2. Tecnologías empleadas.

2.1 Microcontroladores

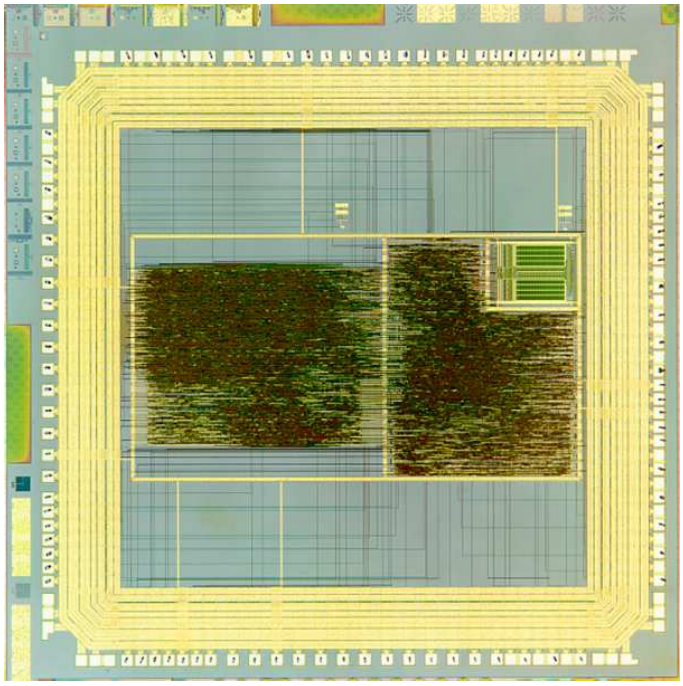


Figura 1. Fotografía del interior de microcontrolador.

Un microcontrolador es un circuito integrado que incorpora una unidad central de proceso (CPU) y una serie de recursos internos denominados periféricos. La CPU permite que el microcontrolador pueda ejecutar instrucciones almacenadas en memoria. Los recursos internos suelen ser memorias (RAM, ROM, EEPROM, FLASH), puerto serie síncrono, puerto serie asíncrono, puertos de entrada/salida, temporizadores, comparadores, capturadotes, conversores analógicos-digitales, etc.

El microcontrolador se puede considerar como una evolución del microprocesador, es una evolución porque puede llevar integradas algunas funciones que en otro tiempo requerían ser añadidas mediante la adición de circuitería externa, el claro ejemplo está en los puertos de entrada/salida y la propia memoria RAM, en los microprocesadores se debe desarrollar una lógica de control y añadir unos circuitos para implementar las funciones anteriores.

En resumen, un microcontrolador ya lleva integrada dichas funciones, facilitando el diseño y reduciendo el espacio, suponiendo una solución económica y fiable.

En la actualidad hay gran variedad de compañías a lo largo del mundo que venden microcontroladores. Empresas como Microchip, Atmel, Freescale, Zilog, Philips, OKI, Cypress, Maxim, etc, suelen ofrecer micros en diferentes formatos.

Atmel y Microchip son los dos fabricantes que acaparan casi la totalidad del mercado, seguidos por Philips y Freescale. Los fabricantes desarrollan los núcleos a los que se le dota de funcionalidades y características, fabrican sus líneas de productos pensadas para las diversas necesidades del mercado.

En este proyecto en concreto, hemos optado por micros de 8 bits de la casa Atmel[1] por las siguientes razones:

- Por ser una casa de gran éxito, pudiendo obtener sus micros a unos precios más competitivos, así como a una gran variedad de herramientas simples y de bajo costo.

- Las arquitecturas de 8 bits disponen de una gran variedad de encapsulados que las soportan. Al mismo tiempo, tienen una simplicidad que les permite llevar a cabo desarrollos de una manera sencilla, con unos tiempos y unos costes asociados bajos. En empleo de arquitecturas superiores, permitiría mejores prestaciones, sin embargo el soporte de las mismas en una placa es más costoso.

- Dentro de la oferta de 8 bits de Atmel, hay una gran variedad de encapsulados through-hole, lo que permite desarrollar las placas más simples con los medios más baratos posibles.

Atmel dispone de su núcleo de 8 bits, AVRcore. Se trata de un núcleo que suele usar en una línea de micros de 8 bits. Paralelamente también trabaja con el popular núcleo del 8051, que es muy usado en multitud de aplicaciones industriales. Lo que diferencia a un micro de otro, dentro de un mismo núcleo, son los periféricos que usa y la velocidad de reloj máxima que puede alcanzar.

La tendencia actual es la siguiente, a pesar del incremento de la memoria integrada en el propio microcontrolador, durante los últimos años ha habido un incremento en el área de los microcontroladores en los siguientes aspectos:

- Velocidad de funcionamiento, existiendo una gran cantidad de dispositivos que operan a 4 MHz, podemos encontrar micros que alcanzan los 16 o 20 MHz. Cabe decir que este parámetro, es un tanto relativo, pues por ejemplo, el Atmega128 de Atmel funciona con una frecuencia de 16 MHz, aunque usando un ciclo de reloj para la gran mayoría de instrucciones que posee. Por otro lado, el modelo MC9S12D de la familia HC12, puede operar a 25 MHz, requiriendo múltiples ciclos de reloj por instrucción.

[1] A partir de esta parte del documento, vamos a dar por supuesto que estamos hablando de micros de Atmel.

- La segunda tendencia se trata de la especialización de los periféricos internos de los microcontroladores. Disponer de una UART, así como de una interfaz SPI, es algo que se ha convertido en común en los microcontroladores de alta gama de los últimos años. En la actualidad se están añadiendo más interfaces con distintas especializaciones. Por ejemplo, la interfaz USB, muy popular en el mercado de los ordenadores, y está introduciéndose rápidamente en el mercado de los micros.

