UNIVERSIDAD POLITECNICA DE VALENCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

DESARROLLO DE UNA RED INALÁMBRICA DE BAJO COSTE PARA PEQUEÑAS APLICACIONES EMPOTRADAS

Proyecto Final de Carrera | Ferran Martí Sanvíctor

INDICE

INDICE

OBJETIVOS E INTRODUCCIÓN

Inicialmente el proyecto comprendía el diseño e implementación del software para que un módulo empotrado fuera capaz de comunicarse mediante una red inalámbrica básica con otros módulos y un host (PC), utilizando para ello pequeños módulos con transmisores y receptores por radiofrecuencia.

No obstante eso, se cambió el enfoque del proyecto, dándole mayor peso al diseño de una interfaz hardware. Dicha interfaz tendrá:

1. 5 entradas digitales (entradas rápidas)
2. 4 entradas analógicas (entradas lentas)
3. 2 salidas conmutadas (relés)
4. Un transceptor para comunicación serie mediante el protocolo RS-485.

El primer paso para el desarrollo del proyecto fue la elección de las herramientas y los componentes a que se iban a usar. Se tuvieron en cuenta diversas opciones en todos los aspectos, y comparando los puntos a favor con los puntos en contra de todas las partes comparadas, se eligió lo que parecía más adecuado para el proyecto, teniendo en cuenta las prestaciones que buscábamos para la interfaz y el posible incremento en la dificultad que nos podía causar la elección de un componente u otro.

Una vez seleccionados los componentes y las herramientas a usar, se hizo el diseño de la interfaz. Empezando por el esquema electrónico del circuito a implementar, la distribución de los componentes sobre la placa y el diseño del circuito impreso que compondrá el hardware que queremos implementar.

Con la teoría hecha, el siguiente paso fue la creación de la placa. Se imprimieron las dos caras que compondrían la placa hechas con el programa de diseño de PCB. Con las caras imprimidas se insoló una placa positiva virgen de doble cara para poder realizar el atacado con ácido y así obtener el dibujo de las pistas final sobre la placa. Con esto, el siguiente paso fue repasar las pistas, ya que al tener la PCB pistas muy finas, era probable que alguna presentara cortes, y por tanto no tuviera una buena continuidad. Después se realizaron los agujeros para los componentes, asegurándose que todos encajaban en ellos sin problemas, y cuando estaba la placa terminada, se hizo el montaje de las piezas con el soldador.

El último paso fue el testeo de la interfaz con un software mínimo creado para comprobar que las configuraciones de las entradas y salidas (circuito de entrada y salida) eran correctas y la interfaz tenía un buen funcionamiento.

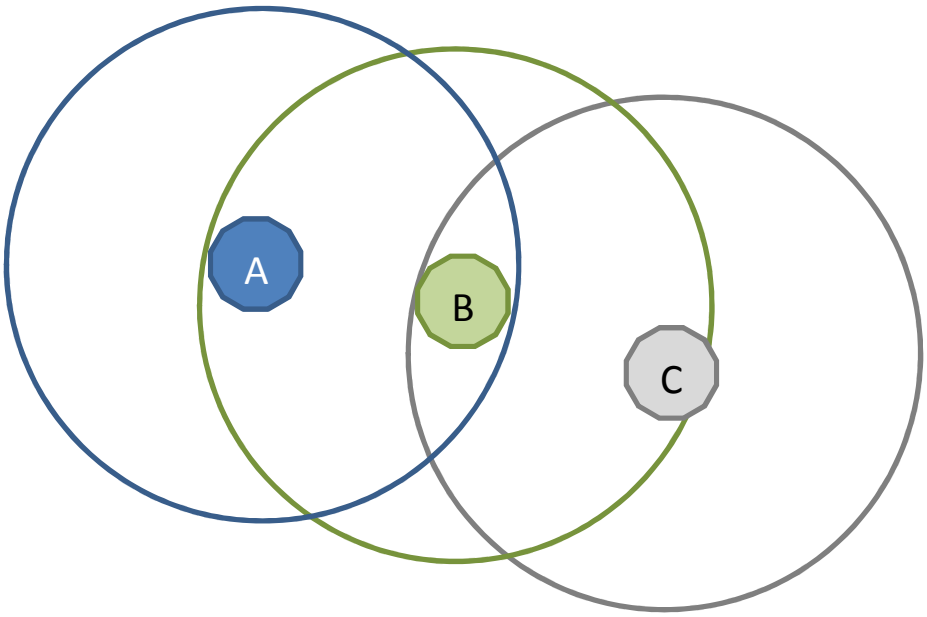
HARDWARE

Módulo por radio-frecuencia

Para la elección del módulo para la transmisión por radio-frecuencia, primero se tuvo que elegir el chip que se quería usar (ya que dependiendo del chip disponíamos de unos modelos u otros).

En este apartado solo nos centramos en dos fabricantes para la elección. Uno de ellos era Digi (www.digi.com) y el otro Texas Instruments (www.ti.com).

Digi nos ofrece una solución basada en el protocolo ZigBee, mientras que Texas Instruments nos presenta su propio protocolo para la transmisión inalámbrica, el SimpliciTI. La mayor diferencia entre el protocolo ZigBee y el SimpliciTI, es que ZigBee utiliza un método de transmisión enrutado, mientras que el SimpliciTI simplemente hace un envío por difusión.



Siendo A, B y C tres nodos usando ZigBee. A envía una trama a C, y si ésta no puede ser recibida porque C no está dentro del rádio de alcance de A, B se encargará de reenviar dicha trama, como si de un "puente" se tratara.

En cambio, si A, B y C son tres nodos usando el protocolo SimpliciTI, la trama de A nunca podría llegar hasta C, porque éste nodo está fuera de su alcance.

En comparación, ZigBee nos presenta un protocolo más versátil, ya que nos permite aumentar la distancia entre nodos siempre y cuando haya nodos que actúen como puentes o repetidores. Sin embargo, para nuestro propósito, SimpliciTI nos resulta más útil, ya que nos va a simplificar la tarea de la programación.

Otro aspecto que se valoró en la elección del módulo a usar fue su capacidad de ampliación, es decir, la cantidad de entradas y salidas a las que nos da acceso la placa. Eso es porque, aunque cada chip tenga sus entradas y salidas, los módulos en los que van montados puede que usen algunas de esas entradas para funciones propias (como pueden ser leds de estado), o que por el tamaño de la placa no se pueda dar acceso limpio (con un header, por ejemplo) a todas las entradas del chip.

En este caso, se compararon el módulo XBee (de Digi) y el módulo WSN-NK01 (de WSNVAL):

En el caso del módulo XBee, nos presenta una conexión con 20 pines, de los cuales solo 8 de estos 20 pueden ser usados como entradas o salidas, otro como solo entrada y dos más con una interfaz UART.

Cuando analizamos el módulo WSN-NK01, disponemos de 18 pines, de los cuales 14 pueden ser usados como entradas o salidas. Asimismo, estas 14 GPIO, también pueden ser configuradas como interfaz UART, lo que nos permite mayor versatilidad a la hora de diseñar la PCB y distribuir los componentes. Además, éste módulo nos permite la alimentación con 5 o con 3,3 voltios, lo cual es una ventaja ya que dependiendo del circuito final que posea la interfaz en desarrollo, podremos elegir que voltaje nos resulta más cómodo de suministrar para su alimentación.

Una vez comparados los factores de mayor peso para tomar la decisión, se decide continuar con el módulo WSN-NK01, ya que su mayor versatilidad con las I/O nos da una clara ventaja en el diseño de la PCB, y la sencillez del protocolo de transmisión nos facilitará la programación de una aplicación de testeo.

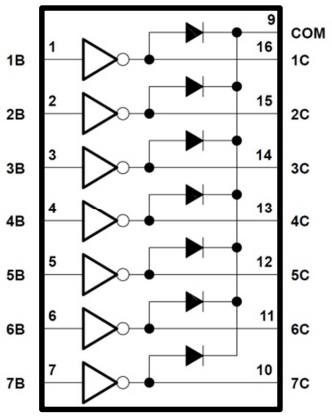
Con esto, debemos saber que, aunque el fabricante del módulo sea WSNVAL, el chip para la transmisión de datos que lleva integrado es de la compañía Texas Instruments, más concretamente el CC1110-F32, que transmite con una frecuencia de poco menos de 1 GHz (dispone de varias bandas de frecuencia, *315/433/868/915*, en este caso utilizaremos las más altas. Rango de frecuencia de 782 a 928 MHz).

Otro hardware

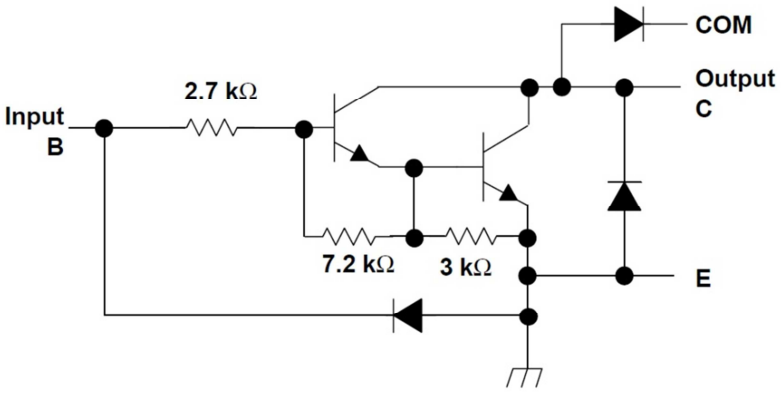
Matriz de transistores Darlington

Aparte del módulo para la comunicación por radio-frecuencia, la interfaz dispondrá de otro hardware que dotará al sistema características de otras características.

La interfaz que vamos a crear dispondrá de dos salidas conmutadas mediante relés. Puesto que la forma de activar o desactivar los relés es mediante los puertos GPIO, necesitaremos un sistema que nos convierta el "1" lógico que nos entrega el módulo en un voltaje suficiente para activar el sistema del relé y que este conmute su salida. Para ello vamos a usar el ULN2003A.



El ULN2003A es una matriz de transistores con una configuración Darlington. Con ello podemos conectar las salidas lógicas del módulo a las entradas del integrado, y como salida obtendremos el voltaje necesario para alimentar los relés que deben ser controlador por el módulo.

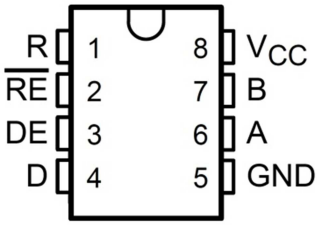


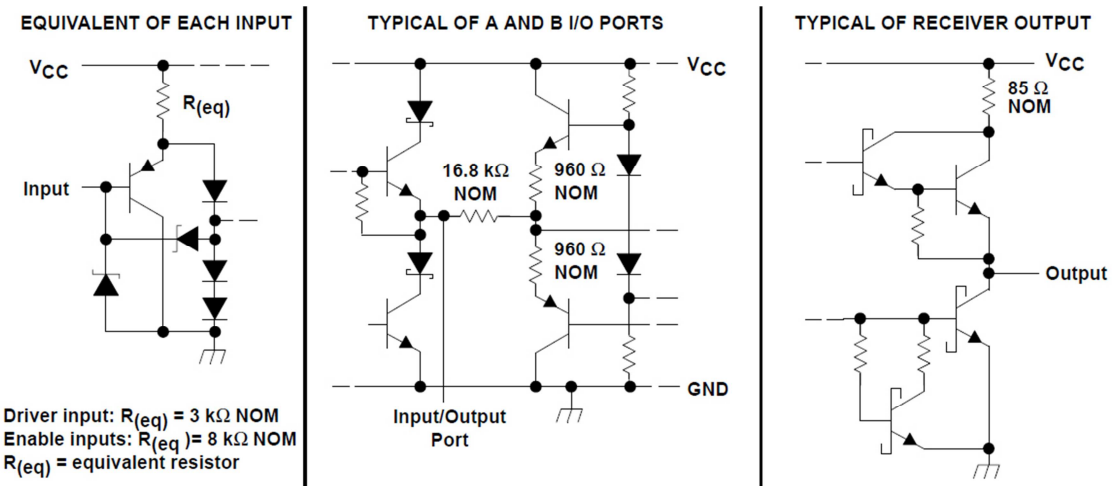
Este es el esquema electrónico de cada una de las entradas/salidas.

