1206	resistor	R18	165	R1206	resistor
R19	165	R1206	resistor	R20	165	R1206	resistor
R21	165	R1206	resistor	R22	165	R1206	resistor
R23	165	R1206	resistor	R24	165	R1206	resistor
R25	165	R1206	resistor	R26	165	R1206	resistor
R27	165	R1206	resistor	R28	165	R1206	resistor
R29	165	R1206	resistor	R30	165	R1206	resistor
R31	165	R1206	resistor	R32	165	R1206	resistor
R33	165	R1206	resistor	R34	820	R1206	resistor
R35	820	R1206	resistor	R36	10K	R1206	resistor
R37	10K	R1206	resistor	R38	10K	R1206	resistor
R39	10K	R1206	resistor	R40	10K	R1206	resistor
R41	10K	R1206	resistor	R42	690	R1206	resistor
R43	6.8	R1206	resistor	R44	22K	R1206	resistor
R45	100K	R1206	resistor	R46	10K	R1206	resistor

Part	*Value*	*Package*	*Library*	*Part*	*Value*	*Package*	*Library*
R47	100	R1206	resistor	R48	10K	R1206	resistor
R49	10K	R1206	resistor	R50	10K	R1206	resistor
R51	100	R1206	resistor	R52	5K6	R1206	resistor
R53	100	R1206	resistor	R54	10K	R1206	resistor
R55	100	R1206	resistor	R56	10K	R1206	resistor
R57	100	R1206	resistor	R58	10K	R1206	resistor
R59	100	R1206	resistor	R60	10K	R1206	resistor
R61	100	R1206	resistor	R62	10K	R1206	resistor
R63	100	R1206	resistor	R64	10K	R1206	resistor
R65	100	R1206	resistor	R66	10K	R1206	resistor
R67	100	R1206	resistor	R68	10K	R1206	resistor
R69	100	R1206	resistor	R70	100	R1206	resistor
R71	100	R1206	resistor	R72	100	R1206	resistor
R73	100	R1206	resistor	R74	100	R1206	resistor
R75	100	R1206	resistor	R76	100	R1206	resistor
R77	220	R1206	resistor	R78	220	R1206	resistor
R79	220	R1206	resistor	R80	220	R1206	resistor
R81	220	R1206	resistor	R82	220	R1206	resistor
R83	220	R1206	resistor	R84	220	R1206	resistor
R85	100	R1206	resistor	SP1	AL60P	AL60P	buzzer
S1	SWI	B3F-10XX	switch-omron	SV1	OUTDAC	MA03-1	con-lstb
S2	SWI	B3F-10XX	switch-omron	SV2	INREFD/A	MA03-1	con-lstb
S3	SWI	B3F-10XX	switch-omron	SV3	INADC	MA03-1	con-lstb
S4	SWI	B3F-10XX	switch-omron	SV4	OUTPOT	MA03-1	con-lstb
S5	SWI	B3F-10XX	switch-omron	SV5	Potenciometro	MA03-1	con-lstb
S6	SWI	B3F-10XX	switch-omron	SV6	Temperatura	MA03-1	con-lstb
S7	SWI	B3F-10XX	switch-omron	SV7	Zumbador	MA03-1	con-lstb
S8	SWI	B3F-10XX	switch-omron	SV8	Humedad	MA03-1	con-lstb
TEMP	Temperatura	MA06-1	con-lstb	SV9	Presión	MA03-1	con-lstb
T1	BC857	SOT23	semicon-smd-ipc	TECL1	Teclado1	MA08-1	con-lstb
T2	BC857	SOT23	semicon-smd-ipc	TECL2	Teclado2	MA08-1	con-lstb
T2	BC857	SOT23	semicon-smd-ipc	TEST_J1	Pines_test	MA08-2	tira16
T3	BC857	SOT23	semicon-smd-ipc	TEST_J2	Pines_test	MA10-2	tira20
T4	BC857	SOT23	semicon-smd-ipc	TEST_J9	Pines_test	MA08-2	tira16
T5	BC847	SOT23	transistor-neu-to92	TEST_J10	Pines_test	MA06-2	tira12
T6	BC847	SOT23	transistor-neu-to92	TP1	PTR1PAD1-13Y	P1-13Y	testpad
T7	BC847	SOT23	transistor-neu-to92	TP2	PTR1PAD1-13Y	P1-13Y	testpad
T8	BC847	SOT23	transistor-neu-to92	V1	MPX12DPDV1	MPX2050DV	presion
T9	BC857	SOT23	semicon-smd-ipc	X1	INPUT AC	SCD-014-A	con-shiua
T10	BC857	SOT23	semicon-smd-ipc	X2	MKDSN1,5/2-5,08	MKDSN	con-phoenix
T11	BC857	SOT23	semicon-smd-ipc				
T12	BC847	SOT23	transistor-neu-to92				