La casa ARM no fabrica expresamente microcontroladores, sin embargo ha proporcionado las arquitecturas de 32 bits que otros fabricantes ofrecen. Estos dispositivos se están implantando en multitud de sistemas tales como móviles, microordenadores, electrodomésticos, etc. Su altas prestaciones, junto con las cualidades asociadas a los microcontroladores de alta integración de subsistemas, le convierte a esta arquitectura como el futuro de los semiconductores programables.

2.1.1 Elementos de un microcontrolador.

Un microcontrolador dispone de:

- CPU: se trata de un circuito digital síncrono, cuya función consiste en coger las instrucciones y los datos guardados en memoria, procesarlos y efectuar una serie de acciones. Le velocidad de procesamiento de los datos y las instrucciones, depende de cómo esté diseñada la CPU, así como de qué frecuencia de reloj máxima puede alcanzar.

- Memoria: es donde se guardan las instrucciones y los datos, actualmente los fabricantes integran diversos tipos de memoria en sus diseños. Se suele combinar las memorias volátiles con las no volátiles. Gracias a la tecnología FLASH, es posible introducir memorias grandes integradas en el en propio encapsulado. Este hecho, permite que se reduzca la necesidad de tener que usar una memoria externa, con el consiguiente ahorro en espacio, tiempo de diseño, componentes externos, así como en el empleo de pins de los puertos de E/S. Otra mejora gran ventaja del empleo de memorias internas, es que los tiempos de acceso a éstas son menores, con lo que la ejecución del programa es más rápida.

- Periféricos: son el conjunto de circuitos integrados asociados a un microcontrolador, cada periférico dispone de una serie de registros que permiten controlar, configurar y monitorizar el estado de dichos circuitos. Describir los periféricos que se pueden integrar en un micro, es una tarea ardua, pues hay una gran cantidad y diversidad de los mismos. Éstos pueden ser usados de dos maneras distintas:

    - Polling. Desde la aplicación se controla al periférico, estudiándose y controlándose su evolución, actuándose cuando el programador de la aplicación lo requiera, suele emplearse los flags para actuar.

- Interrupciones. El periférico es capaz de cortar el desarrollo normal del programa para ser atendido, ejecutando un conjunto de acciones programadas frente a ese evento.

Seleccionar de qué manera actuar, depende del programador, pues cada forma de afrontar el diseño de la aplicación tiene sus ventajas e inconvenientes. Por ejemplo, Polling es simple de usar, pero requiere que el microcontrolador desperdicie tiempos que podría usar en otras tareas. Por el contrario, las interrupciones permiten atender al periférico en concreto que solicita la atención de la CPU, sin embargo un empleo descontrolado de interrupciones, podría introducir el microcontrolador en un estado de indeseado que perdiera el puntero de programa, de ese estado únicamente se saldría con un reset la aplicación. Es por tanto importante llevar un control adecuado de las interrupciones.

2.1.2 Programación de una aplicación para micros de Atmel.

Las instrucciones asociadas a un microcontrolador son únicas para el mismo. Aunque los núcleos sean los mismos, la propia configuración de un micro puede marcar que un pedazo de código no pueda ser utilizado en otro micro de la misma casa, ello es debido a la falta de convergencia en sus periféricos.

Emplear el ensamblador del micro para desarrollar el programa, sigue siendo usual en aquellas aplicaciones que requieren un control exhaustivo de los procesos que intervienen en un sistema. Por el contrario, existe una serie de soluciones en el mercado que vienen a paliar la necesidad de conocer a fondo el código ensamblador de un micro. Estamos hablando de los denominados compiladores cruzados, que tratan de poder programar el micro con lenguaje de alto nivel (C, Basic, Pascal, etc), mejorando los tiempos de desarrollo, aunque perdiendo un poco de control de los recursos internos del micro.

Existe una gran comunidad que defiende el empleo de los lenguajes ensambladores por los motivos expuestos anteriormente, aunque a medida que la aplicación a implementar es más ambiciosa, el desarrollo del programa en un lenguaje de alto nivel es más fácil de gestionar, ampliar y mantener.

De los diversos lenguajes de programación que hay para los “Sistemas Empotrados” (traducción literal de “Embedded Systems), el lenguaje C es el más empleado por los desarrolladores. Si nos centramos en los micros de Atmel, existen una gran variedad de herramientas que nos permiten desarrollar nuestra aplicación en C. La más extendida es WinAVR, que es gratis y que dispone de una gran comunidad que la mantiene. Incorpora además una nueva ventaja, hoy en día está integrada en el entorno AVR Studio de Atmel, por lo que la interacción de con el depurador es inmediata.

Los entornos de desarrollo de pago, son programas dirigidos a profesionales, están muy bien conseguidos y llevan añadidas multitud de funcionalidades extra que ayudan en el desarrollo de software. Algunos disponen de un generador de código automático, que

facilita las primeras etapas de diseño de una aplicación, otros incluso permiten depurar aplicaciones en tiempo real con algunas herramientas hardware. Así pues, aunque WinAVR es una solución muy honrosa, no es descabellado elegir los entornos profesionales para llevar a cabo una labor de diseño, ya que ese conjunto extra funcionalidades permite un desarrollo más cómodo para el programador.