En nuestro caso, el ULN2003A será alimentado por 12v, ya que los relés que vamos a instalar necesitan dicha tensión para ser activados y poder conmutar la salida.

Transceptor

La interfaz que queremos desarrollar será compatible con el protocolo serie RS-485. Para ello, debemos dotar el circuito con un transceptor para poder conectar un sistema con comunicación por protocolo serie RS-485 con la interfaz UART del CC1110-F32 (el chip de nuestro módulo). Para este fin se ha escogido el SN75176B.

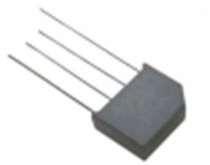




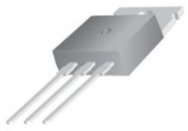
Alimentación

La interfaz ira alimentada con 12v en alterna, por lo que necesitaremos de un puente rectificador para convertirla en continua, y un regulador de tensión para adaptar la diferencia de potencial a la que va a necesitar la interfaz.

Como puente regulador se ha instalado un KBL06, y como regulador de tensión un LM7805, ya que tanto el transceptor, como el módulo, como los componentes pasivos funcionaran a 5v.



KBL06



LM7805

HERRAMIENTAS SOFTWARE

Programa de diseño de PCB

Se usó un programa específico de ordenador para el diseño de los esquemas electrónicos y los diferentes diseños para los circuitos de la PCB (si hicieron varios diseños, y se eligió el más apropiado, basándose su elección en el tamaño final de la placa y la distribución de los componentes).

Para la elección de dicho programa, se barajaron diversas posibilidades:

1. Protel (actualmente Altium Designer).
   http://www.altium.com/
2. Proteus
   http://www.labcenter.com/
3. OrCAD
   http://www.cadence.com/
4. Eagle
   http://www.cadsoft.de/

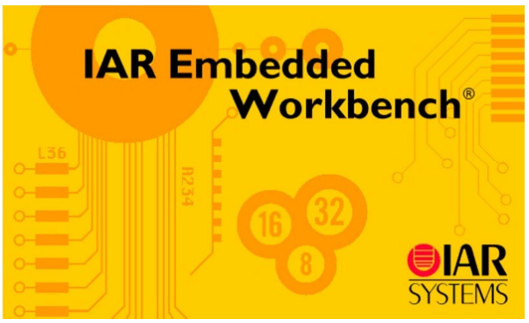
Para la elección del software de diseño se tuvo en cuenta la simplicidad de uso de la aplicación, las prestaciones ofrecidas, y la facilidad en el aprendizaje, ya que antes de empezar a usar la herramienta se deben adquirir ciertas destrezas para agilizar el diseño final.

La elección fue el uso del software de diseño Eagle (CadSoft Eagle, actualmente en su versión 5.10).

Eagle permite el uso del programa mediante diferentes modos de licencia. Se puede pagar una licencia, con lo que dispondríamos de la Standard Edition (edición estándar), que nos permite el uso completo del programa sin ningún tipo de restricción; también hay un tipo de licencia pensada para los aficionados (la Non-Profit License), que habilita prácticamente todas las funciones del programa, como si de una licencia estándar se tratara, pero bajo la condición de no poder lucrarse con el uso que se le dé, es decir, los circuitos y PCB diseñados bajo esta licencia no pueden ser vendidos. El tercer modo que nos ofrece el programa es el que vamos a usar; este nos permite el uso del software sin el pago de una licencia. Este modo de funcionamiento tiene ciertas limitaciones (la más importante de ellas es que limita el tamaño máximo de la placa que se está diseñando), pero estas limitaciones no van a suponer un impedimento para el desarrollo del proyecto, ya que en ningún caso vamos a requerir el uso de ninguna función de las bloqueadas por la licencia Freeware.

Entrono de programación

El chip elegido para la creación de la interfaz ha sido el CC1110-F32, de la compañía Texas Instruments. Al estar basado este chip en la arquitectura 8051 de Intel, se necesita un compilador que sea compatible con dicha tecnología. No obstante eso, la interfaz será compatible con un módulo específico de comunicación por RF. Este módulo (el WSN-NK01) tiene sus propias especificaciones, y aunque el chip central del módulo sea el CC1110-F32, no nos va a servir cualquier compilador para 8051, sino el que el fabricante nos recomiende.



En este caso, WSNVAL (fabricante del módulo que vamos a usar), nos recomienda usar el entorno de programación de IAR para 8051 (IAR Embedded Workbench®), disponible en la web http://www.iar.com/.

Como en el caso de la herramienta para el diseño de la PCB, IAR nos ofrece una solución abierta para el uso de su software. Podemos obtener una licencia para la versión Kickstart edition, que tiene limitaciones con el tamaño máximo del código que podemos escribir. Sin embargo, también nos ofrece una licencia de evaluación para la versión completa, que nos ofrece la posibilidad de usar el entorno sin ninguna restricción por un periodo de 30 días. Si se instala el software al inicio de las pruebas (no al inicio del proyecto), y se tiene una idea clara de lo que se quiere hacer, 30 días son suficientes para desarrollar una o varias aplicaciones de test para comprobar que el funcionamiento de la interfaz es el deseado y no hemos cometido errores en el proceso de diseño. Para el desarrollo de la interfaz y sus pruebas se ha usado la versión de evaluación de 30 días.

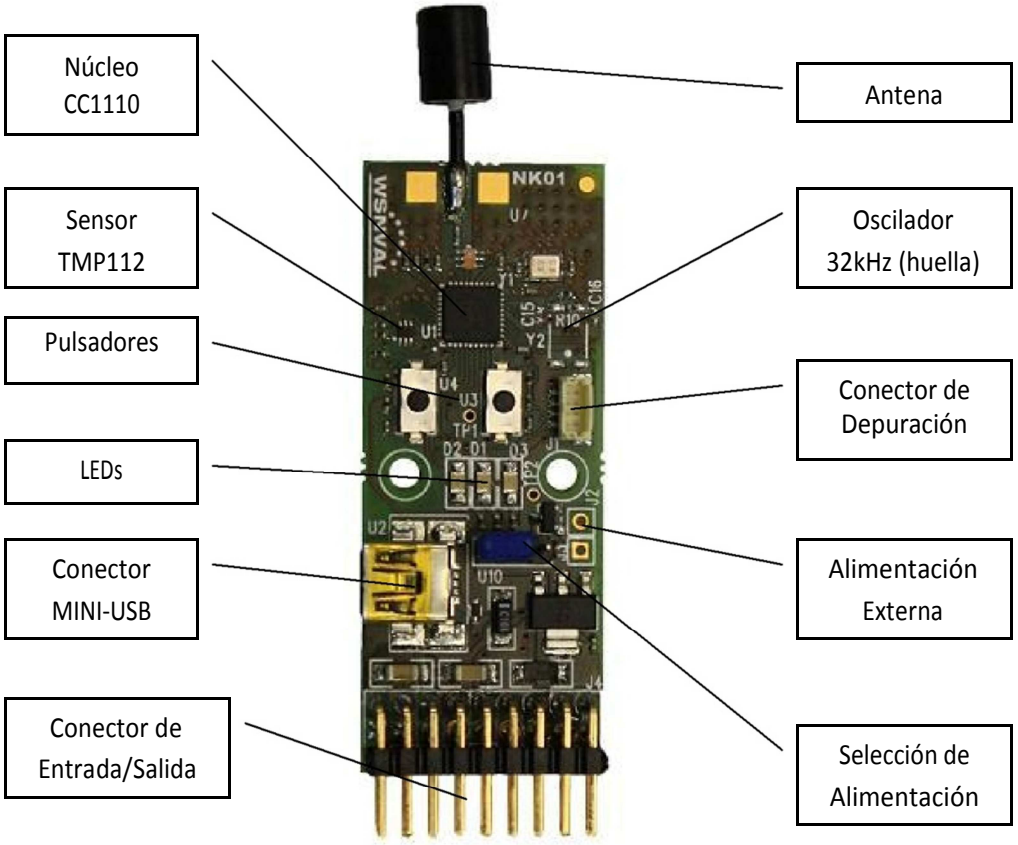
Los pasos a seguir para obtener el software y la licencia son muy simples:

1. Acceder a la web de IAR
2. Seleccionar Development solutions for 8051
3. Download evaluation versions of our products for 8051
4. Elegir entre 30-day evaluation edition (la usada en este proyecto) y la Kickstart edition
5. En cualquier caso, se ha de presionar el botón continuar, y ha de completarse el registro
6. Se recibirá un correo de confirmación en la dirección facilitada durante el registro
7. Al presionar el Link del correo, se nos muestra el “License Number”, la “License Key”, y el link de descarga de la aplicación (tanto desde servidor HTTP como por servidor FTP).

MODULO (WSN-NK01)

Descripción del Hardware

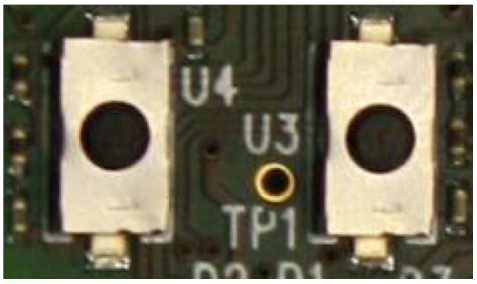
En un formato 23 mm x 58 mm se integra una solución completa de RF, junto con elementos como pulsadores, LEDs, un sensor de temperatura, un conector de entrada/salida y un sistema de alimentación flexible que permite el funcionamiento a partir de baterías o a partir de un USB.



Conector de Entrada/Salida

Se dispone de un conector de paso 2.54 mm en la parte inferior donde se proporciona interconexión con la mayoría de los pines del microcontrolador. Asimismo, se pueden alimentar dispositivos con niveles de alimentación de 5 y 3.3V utilizando los pines seleccionados.

Pulsadores



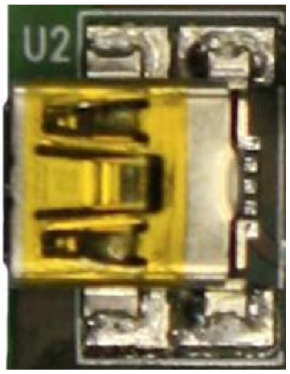
El modulo de evaluación de WSNVAL dispone de dos pulsadores (U3 y U4), ambos activos a nivel bajo, con dos funciones muy diferenciadas:

- **Pulsador de Propósito general U3:** La acción sobre este elemento fuerza un nivel lógico '0' en el pin P1_4 del microcontrolador; en reposo el nivel lógico en este pin es igual a '1'. La variación de este pin puede ser configurada como una fuente de interrupción reconocible por el microcontrolador.
- **Pulsador de RESET U4:** La acción sobre este componente fuerza el reinicio del microcontrolador.