Nosotros hemos optado por una solución intermedia, buen precio y buenas prestaciones, nos referimos al Codevision, este software dispone de todo lo necesario de un compilador, nos permite poder añadir los más nuevos programadores al entorno, refiriéndonos como tales a los dispositivos hardware que permiten grabar los programas en la/s memoria/s. Aunque como inconveniente resaltamos que para la depuración se apoya en el AVR Studio de Atmel.

2.1.3 Programadores de Atmel.

Los programadores son los dispositivos que permiten grabar los programas desarrollados por los usuarios en los microcontroladores. Existen una gran variedad de este tipo de herramientas, algunos son comercializados por Atmel o bien por otros fabricantes tales como Kanda Systems, Priio, Propox, etc.

Hay cuatro formas de programar un micro de Atmel, dependiendo de la interfaz elegida.

- Interfaz paralelo.
- Interfaz serie.
- Interfaz Jtag.
- Bootloader

Todas estas interfaces permiten programar el micro en la propia placa, con las ventajas que ello conlleva. El interfaz Jtag es un estándar de programación de micros y FPGAs. No todos los micros de Atmel están acogidos a dicho interfaz. Tiene la ventaja de poder estudiar el estado del micro en tiempo de ejecución.

Todos los micros pueden ser programados de forma paralela o de forma serie, sin embargo en la realidad, la gran mayoría de los productos de que hay el mercado se basan en la interfaz serie, que conlleva una ahorro en el número de pins asociados, aunque es más lenta.

“Bootloader” es el nombre de un programa que se guarda en la memoria de micro, permite programar otra parte de la misma sin necesidad de un dispositivo específico para la programación. Es un método eficiente, únicamente se requiere grabar de manera residente una aplicación, ésta arrancaría al encender la placa que lleva el micro, y espera que se le vayan enviando los bytes que conforman la aplicación a grabar, los datos se envían por el puerto serie y la aplicación está pensada para irlos guardando en una zona de la memoria interna. Para llevar a cabo tal técnica de programación, sólo se requiere una aplicación, un zona de memoria libre y una software que envíe el programa byte a byte por el puerto serie, no se requiere ningún programador.

En los datasheets se muestran las especificaciones de este tipo de interfaces, así mismo Atmel dispone de notas de aplicación de diseño de "bootloaders".

Los programadores serie pueden emplear el puerto serie, el puerto paralelo y últimamente también el interfaz USB. Mediante esos interfaces el programa que controla el programador envía los bytes que van a ser grabados en la memoria.

Los entornos desarrollo permiten poder seleccionar el programador.

De entre todas las herramientas que hay en el mercado, cabe destacar el programador ISP paralelo, se trata de un programador simple, reconocido por todos los entornos de desarrollo, así como una gran mayoría de software de programación de semiconductores, cuyo esquemático viene publicado en Internet. Su coste es de 12 € aproximadamente, su esquemático queda como sigue:

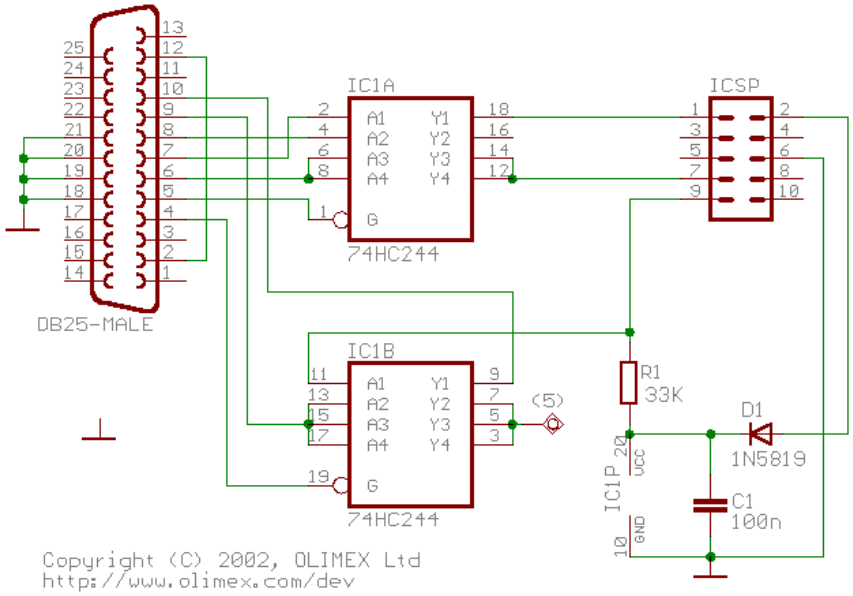


Figura 2. Esquemático del programador AVR ISP desde el puerto paralelo.

Se observa en la figura 2 que el programador sólo requiere un integrado 74HC244, 8 buffers con 3 estados. El programador es controlado por un programa que usa el puerto paralelo, mandando los comandos necesarios al micro que está conectado al puerto ISP.

Existe otra variante del mismo circuito que es aún más barato, se trata del programador que lleva integrada la popular placa STK500, a diferencia del anterior ésta usa el puerto serie, su esquemático sería:

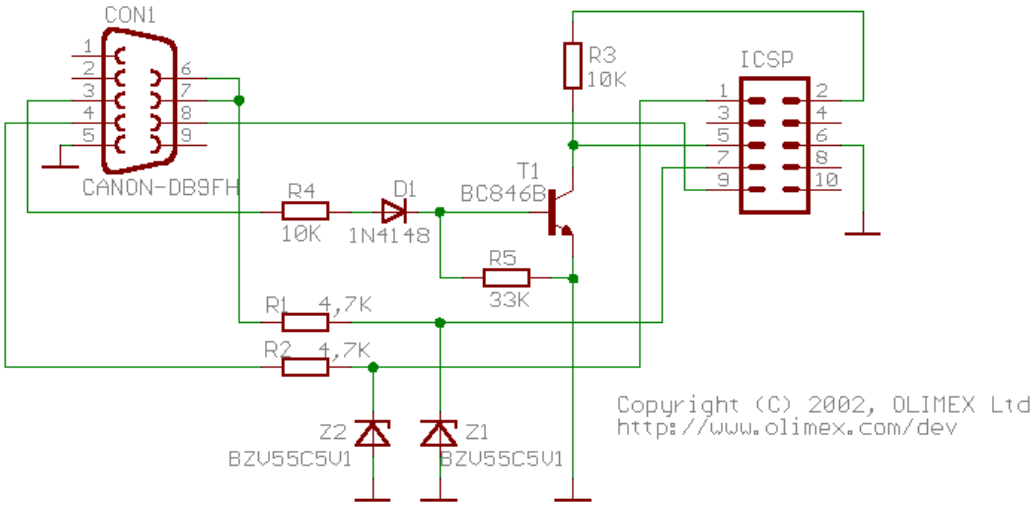


Figura 3. Programador por interfaz serie.

La placa STK500 es muy popular en el mundo de Atmel, puesto es una solución muy didáctica, siendo una excelente herramienta para introducirse en el mundo de los microcontroladores. A diferencia del STK200, el STK500 viene incluida entre las herramientas de programación del entorno AVR Studio, de todas maneras ambas gozan de gran difusión en un gran número de desarrolladores.

Gracias al éxito que está disfrutando el puerto USB en los últimos años, están apareciendo unos programadores específicos para tal puerto. El concepto es el mismo, control del puerto ISP pero desde otro interfaz diferente. Esta herramienta es ideal para la nueva generación de ordenadores portátiles que van eliminando los puertos series de sus especificaciones. Un ejemplo de este producto sería:

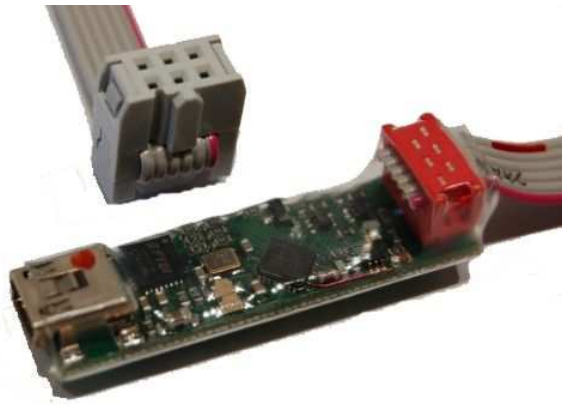


Figura 4. CrispAVR-USB, cortesía de http://www.chip45.com

La idea es la misma que para el programador serie, aunque se incluye un integrado llamado FT232. Se trata de un semiconductor que implementa un esclavo USB y que se comunica con un maestro (USB desde el PC). Asociado a dicho integrado existen unos drivers que

permiten usar el puerto USB como si fuera un puerto serie ordinario, asignando un puerto serie nuevo a dicho integrado y sin necesidad de cambiar el software de programación, lo único que se requiere es definir el nuevo puerto COM a usar.

Para finalizar comentamos el nuevo lanzamiento de Atmel, se trata del AVR Dragon. Un programador-emulador que permite programar todos los micros de Atmel desde todos los interfaces posibles. Se trata de una herramienta de bajo costo (63 €) que viene a sustituir el STK500.

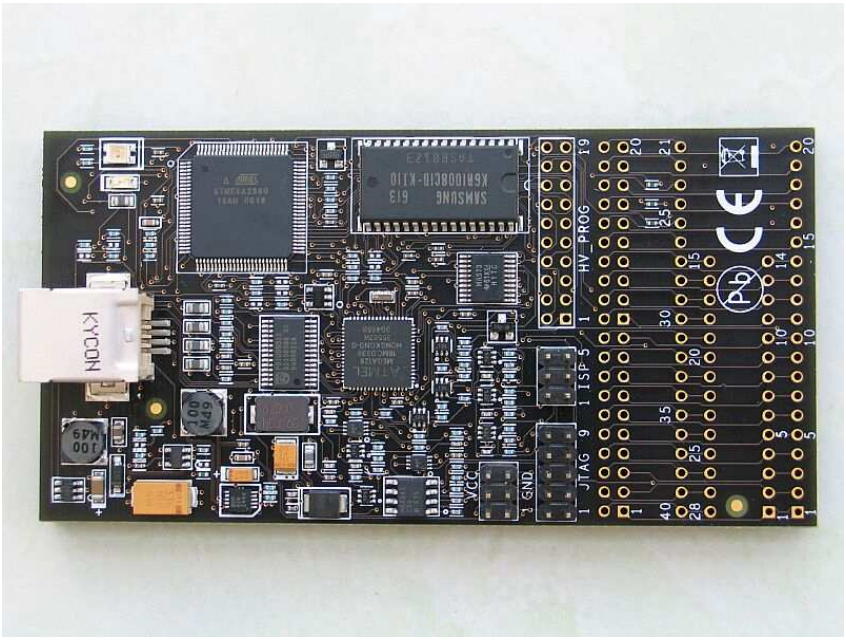


Figura 5. AVR Dragon de Atmel.

2.1.4 Codevision.

El Codevision presenta un entorno de desarrollo desde el que se pueden programar las aplicaciones para microcontroladores de Atmel. Codevision es un producto de HP Info Tech. Su interfaz sencillo presenta una atractiva herramienta para aquellos desarrolladores que usan el C para sus diseños. Empresas como Boeing, HP, Siemens, Braum, e inclusive la agencia espacial NASA, usan este software.