Alimentación del módulo

Conector de Mini-USB

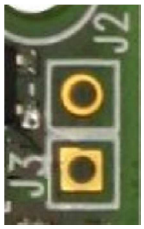
La alimentación del módulo de evaluación se puede realizar a través de una conexión Mini-USB estándar. De este modo conectando el dispositivo a un puerto USB de un PC, todos los dispositivos incluidos en el módulo funcionarán de una manera adecuada y todos los pines de alimentación presentes en el conector de Entrada/Salida estarán disponibles a las tensiones especificadas.



Esta interfaz NO proporciona conectividad entre el microcontrolador y el PC conectado. WSNVAL dispone de un dispositivo específico para realizar dicha función denominado WSN-SUM-USB. Por favor, diríjase a la documentación de este dispositivo en caso de necesitar más información.

Alimentación Externa (Conector)

Los pines denominados como J2 y J3 facilitan la conexión de una fuente de energía externa como baterías o fuentes de tensión reguladas de una manera sencilla, cómoda y segura.

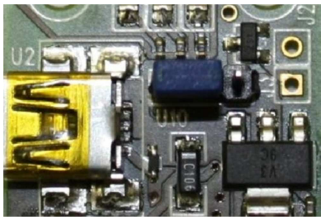


El valor de tensión conectado a dichos pines (tensión positiva a J2 y tensión de referencia a J3) servirá para alimentar todos los dispositivos conectados a tensiones de 3.3 V. La tensión de 5V disponible en el conector de Entrada/Salida no será válida en caso de alimentar únicamente el dispositivo con este conector.

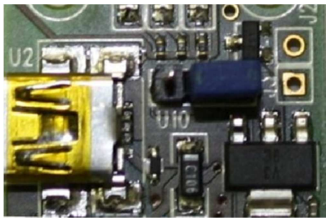
Todos los dispositivos incluidos en el módulo de evaluación funcionarán correctamente con valores de alimentación de entre 2 y 3.6 V. Sin embargo, en caso de conectar alguna circuitería externa deberá tenerse en cuenta el rango adecuado para dicha aplicación.

Selector de alimentación

Para seleccionar la fuente de alimentación a utilizar por el dispositivo se dispone del conector U10 de paso 2.54 mm entre sus pines.



Alimentación a partir del conector Mini-USB

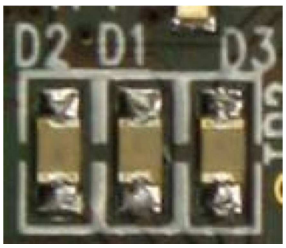


Alimentación a partir de la alimentación externa

Colocando pequeño JUMPER entre el conector central y uno de los extremos se selecciona como fuente de alimentación de 3.3V la procedente del conector Mini-USB (conexión entre los pines central e izquierdo) o del conector de alimentación externa (conexión entre los pines central y derecho).

LEDs

El dispositivo incorpora 3 diodos luminosos (LEDs) de colores rojo, amarillo y verde. Estos diodos están conectados en configuración de ánodo común con tres puertos del microcontrolador.



Nombre	Color	Pin CC1110
D1	Amarillo	P0_1
D2	Rojo	P0_0
D3	Verde	P0_6

Para un correcto funcionamiento de cada uno de los diodos, el pin correspondiente debe estar configurado como SALIDA DIGITAL en configuración PUSH-PULL. Estableciendo la salida a nivel ALTO el LED permanecerá APAGADO, mientras que si la salida está a nivel BAJO en LED lucirá.

Sensor de Temperatura

Se ha incorporado al dispositivo un sensor de temperatura TMP112 de Texas Instruments ®. Este sensor de temperatura además de una precisión de 0.5ºC, dispone de una resolución de 12 bits y una interfaz de comunicación serie SMBus ™.

El diagrama de conexiones se resume en la tabla siguiente. La alimentación del chip a través de la línea V+ se realiza mediante un pin (P1_2) del microcontrolador. Para un correcto funcionamiento de este chip el pin P1_2 del microcontrolador debe estar configurado como SALIDA DIGITAL en configuración PUSH-PULL y su valor ha de estar fijado a nivel alto. En otro caso el chip TMP112 no estará correctamente alimentado.

TMP112 Pin	Pin CC1110
V+	P1_2
ADD0	GND
SCL	P1_0
SDA	P1_3
ALERT	P1_1

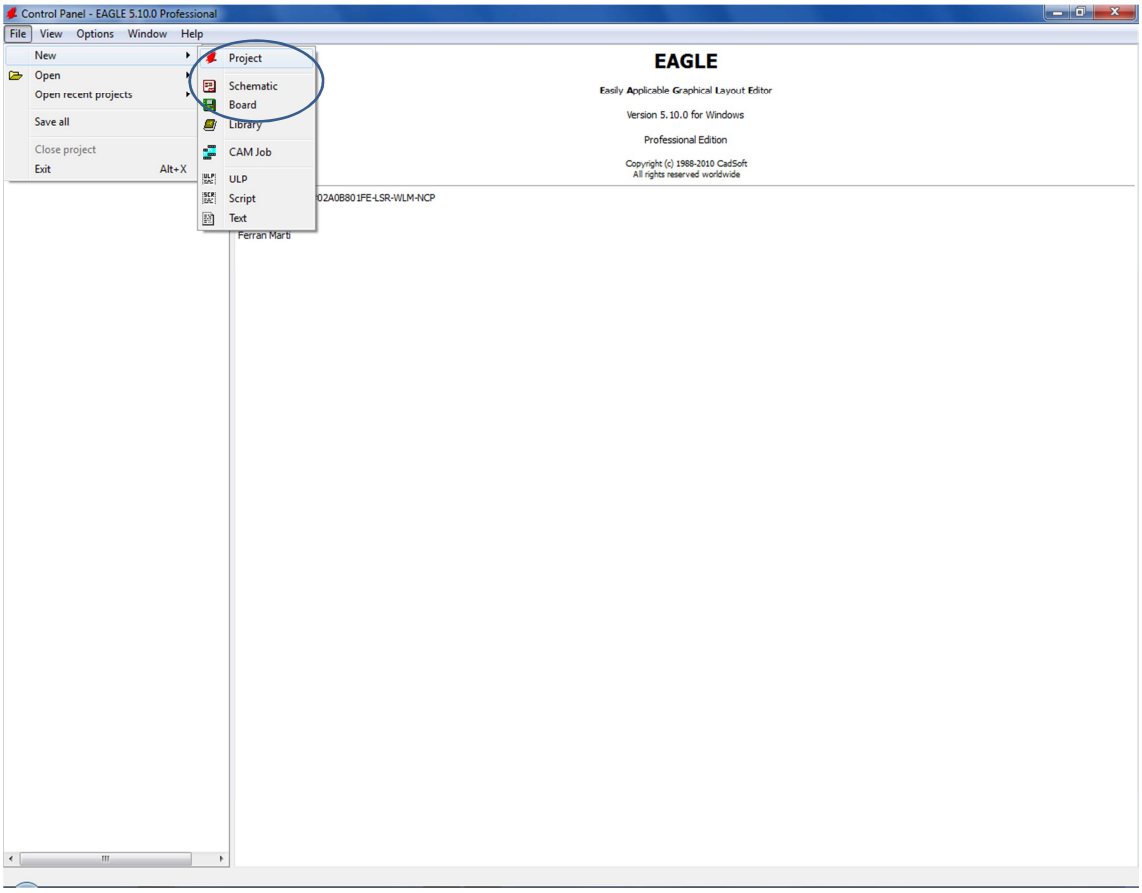
Asimismo, las líneas SCL, SDA y ALERT del sensor de temperatura están conectadas a la línea V+ mediante un PULL UP adecuado siguiendo las recomendaciones de este dispositivo.

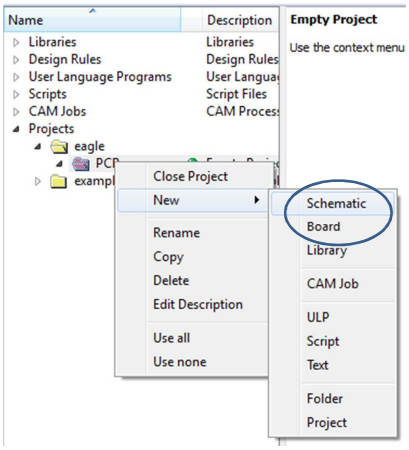
Para la comunicación entre el microcontrolador y el sensor de temperatura se sigue el protocolo SMBus ™. Se proporciona una librería software para realizar la comunicación de manera adecuada y siguiendo el esquema de conexión expuesto anteriormente.

ESQUEMA ELECTRONICO

Entorno Eagle

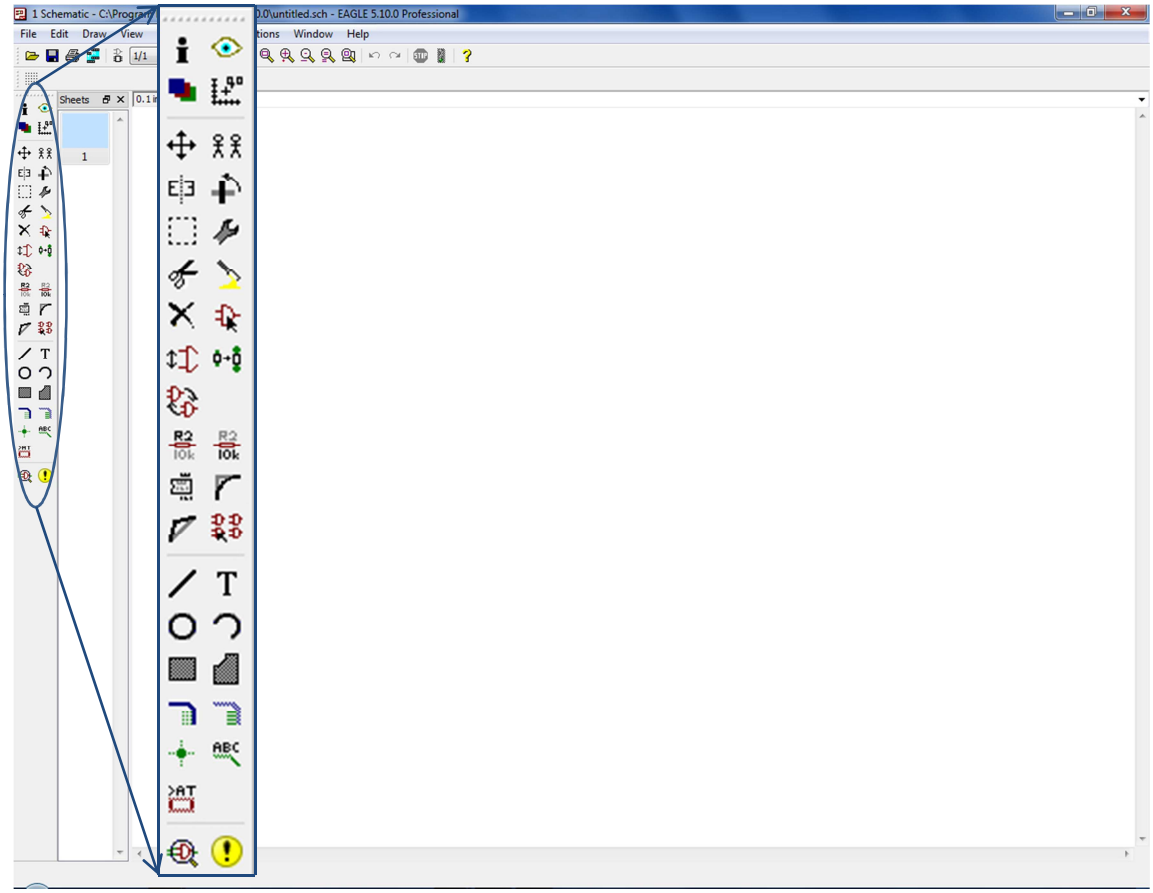
Para la realización del esquema electrónico se ha trabajado con el entorno Eagle. Para empezar se debe crear un proyecto, y crear dentro el archivo schematic (no es necesario crear un proyecto, se puede crear directamente el schematic y después el board).