Si se desea ver las características de este entorno, entre en el siguiente link:

http://www.hpinfotech.ro/html/cvavr_features.htm

Básicamente dispone de un editor con sus extras. Se puede modificar las opciones de proyecto, las herramientas de programación, la configuración de la edición, etc.

Dispone además de una opción de generación de código automática, de esta manera se pueden configurar de forma rápida los periféricos a usar, para el conjunto de micros sobre los que puede actuar.

Entre las opciones que Codevision ofrece está:

- Edición de ficheros.
- Compilación.
- Programación.
- Debugging (con ayuda del AVR Studio).
- Generación de código automática.
- Lectura de dispositivos AVR.

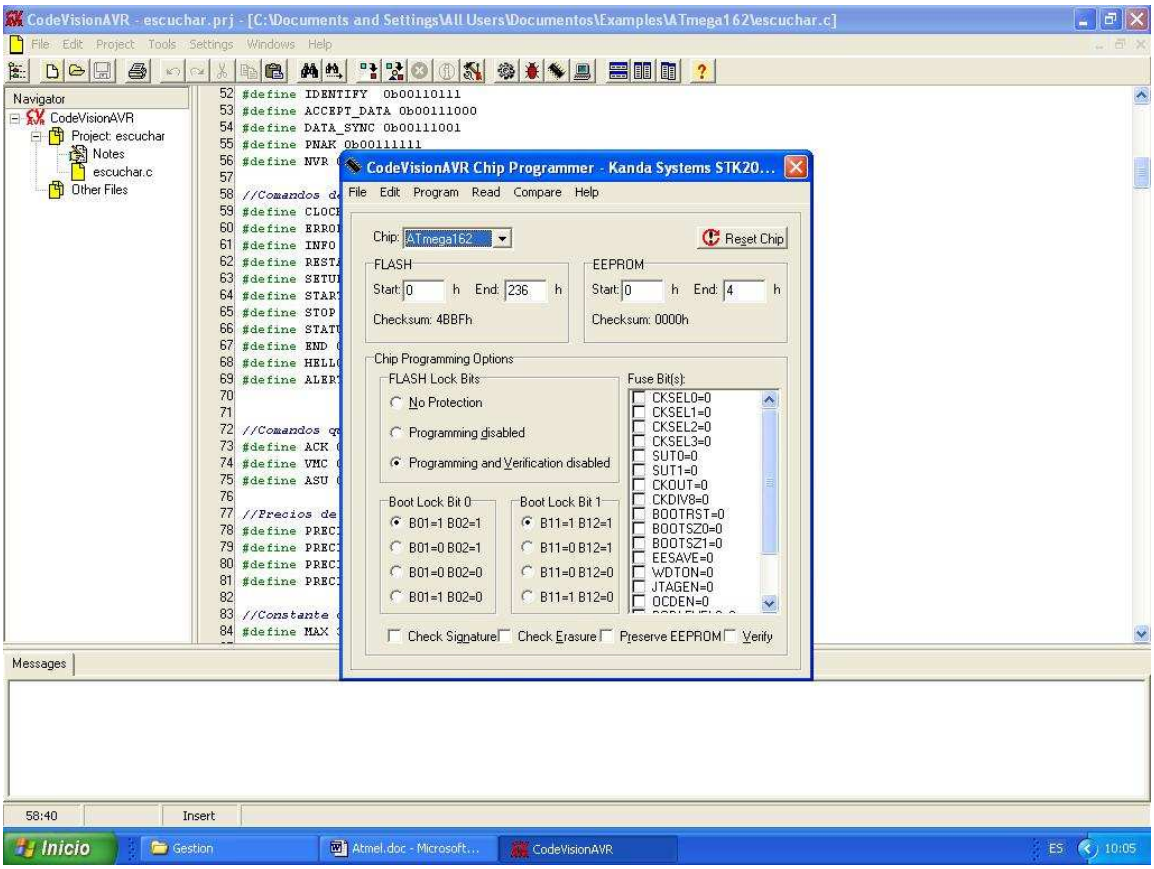


Figura 6. Screeshot del Codevision.

2.1.5 Atmega8 y Attiny2313.

Para llevar a cabo este proyecto, se han desarrollado dos placas, una de ellas lleva implementa un ECC (maestro) y otra un EC (esclavo). Sin entrar en detalle en cada diseño, pues ese no es el alcance de este apartado, comentamos que para el EC usamos un microcontrolador Atmega8 de Atmel, mientras que para el ECC usamos un Attiny2313. Cabe destacar que el ECC es el elemento central del sistema, y usa un micro de menor envergadura que el EC, esto no supone ninguna contradicción, pues la complejidad de cada sistema va referida al conjunto.

La serie Atmega viene a ser una línea de gama media-alta, son dispositivos con unas buenas especificaciones de memoria así como de periféricos, Microchip dispone de la serie 16-18 como homóloga de la Atmega.

La serie Attiny es una gama baja, pensada para aquellas soluciones que requieren unas especificaciones pobres, aunque dotando de un mínimo de funciones extras. La serie Attiny es la equivalente a 12 de Microchip, las especificaciones de memoria son bajas, no superan los 4Kbytes, además llevan incluidos pocos periféricos.

2.1.5.1 Atmega8.

Las características de este micro son:

• Alto rendimiento y bajo consumo.

• Arquitectura avanzada RISC.

- 130 potentes instrucciones – La mayoría ejecutables en un ciclo de reloj.
- 32 x 8 Registros de propósito general.
- Hasta 16 MIPS a 16 MHz.
- Multiplicador de 2-ciclos en chip.

• Memoria para datos y programas no volátiles.

- 8K Bytes de flash.
- Duración: 10,000 ciclos de borrado/escritura.
- Sección de arranque independiente con Bits de bloqueo.
- Verdadera operación de lectura mientras se escribe.
- 512 Bytes EEPROM.
- Duración: 100,000 ciclos Escritura/borrado
- 1K Bytes SRAM interna.
- Hasta 64K Bytes direccionables para memoria externa.
- Opción de bloqueo para seguridad del software.

• Características de periféricos.

- Dos 8-bit Timers/Counters con pre-scalers separados y modos de comparación.
- Un 16-bit Timers/Counters conpre-scalers separado y modos de comparación.
- Contador en tiempo real con un oscilador separado.
- 8 canales analógicos con 10 bits de precisión
- 1 USARTs programable y dual.
- Master/Slave SPI (Interfaz serie síncrona).
- Temporizador programable Watchdog con un oscilador separado en el chip.
- Comparador analógico interno.
- 6 canales ADC en el encapsulado PDIP.
- 3 canales PWM.

• Características especiales de microcontrolador.

- Power-on Reset y Programable Brown-out Detección.
- Oscilador interno RC calibrado.
- Fuentes externas e internas de interrupciones.
- Cinco modos de "sueño": Idle, Power-save, Power-down, Stand-by y Extended Standby.

• Entradas y salidas, encapsulados.

- 23 Líneas de entrada/salida de propósito general..
- 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF

• Voltajes de operación.

- 2.7 - 5.5V (ATmega8L)
- 4.5 - 5.5V (ATmega8)

• Velocidades.

- 0 - 8 MHz (ATmega8L)
- 0 - 16 MHz (ATmega8)

• Consumo 4 Mhz, 3V, 25°C

- Activo: 3.6 mA

- Modo de reposo: 1.0 mA
- Modo de ahorro: 0.5 µA

El encapsulado que se va a usar es el PDIP, que es más fácil de manejar.

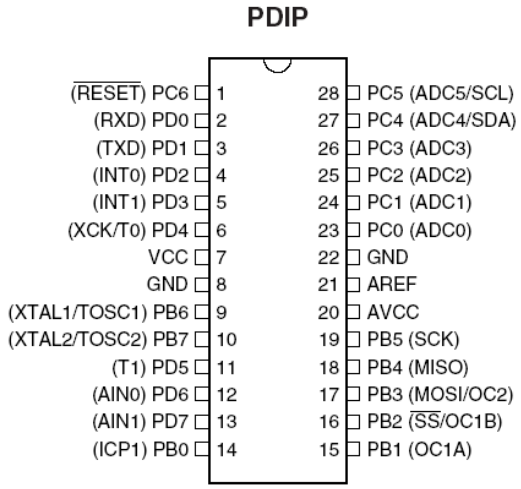


Figura 7. El encapsulado PDIP 28 es ideal para el prototipado.

2.1.5.1 ATtiny2313.

- 120 potentes instrucciones – La mayoría ejecutables en un ciclo de reloj.
- 32 x 8 Registros de propósito general.
- Hasta 20 MIPS a 20 MHz.
- Alto rendimiento y bajo consumo(usa la arquitectura AVR RISC).
- Multiplicador de 2-ciclos en chip.

• Memoria para datos y programas no volátiles.

- 2K Bytes de ISP.
- Duración: 10,000 ciclos de borrado/escritura.
- Sección de arranque independiente con Lock Bits.
- Verdadera operación de lectura mientras se escribe

- 128 Bytes EEPROM
- 128 Bytes SRAM interna

- Duración: 100,000 ciclos Escritura/borrado
- 1K Bytes SRAM interna.
- Hasta 64K Bytes de espacio de memoria externa.

• Características de periféricos.

- Un 8-bit Timers/Counters con separados pre-scalers y modos de comparación.
- Un 16-bit Timers/Counters con separados pre-scalers y modos de comparación.
- Contador en tiempo real con oscilador separado.
- 1 USARTs programable y dual.
- Programable temporizador Watchdog con un oscilador separado en el chip.
- Comparador analógico en chip.
- 4 canales PWM.
- USI – Universal Serial Interface

• Características especiales de microcontrolador.

- Power-on Reset y Programable Brown-out Detección.
- Oscilador interno RC calibrado.
- Fuentes externas e internas de interrupciones.
- Cinco modos de "sueño": Idle, Power-save, Power-down, Stand-by y Extended Standby.
- Programable via SPI.
- Cable de "debug" para chequeo desde la placa.
- Función de reseteo mejorada tras arrancar el micro.

• Entradas y salidas, encapsulados.

- 18 Líneas de entrada/salida de propósito general..
- 20-pin PDIP, 20-pin SOIC, 20-pad QFN/MLF.

• Voltajes de operación.

- 1.8 - 5.5V (Attiny2313V)
- 2.7 - 5.5V (Attiny23213)

• Velocidades.