Para crear el archivo schematic dentro del proyecto, este debe de estar seleccionado y activo. Se debe presionar el botón secundario del ratón sobre él y añadir el nuevo archivo.

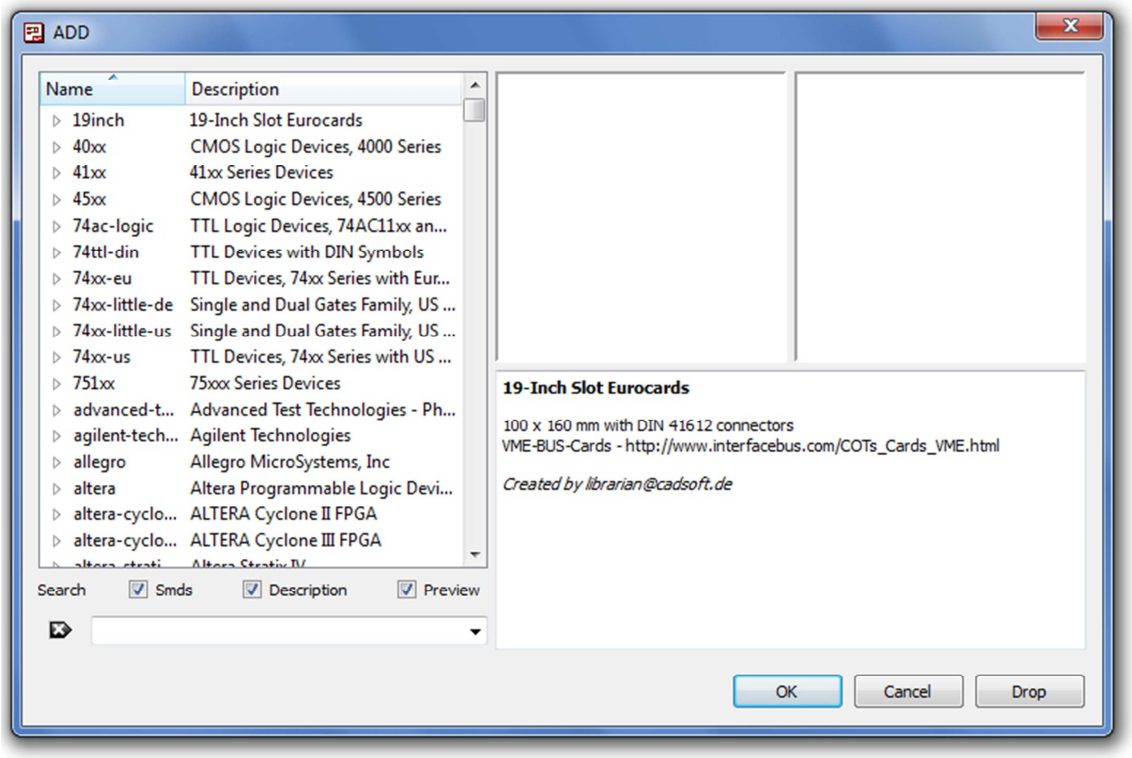
Una vez se está trabajando con el editor del Eagle, se dispone de una barra de herramientas lateral, que nos da acceso a todas las funciones que vamos a necesitar para la creación de nuestros diseños:



Las principales herramientas que vamos a usar son las de “mover”, “copiar”, “espejo”, “rotar”, “pegar”, “añadir”, “nombre”, “valor”, “línea y “texto”; que se corresponden con los siguientes símbolos:

Mover		Copiar	
Espejo		Rotar	
Pegar		Añadir	
Nombre		Valor	
Línea		Texto	

Al abrir el schematic nuevo tendremos el documento vacío. Para empezar a añadir componentes debemos pulsar sorbe el botón añadir (add). Al pulsarlo se abrirá un cuadro de dialogo:



En el cuadro de dialogo “ADD”, tenemos varias zonas. La primera es la lista (un explorador) con todas las librerías disponibles, y dentro de cada librería dispondremos de los componentes disponibles. A su derecha veremos que hay tres cuadros (2 arriba y un tercero más grande debajo). En el primero se muestra el símbolo para hacer el esquema electrónico, en el segundo el footprint asociado para la PCB, y en el inferior se muestra una descripción del componente.

Debajo del explorador de librerías, tenemos un buscador (Search). Aquí podemos realizar un filtrado de los componentes que se deben mostrar en el explorador de componentes. Para realizar filtrados se debe introducir el nombre del componente que deseamos insertar en nuestro archivo, siempre en inglés, ya que la interfaz del programa no dispone de otro idioma (por ejemplo, para buscar resistencias se debe introducir la palabra “resistor”, o para condensadores la palabra “capacitor”).

Una vez se tiene el diseño del esquema electrónico terminado, se debe procedes a la creación de la PCB. Para ello, Eagle dispone de una herramienta que automatiza la dicha creación, pero también se puede crear manualmente. En el caso de nuestro proyecto, la PCB se creó manualmente, pero para usar la herramienta de automatización, solo se debe pulsar el botón “Board” situado en la barra de herramientas del entorno.

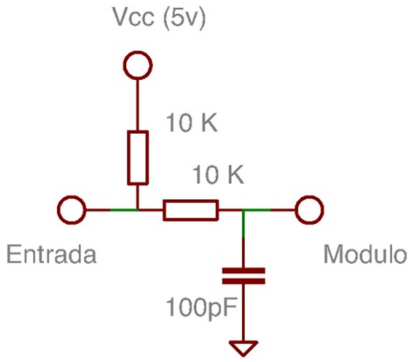
Entradas

La interfaz dispondrá tanto de entradas digitales, llamadas entradas rápidas, como de entradas analógicas, llamadas entradas lentas. Para el diseño de dichas entradas, el datasheet del módulo nos proporciona el esquema a seguir, dejando los valores de las resistencias y el condensador en blanco, ya que cada tipo de entrada tendrá una configuración especifica.

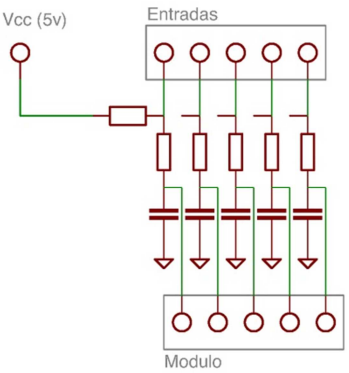
Debido a que la interfaz dispondrá de 5 entradas rápidas y 4 lentas idénticas, y para reducir el tamaño de la PCB, se van a usar arrays de resistencias de nodo común.

Entradas Rápidas

Valores de los componentes:

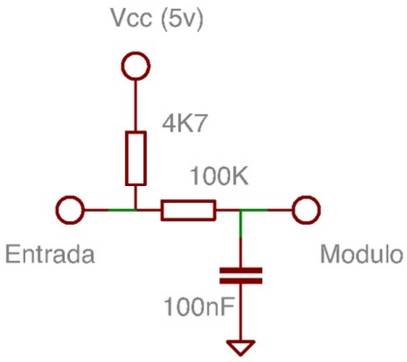


Array de resistencias:

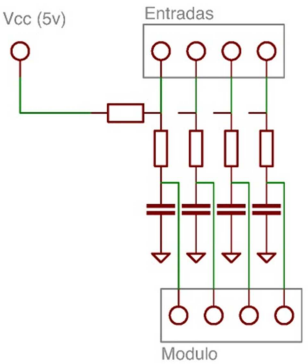


Entradas Lentas

Valores de los componentes:



Array de resistencias:



Salidas

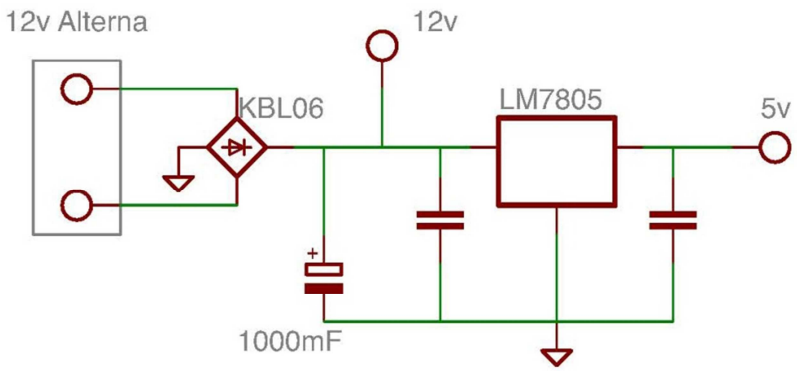
En el caso de las salidas es más sencillo. No hay esquema ya que su conexionado es muy sencillo. Se debe conectar la salida del módulo (que proporcionara valores lógicos) al ULN2003A. Este se encargará de convertir los 5v del valor lógico "1" en los 12v que necesitan los relés para funcionar.

Chip Serie

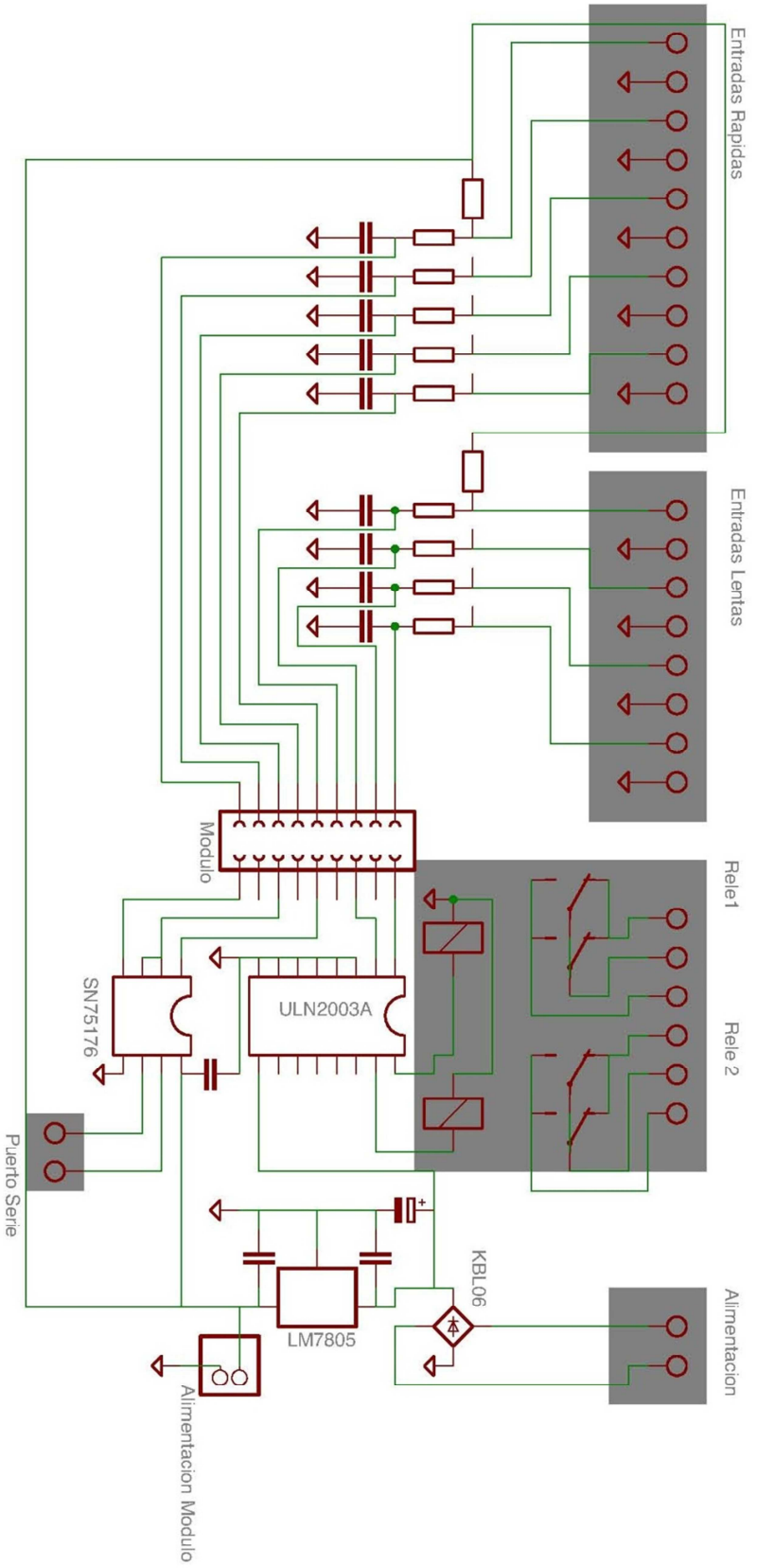
Para evitar un posible mal funcionamiento, en la entrada del chip SN75176B (chip para el protocolo serie), se conecta un condensador de desacoplo de 100nF, lo más cerca posible del chip, conectando uno de los extremos del condensador a la alimentación del chip y el otro a masa.

Alimentación

La alimentación del módulo se realizará mediante 12v en corriente alterna. Para poder usar dicha alimentación se requiere el montaje de un puente rectificador, de un condensador y de un regulador de voltaje para poder convertir esos 12v en C.A. a 5v en D.C., que será la alimentación que usara la interfaz.



Esquema completo



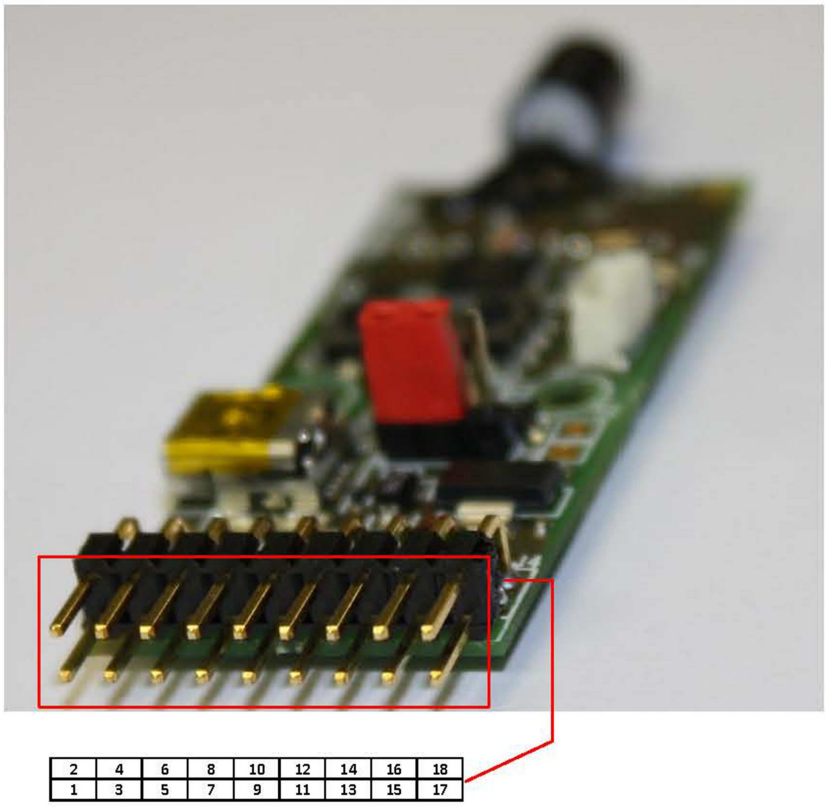
PCB

Módulo

Antes de empezar con el diseño de la PCB se debe tener claro la configuración que se le va a dar el módulo.

En este caso sabemos que la interfaz dispondrá de 5 entradas rápidas, 4 lentas, 2 salidas y un transceptor RS-485. Con esto debemos elegir en que puertos del chip que lleva nuestro módulo se va a configurar cada cosa.

Para ello necesitamos el pinout del módulo:



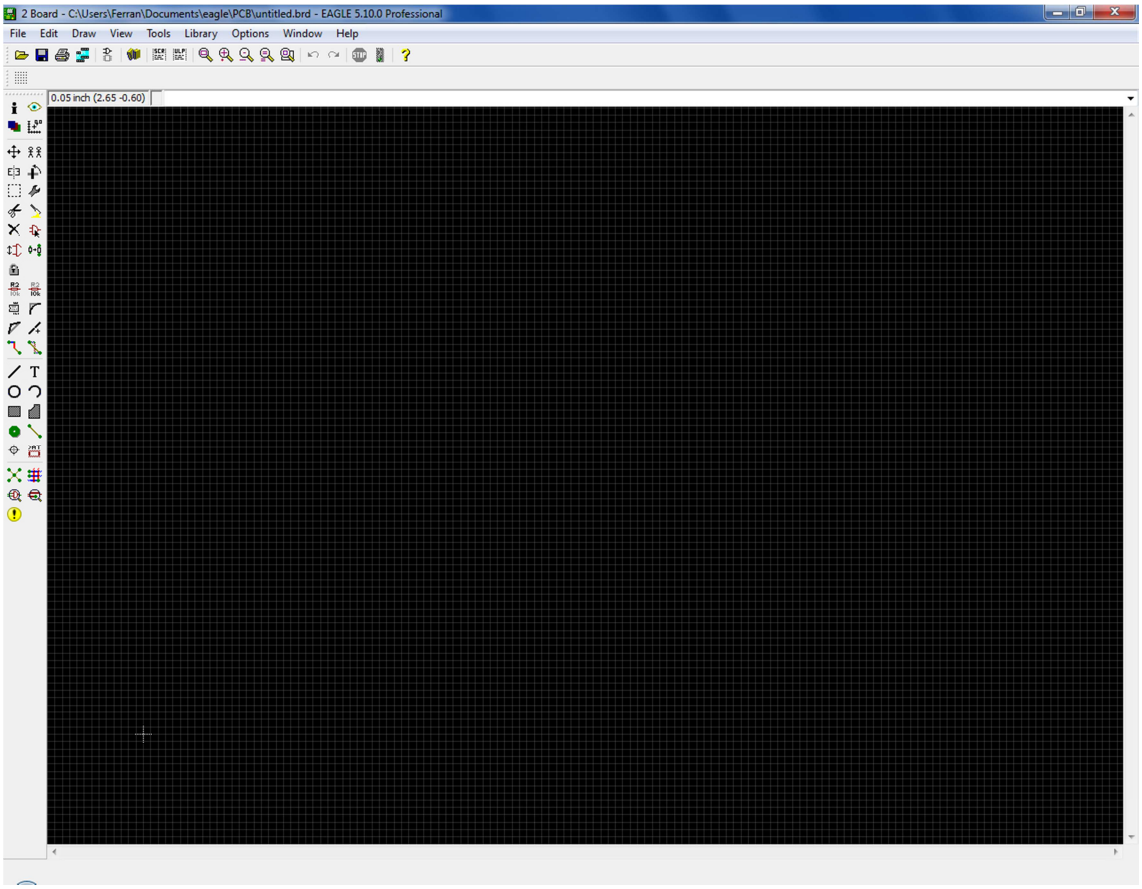
PIN	FUNCION	PIN	FUNCION
1	P0_2	2	P1_0
3	+ 5v	4	P1_1
5	P0_3	6	P1_2
7	+ 3.3v	8	P1_3
9	P0_4	10	P1_4
11	GND	12	P1_5
13	P0_5	14	P1_6
15	DNG	16	P1_7
17	P0_7	18	P2_0

Una vez analizadas las posibilidades se va a configurar de la siguiente manera:

- Entradas rápidas: P1_4, P1_5, P1_6, P1_7, P2_0
- Entradas lentas: P1_0, P1_1, P1_2, P1_3
- Salidas (relés): P0_5, P0_7
- Transceptor: P0_2, P0_3, P0_4

Entorno Eagle

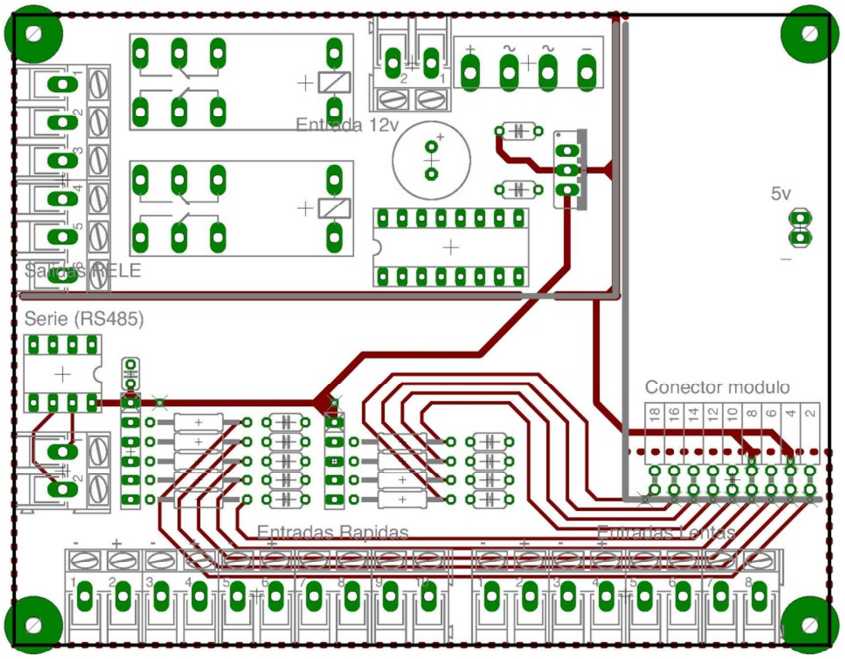
El entorno Eagle para el diseño de PCB es idéntico al de creación de esquemas electrónicos, salvo por el fondo, que esta vez es negro, y por algunas herramientas del panel lateral:



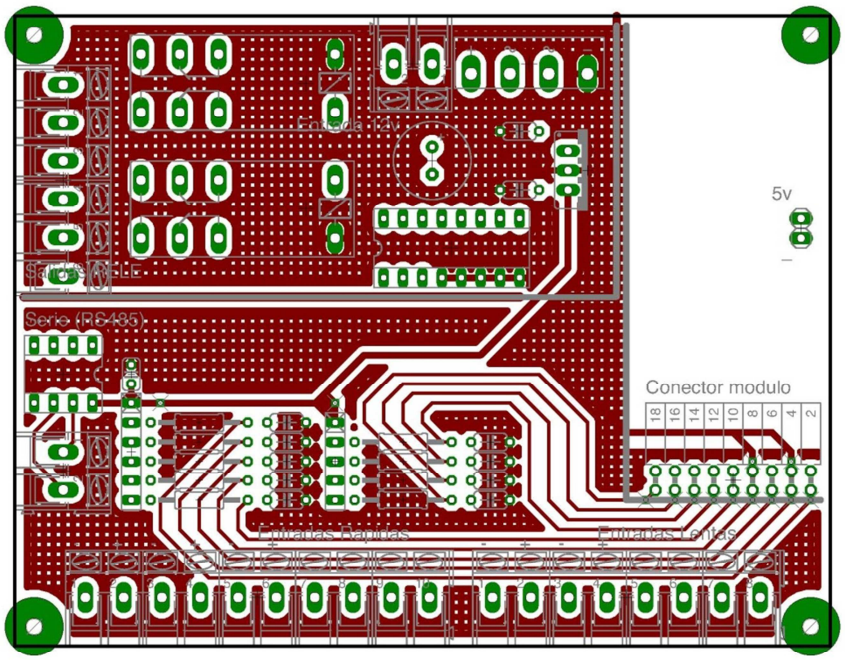
Para la distribución de los componentes y las pistas no se va a seguir ninguna distribución específica, simplemente se intentará hacer para que la PCB quede del menos tamaño posible.

Cara superior PCB (top layer)

Aquí se puede ver la distribución de los componentes y las pistas de la cara superior de la PCB:

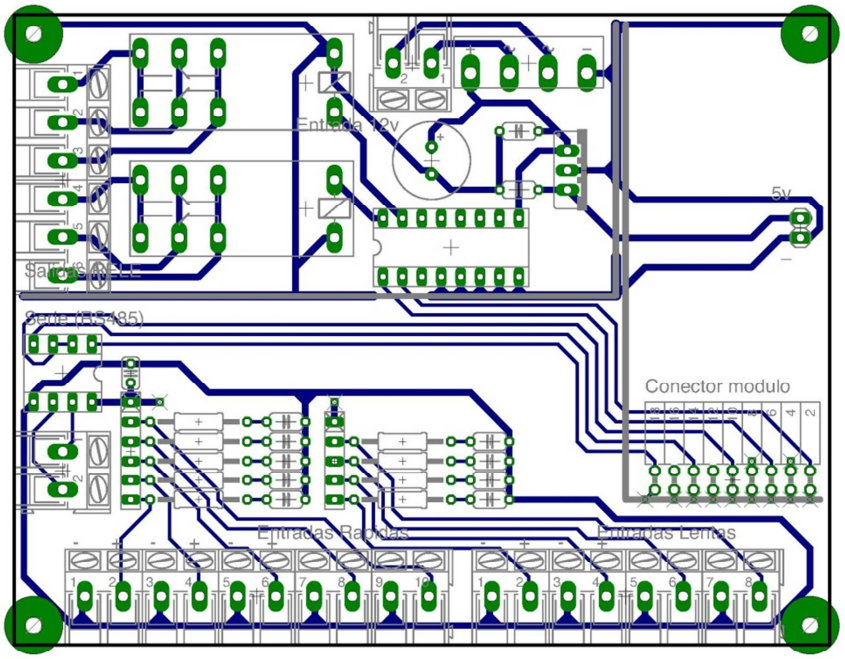


La siguiente imagen muestra la cara superior con los componentes, las pistas y el plano de masa:

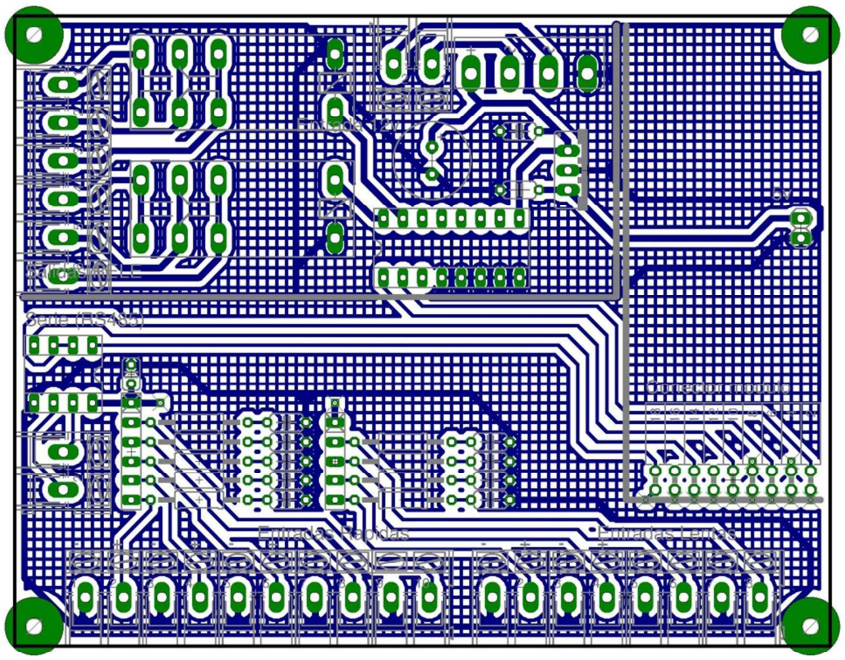


Cara inferior PCB (bottom layer)

Aquí se puede ver la distribución de los componentes y las pistas de la cara inferior de la PCB:



La siguiente imagen muestra la cara inferior con los componentes, las pistas y el plano de masa:



Renderizado

El software Eagle posee una herramienta para el renderizado de sus diseños. Realmente es una aplicación aparte, que posteriormente usa el programa povray para dibujar el diseño creado.

La descarga de la aplicación es a través de la web del creador (Eagle3D), y el povray se descarga de su página oficial (povray).

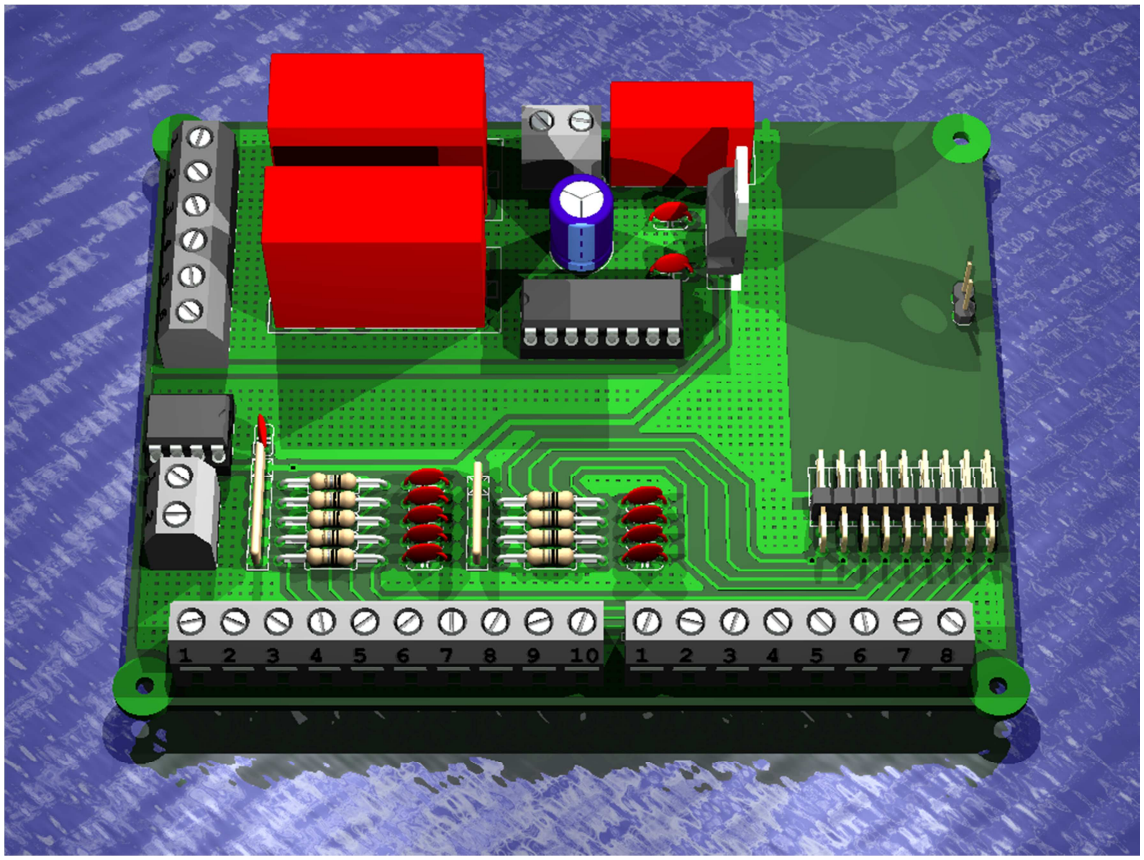
Para generar el archivo que deberá ser cargado con el povray, se debe ejecutar la herramienta de Eagle 3D desde la opción:

File
- ↘ Run…

Y guardar el archivo en la ruta deseada.

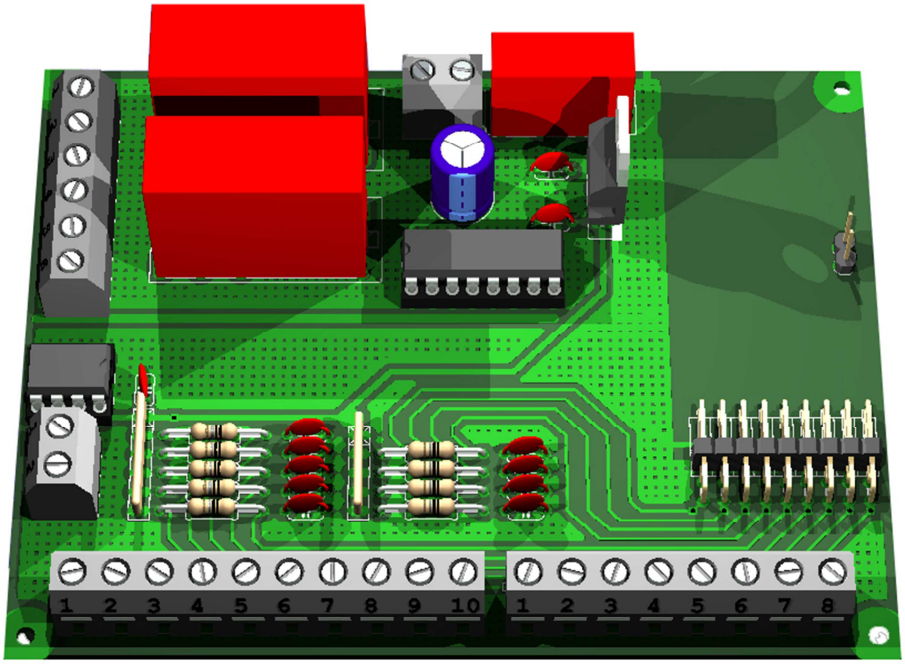
Posteriormente se debe abrir el archivo generado (.pov) con el programa povray, y presionar sobre el botón "Run". Automáticamente se abrirá una ventana con la imagen, mostrando el resultado, y se generará una imagen ".bmp" en la misma ruta donde se encuentra el archivo ".pov".