- ATtiny2313V: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V
- ATtiny2313: 0 - 10 MHz @ 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V

• Consumo 4 Mhz, 3V, 25°C

- Activo:
  - 230 uA a 1MHz y 1.8V
  - 20 uA a 32 KHz y 1.8V
- Modo de reposo: 1.0 mA
  - <0.1 uA a 1.8V

En encapsulado PDIP20:

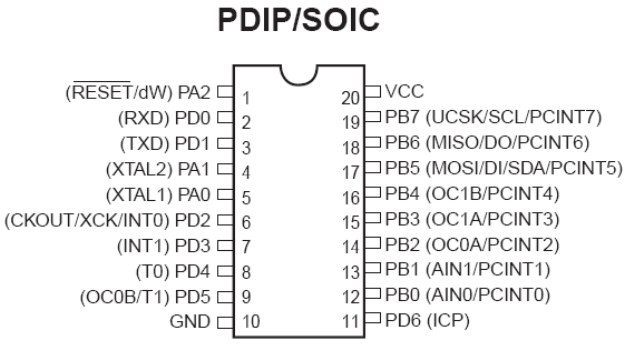


Figura 8. Patillaje para el Attiny2313 en PDIP20.

2.2 Elementos RF.

Tal y como se dijo en la introducción, existen una gran variedad de soluciones RF que se pueden adaptar, tantas como criterios de elección. Es decir, se puede elegir atendiendo a los siguientes aspectos:

- Frecuencia. El espectro de frecuencias es amplio y verdaderamente puede haber una solución electrónica para una determinada frecuencia. Se puede optar por una frecuencia que esté libre, es decir, exenta de las restricciones impuestas por el Cuadro Nacional de Frecuencias, cuya longitud de onda tenga una ventaja adicional, recordamos que las bajas frecuencias son más inmunes a los fenómenos medioambientales y a los obstáculos que las altas, sin embargo requieren antenas del orden de sus longitudes de onda ($\lambda/4$). En el mercado, se nos ofrecen un conjunto de bandas en el que hay gran variedad de sistemas de radiofrecuencia, nos podemos centrar en las siguientes:

  - 433 MHz
  - 868 Mhz
  - 2,4 GHz

  Se encuentran en el ámbito de las microondas, por lo que las antenas que requieren son pequeñas, pudiendo concordar con la exigencia de productos de tamaño reducido. En el caso de 433 MHz se requeriría una antena de 17 cm.

  En la actualidad hay una gran variedad de productos centrados en dichas bandas

- Estándares. Los estándares de radiofrecuencia permiten ahorrar tiempo de desarrollo, al mismo tiempo que dotan al sistema de fiabilidad y universalidad. Acogerse a un estándar puede suponer ser una buena solución, puesto que la etapa de desarrollo del protocolo de conexión queda superada de manera simple. Sin embargo, los productos acogidos a un estándar son más caros y llevarlos a una placa requiere una serie de infraestructuras adicionales tales, que sólo una empresa con grandes medios puede tenerlos.

  Estos estándares pueden ser:

  - Bluetooth.
  - Wifi.
  - Zigbee.
  - HomeRF.
  - Z-Wave.

- Ancho de banda. En una red domótica se puede transmitir datos de estado, de control, comandos de actuación y lectura, etc. Sin embargo podría transmitirse vídeo y/o audio. Este tipo de tráfico como es sabido tiene una necesidades de ancho

de banda grande, por lo que podríamos elegir como criterio de diseño dispositivos que cumplan tales requerimientos. En el siguiente cuadro tenemos la comparativa de anchos de bandas:

Protocolo	Régimen binario
Wi-Fi	Hasta 54 Mbps
Bluetooth	1 Mbps
ZigBee	250 Kbps
HomeRF	1,6 Mbps

- Coste. Este punto habla por sí mismo, basta con dar un "paseo" por la Red (Internet) y consultar precios.

Atendiendo a los criterios expuestos anteriormente, concluimos:

- Buscamos un dispositivo que opere en las bandas ISM.

- No nos acogemos a ningún estándar por motivos de coste, puesto que un dispositivo de los mencionados anteriormente suele rondar en media un precio de 40 €, a la vez que tienen formato SMD, o bien, si operan en el rango de las microondas se requiere contemplar la Teoría de Líneas de Transmisión en el diseño. Todos esos aspectos encarecen el desarrollo, y requieren además el empleo de instrumentación costosa (Analizadores de Red y Espectro).

- Las prestaciones de ancho de banda son bajas, para usar anchos de bandas elevados debemos acogernos a un estándar (desechado previamente por costoso) ó diseñar un transceptor a medida, cosa que sale del alcance de este proyecto.

- Coste, existen una gran variedad de dispositivos que usan modulaciones simples tales como AM, OOK, FM, FSK, que nos permiten establecer conexiones RF con unos anchos de banda bajos.

Se elegirá una solución de las llamadas "Cajón de Sastre", por no estar acogidas a ningún estándar, usando un tipo de modulación determinado y necesitado el desarrollo de un software de control para poder ser empleada. Otro factor importante es que sean fácilmente integrables en una placa. Por ambas razones es justificable esta solución, teniendo como contra partida tener que desarrollar un protocolo de comunicación que garantice las conexiones y el trasvase de información.

Existen una gran variedad de semiconductores acogidos a esta definición, Nordik, Atmel, Chipcom, Jennic, Integration, etc. Por otro lado, también los denominados módulos híbridos, que son una solución final, usándose el dispositivo directamente, sin apenas empleo de elementos externos de polarización, a esta familia pertenecen los módulos de

Aurel, Linx, MaxStream, etc. En concreto hemos elegido a HopeRF, empresa ubicada en la próspera ciudad China de Shenzhen.

HopeRF es una empresa de microelectrónica, desarrollan sus propios semiconductores, aunque sin encapsular. Los dispositivos resultantes son sus transceptores, sensores e incluso transistores para radiofrecuencia, véase http://www.hoperf.com

Sus transceptores son usados en multitud aplicaciones, tales como juguetería, estaciones meteorológicas e industria,.