Este es el resultado después de renderizar el diseño del proyecto:



Debido a que no es un programa específicamente diseñado para el renderizado de PCB, sino que es una “adaptación”, existen algunas limitaciones en cuanto a diseño de fondo y de componentes.

La siguiente imagen es el mismo renderizado de la PCB, con el fondo eliminado mediante un software de retoque de imagen:



Los bloques rojos se corresponden con los relés y con el puente KBL06. Aparecen como bloques rojos debido a que en las librerías del software Eagle 3D no están definidos dichos componentes.

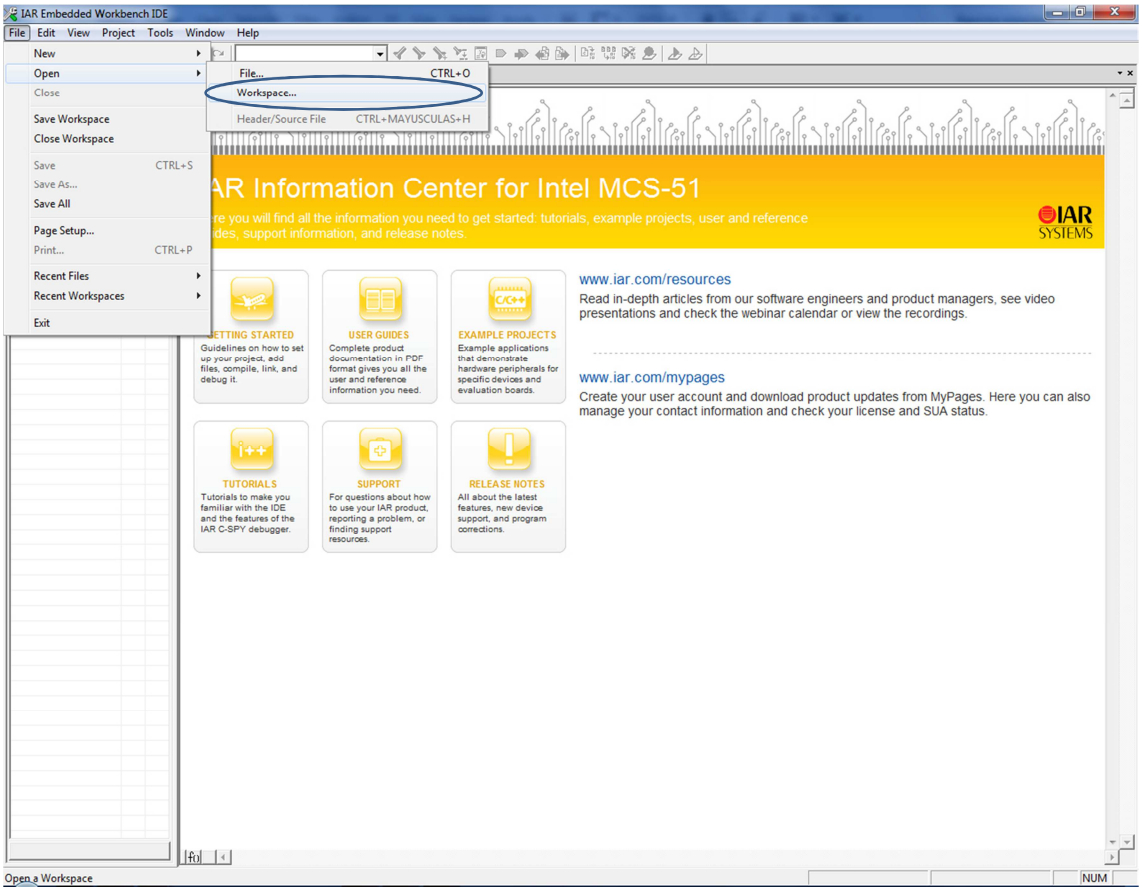
TEST DE LA INTERFAZ

Test de la Interfaz

Para el testeo de la interfaz se desarrolló una pequeña aplicación. La aplicación habilita las interrupciones en el puerto P1, más concretamente usa la entrada 4 de puerto P1 (P1_4). El programa usa la entrada rápida 1 de la interfaz (que a la vez se corresponde con el pulsador U4 integrado en el módulo), y cuando en esta se produce una interrupción, en otro nodo activa o desactiva la relé 1 y los leds (led rojo siempre encendido (led de control), cuando se activa la relé se enciende el led verde y se apaga el amarillo, y cuando se apaga la relé se enciende el led amarillo y se apaga el verde).

Para crear la aplicación se usó el IAR Embedded Workbench® para arquitectura 8051, ya que el cc1110 está basado en dicha arquitectura. También se debe tener instalado el SimpliciTI, ya que es la librería para la comunicación RF de Texas Instruments (el chip que lleva el módulo que se está usando).

Se debe abrir el espacio de trabajo del programa para el test:

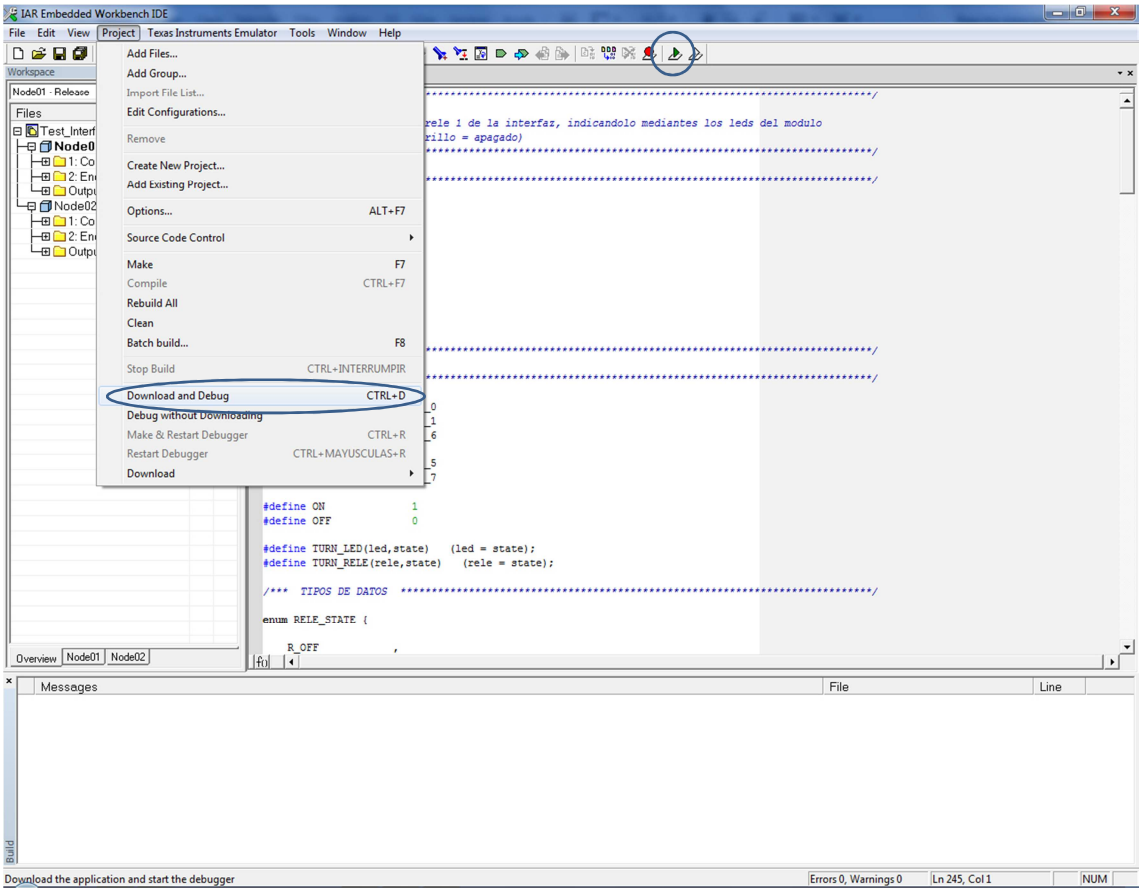


Para compilar en código y descargarlo al módulo:

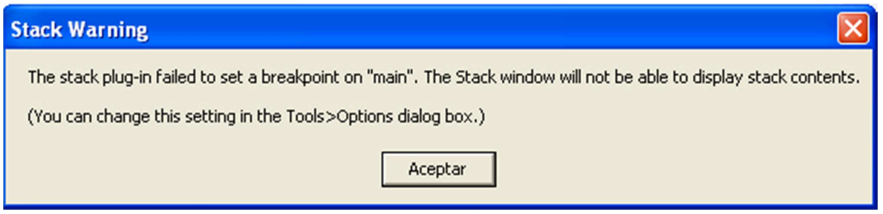
Project

↘ Download and Debug CTRL+D

O presionando en el botón que hay en la barra de herramientas 



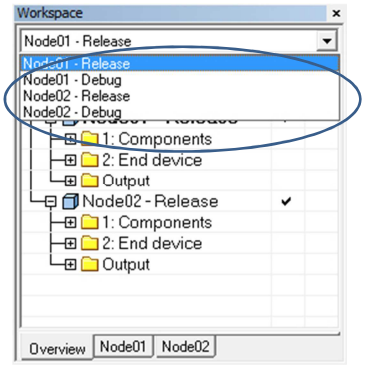
Si al realizar este proceso da un mensaje de advertencia como el siguiente:



No hay problema, el módulo se habrá programado correctamente. Esta advertencia es causada por la configuración con la que es descargado el programa al módulo. IAR nos crea dos configuraciones para el proyecto, la “debug” y la “release”. En la configuración debug no muestra ninguna advertencia, simplemente carga el código en el módulo y funciona en modo depuración, con el programador (o depurador) del módulo conectado al PC.

Si se carga la configuración release nos mostrara la advertencia, pero esto no supondrá ningún problema, ya que al desconectar el módulo del programador, éste se habrá instalado correctamente.

Para cambiar la configuración con la que estamos trabajando (debug o release), hay un desplegable en la parte superior de la barra del workspace. Simplemente desplegamos el menú y elegimos la configuración.



Programación del módulo

El lenguaje de programación que se usara es C, ya que nos facilita la escritura del código comparado con el lenguaje ensamblador.

Para poder tener un acceso directo a los puertos del chip (escribiendo solo P0_1, por ejemplo) es necesario cargar la librería (.h) correspondiente. En este caso, como estamos usando el CC1110-F32, será necesario cargar dicha librería:

```
#include <ioCC1110.h>
```

Para simplificar la programación, también es interesante definir las salidas que van a ser usadas. En este caso, definimos los leds (que van integrados en el módulo) y lo relés que tiene la interfaz:

```
#define LEDRED        P0_0
#define LEDYELLOW     P0_1
#define LEDGREEN      P0_6

#define RELE1         P0_5
#define RELE2         P0_7
```

También es interesante definir los estados posibles de los leds o los relés, así como las operaciones para cambiar su estado:

```
#define ON            1
#define OFF           0

#define TURN_LED(led,state)   (led = state);
#define TURN_RELE(rele,state)   (rele = state);
```

Ha de definirse las “direcciones” de los nodos, ya que actuara como identificador, para que un módulo no active una de sus salidas, sino la salida del otro nodo:

```
#ifdef NODE_01
static addr_t     Destination = { 0x10 , 0x00, 0x00, 0x02 };
#else
static addr_t     Destination = { 0x10 , 0x00, 0x00, 0x01 };
#endif
```

Se tiene que configurar las interrupciones para que puedan ser manejadas:

```
#pragma vector=P1INT_VECTOR
__interrupt void P1_external_ISR (void) {

  if ((P1IFG & 0x10) == 0x10) {                 /* Si el origen de P1_4 */

    Pushed = TRUE;
    P1IFG &= ~0x10;
  }

  P1IF = 0;
}
```

Dentro de la función PIN_config se deberán configurar los puertos para que sean usados como entradas o salidas, y para habilitar las interrupciones en los puertos que lo requieran, a la vez que configurar el estado de los leds por defecto:

```
  /* Configuracion de P0 */

  P0SEL = 0x00;                         /* P0 como I/O */
  P0DIR = 0xFF;                         /* Configurados como salida */

  TURN_LED(LEDRED, OFF);
  TURN_LED(LEDGREEN, ON);
  TURN_LED(LEDYELLOW, ON);

  /* Configuracion de P1 */

  P1DIR |= ~0x10;                        /* P1_4 como entrada */
  P1INP |= 0x10;                        /* Entrada triestado */
  PICTL &= ~0x02;
  P1IEN |= 0x10;                        /* Activa la mascara de interrupciones para P1_4 */
  IEN2  |= 0x10;                        /* Activa las interrupciones de la CPU para el puerto P1 */

  P1IFG &= ~0x10;                        /* Elimina las interrupciones falsas */
  P1IF  = 0;
```

En el main se definen las variables y se inicializa todo lo necesario para el correcto funcionamiento del sistema:

```
uint8_t       frame[2];
uint8_t       radio_state;

BSP_Init();
SMPL_Init(0);

MRFI_SetRxAddrFilter((uint8_t *) nwk_getMyAddress());
MRFI_EnableRxAddrFilter();                  /* Filtra los paquetes para este nodo */

radio_state = MRFI_GetRadioState ();
NWK_CHECK_FOR_SETRX(radio_state);

PIN_config ();                              /* Configura los pines de I/O */
```

Y en el bucle del main se crearan 2 “rutinas” a seguir. La primera de ellas comprueba si se ha recibido un mensaje del otro nodo, en cuyo caso analizará el contenido y realizara las operaciones correspondientes para configurar la salida como es necesario:

```
    if (net_frame_get (frame) == TRUE) {

      RELE_change (frame[0]);
    }
```

Después de comprobar la recepción, comprobara si se ha producido alguna interrupción. Si se ha producido, significará que se ha activado una de las entradas, y como resultado deberá enviar el estado al otro nodo, para que actualice el estado de sus salidas:

```
if (Pushed == TRUE) {

      frame_send ();

      P1IFG &= ~0x10;                       /* Elimina las falsas interrupciones despues de enviar */
      P1IF = 0;
      Pushed = FALSE;
    }
```

Código Fuente

Código fuente de la aplicación de test:

```
/*****************************************************************************
*************/
/**
* - Activa y desactiva la rele 1 de la interfaz, indicandolo mediantes los leds del modulo
*  (verde = encendido, amarillo = apagado)
*****************************************************************************
*************/

/***                                                FICHEROS                                INCLUDE
*****************************************************************************/

#include <ioCC1110.h>

#include "bsp.h"
#include "mrfi.h"
#include "nwk_types.h"
#include "nwk_api.h"
#include "nwk_globals.h"
#include "nwk.h"
#include "common_defs.h"

/***                                                                                        DEFINES
*****************************************************************************
/

/***                                Manejo                                de                                salidas
*****************************************************************************/

#define LEDRED        P0_0
#define LEDYELLOW     P0_1
#define LEDGREEN      P0_6

#define RELE1         P0_5
#define RELE2         P0_7

#define ON            1
#define OFF           0

#define TURN_LED(led,state)   (led = state);
#define TURN_RELE(rele,state)   (rele = state);

/***                                          TIPOS                        DE                        DATOS
*****************************************************************************/

enum RELE_STATE {

   R_OFF          ,
   R_ON           ,
};

typedef enum RELE_STATE    RELE_STATE_t;
```

```
/***                                                    VARIABLES                                                    GLOBALES
*************************************************************************/

/** TRUE cuando el usuario envia una señal a P1_4 */
static volatile bool_t  Pushed = FALSE;

/** Destino del otro nodo */
#ifdef NODE_01
static addr_t    Destination = { 0x10 , 0x00, 0x00, 0x02 };
#else
static addr_t    Destination = { 0x10 , 0x00, 0x00, 0x01 };
#endif

/***                        DEFINICIONES          DE          LA          FUNCIONES          LOCALES
*****************************************************/

static void      PIN_config (void);
static bool_t    net_frame_get (uint8_t * frame);
static bool_t    frame_send (void);
static void      RELE_change (uint8_t value) ;

/*
*****************************************************************************************
***************
*    RUTINAS DE LAS INTERRUPCIONES
*****************************************************************************************
***************
*/

/**
* - Flag interrupcion externa P1
*
* - Borra los flags de las interrupciones
**/
#pragma vector=P1INT_VECTOR
__interrupt void P1_external_ISR (void) {

  if ((P1IFG & 0x10) == 0x10) {                /* Si el origen de P1_4 */

    Pushed = TRUE;
    P1IFG &= ~0x10;
  }

  P1IF = 0;
}

/*
*****************************************************************************************
***************
*    FUNCIONES GLOBALES
*****************************************************************************************
***************
*/

/**
* - Funcion main de la aplicacion
```

```
*
* - Inicia SimpliciTI API, configura los pines del modulo y el sleep timer para salir
*   del low power mode.                                          \n
**/
void main (void) {

  uint8_t       frame[2];
  uint8_t       radio_state;

  BSP_Init();
  SMPL_Init(0);

  MRFI_SetRxAddrFilter((uint8_t *) nwk_getMyAddress());
  MRFI_EnableRxAddrFilter();                  /* Filtra los paquetes para este nodo */

  radio_state = MRFI_GetRadioState ();
  NWK_CHECK_FOR_SETRX(radio_state);

  PIN_config ();                          /* Configura los pines de I/O */

  while (1) {

    if (net_frame_get (frame) == TRUE) {

      RELE_change (frame[0]);
    }

    if (Pushed == TRUE) {

      frame_send ();

      P1IFG &= ~0x10;                     /* Elimina las falsas interrupciones despues de enviar */
      P1IF = 0;
      Pushed = FALSE;
    }
  }
}

/*
*********************************************************************************************
*    FUNCIONES LOCALES
*********************************************************************************************
*/

/**
* - Configura el pinout para la aplicacion
**/
static void PIN_config (void) {

  /* Configuracion de P0 */

  P0SEL = 0x00;                           /* P0 como I/O */
  P0DIR = 0xFF;                           /* Configurados como salida */

  TURN_LED(LEDRED, OFF);
```

```
  TURN_LED(LEDGREEN, ON);
  TURN_LED(LEDYELLOW, OFF);
  TURN_RELE(RELE1, OFF);
  TURN_RELE(RELE2, OFF);

  /* Configuracion de P1 */

  P1DIR |= ~0x10;                       /* P1_4 como entrada */
  P1INP |= 0x10;                        /* Entrada triestado */
  PICTL &= ~0x02;
  P1IEN |= 0x10;                        /* Activa la mascara de interrupciones para P1_4 */
  IEN2  |= 0x10;                        /* Activa las interrupciones de la CPU para el puerto P1 */

  P1IFG &= ~0x10;                       /* Elimina las interrupciones falsas */
  P1IF  = 0;

}

/**
* - Envia el "mensaje" al otro nodo
**/
static bool_t frame_send (void) {

  ioctlRawSend_t send;
  smplStatus_t   retvalue;

  uint8_t       fr0_values[1];
  static uint8_t LastValueSent = R_OFF;

  uint8_t  radioState = MRFI_GetRadioState();
  NWK_CHECK_FOR_RESTORE_STATE (radioState);

  fr0_values[0]  = LastValueSent;

  send.addr   = &Destination;
  send.msg    = fr0_values;
  send.len    = 1;
  send.port   = SMPL_PORT_USER_BCAST;

  retvalue = SMPL_Ioctl (IOCTL_OBJ_RAW_IO, IOCTL_ACT_WRITE, &send);

  if (retvalue == SMPL_SUCCESS) {

    LastValueSent = (LastValueSent + 1) % 2;       /* 2 es el numero de los diferentes estados a enviar */
    return TRUE;
  }

  return FALSE;
}

/**
* - Cambia el estado del modulo (cambia el estado de las reles y el de los leds)
**/

static void RELE_change (uint8_t value) {

  RELE_STATE_t    action = (RELE_STATE_t) value;
```

```
  switch (action) {

  case R_ON:
    TURN_LED(LEDYELLOW, ON);
    TURN_LED(LEDGREEN, OFF);
    TURN_RELE(RELE1, ON);
    break;

  case R_OFF:
    TURN_LED(LEDGREEN, ON);
    TURN_LED(LEDYELLOW, OFF);
    TURN_LED(LEDRED, OFF);
    break;
  }
}

/**
* - Comprueba y recibe los mensajes de entrada
**/
static bool_t net_frame_get (uint8_t * frame) {

  uint8_t  radioState = MRFI_GetRadioState();
  NWK_CHECK_FOR_SETRX (radioState);

  ioctlRawReceive_t recv;

  recv.port = SMPL_PORT_USER_BCAST;
  recv.msg  = frame;

  if (SMPL_Ioctl(IOCTL_OBJ_RAW_IO, IOCTL_ACT_READ, &recv) != SMPL_SUCCESS) {

    return FALSE;
  }

  return TRUE;
}

/***                                                                                    EOF
*******************************************************************************
*****/
```

En principio no se va a realizar ningún otro test sobre la interfaz, ya que con la aplicación realizada se está probando, de forma satisfactoria, la configuración de las entradas y las salidas.

Puesto que esta interfaz es un sistema para adaptar el modulo elegido (WSN-NK01) a casi cualquier sistema empotrado para convertir actuadores he interruptores físicos en inalámbricos, estaría todo probado. A excepción de la comunicación por protocolo RS-485, que por falta de tiempo no se ha podido realizar una aplicación para su testeo.

BIBLIOGRAFIA

Para la realización del proyecto no se han consultado libros, solo se han analizado las capacidades de los componentes a través de sus datasheets, y el funcionamiento del software usado mediante las webs de sus desarrolladores.

IINFORMACIÓN

Título del proyecto

- Desarrollo de una red inalámbrica de bajo coste para pequeñas aplicaciones empotradas

Alumno

- Ferran Martí Sanvíctor

Director

- José Vicente Busquets Mataix

Universidad (campus)

- Universidad Politécnica de Valencia (UPV), campus de Vera

Escuela

- Escuela Técnica Superior de Ingeniería Informática (etsinf)

Departamento

- Departamento de Informática de Sistemas y Computadores (DISCA)

Curso

- 2009 - 2010

Titulación

- Ingeniería Técnica en Informática de Sistemas (ITIS)

Intensificación

- Administración de Sistemas y Redes