La elección de estos dispositivos se debe fundamentalmente al coste. Cada chip transceptor cuesta 1,5 €, siendo además muy fácil de programar e integrar. Su comportamiento es similar a los chips de Integration, véase http://www.integration.com

2.2.1 RF12.

El RF12 de HopeRF es un chip único, de bajo consumo, con un transceptor FSK multicanal, para uso en aplicaciones que requieren cumplir con los estándares de la ETSI y de la FCC. Desarrollados para ser usados en las frecuencias de 433, 868 y 915 MHz, correspondientes a las bandas ISM, es un chip de radiofrecuencia analógico. Tiene un sistema de radio que incluye un PLL multibanda con un amplificador de potencia, LNA, mezcladores con down-converters de I/Q, filtros en la banda base y demodulador I/Q. Todas la funciones de radiofrecuencia están integradas. Sólo un cristal externo y un filtrado de bypass con condensadores pueden ser necesarios.

El RF12 lleva un PLL completamente integrado, con una respuesta rápida ante la desviación en frecuencia, pudiendo ajustarse además a la portadora rápidamente frente a los desvanecimientos, y minimizando el efecto de las interferencias multicamino, con lo que se dota al sistema de un robusto enlace de radio.

Sus características son:

- Alta integración, requiriendo poco elementos externos
- Rápida configuración, programable y con un PLL de alta resolución.
- Capacidad de recuperación rápida.
- Alto régimen binario, hasta 115.2 Kbps .
- Entrada/Salida de antena diferencial.
- Amplificador de potencia integrado
- Desviación de frecuencia programable en TX (15 to 240 kHz)
- Ancho de banda programable en banda base RX (67 to 400 kHz)
- Valores analógicos y digitales de RSSI.
- Control automático de frecuencia.
- Detector de la calidad de datos recibidos
- Recuperación de reloj por filtrado de datos internos.
- Sincronización por reconocimiento de patrones en recepción.
- Interfaz SPI.

- Señales de CLK y RESET
- Registro de 16 bits en la cola (FIFO).
- 2 registros de transmisión de 8 bits.
- Ciclo de bajo consumo.
- Referencia de cristal externo de 10 MHz
- Temporizador despertador.
- Tensión de alimentación de 2.2 a 3.8 V.
- De 2.2 a 5.4 V de alimentación.
- Bajo consumo
- Consumo en stand by (0.3 µA)
- Soporta paquetes cortos menos de 3 bytes).
- Alta estabilidad frente a la temperatura.
- Encapsulado TSSOP de 16 pins.

2.2.1.2 Aspectos técnicos.

De manera similar a un microcontrolador, el RF12 tiene dos modos de funcionamiento, que son el modo normal y el modo de configuración.

Mediante la configuración se establecen los parámetros que el sistema de radio va a usar, entre ellos tenemos la selección de la banda de frecuencias, la capacidad que se le va cargar a la antena, el tipo de comunicación a usar, la desviación de frecuencia máxima, la sensibilidad, el ancho de banda, etc.

El modo normal es el que el sistema radio usa para intercambiar información. Para transmitir y recibir datos el sistema hace uso de una serie de comandos, de manera similar al modo de configuración, pues transmitir un byte consiste simplemente en mandar un comando al RF12, y de la misma forma recibir es enviar un comando y esperar una respuesta por parte del transceptor.

La comunicación con el microcontrolador se hace por medio de un interfaz SPI (shyncronous serial perypheral). Dicho interfaz es un estándar muy usado en los sistemas empotrados, se caracteriza por usar una señal de reloj para sincronización.

Cuando el RF12 recibe un comando debe activarse una señal de CHSEL, al estilo de la memorias, conversores A/D, LATCHES, etc. Hablamos de la señal nSEL. Cada vez que se quiera interactuar con el RF12 debe usarse tal señal.

2.2.1.2.1 Descripción.

El RF12 es un Transceptor FSK destinado a trabajar en las frecuencias 315, 433, 868 y 915 Mhz. El receptor usa la aproximación ZERO-IF con demodulación I/Q (fase y cuadratura), usando un mínimo número de componentes externos. El RF12 incorpora los elementos que vienen a continuación, y que pueden verse en la figura 9:

- PLL: este dispositivo determina la frecuencia de operación, preservándose la precisión basada por VCO (integrado en el chip) que da la referencia de frecuencia. Los PLLs de alta resolución permiten el uso de varios canales dentro de la misma banda. El uso del VCO dentro del PLL es para dar la señal de calibración necesaria, para lo cual se emplean unos pocos microsegundos. La calibración empieza cuando el sintetizador arranca, si la temperatura o la tensión de alimentación cambian, el VCO se recalibra de manera sistemática.

- RF Power amplifier (PA): tiene una salida en colector abierto diferencial, que alimenta el bucle de una antena con un valor de potencia de salida programable. Se incluye un sintonizador de antena para evitar en conocido "hand effect".

- LNA: la entrada de este amplificador tiene una impedancia de 250 Ω, si se le incluyese a la entrada una antena de 50 Ω, se requiere un circuito de adaptación de impedancias para minimizar la figura de ruido en el receptor.

  La amplificación del LNA puede ser seleccionada (0, -6, -14, -20 dB relativos al valor más alto), de forma proporcional a nivel de señal recibido. Puede ser útil en entornos ruidosos.

- Filtros en la Banda Base: el BW del receptor es programable seleccionando el BW del filtro de la Banda Base. Esto permite ajustar el receptor de acuerdo a la señal recibida.

  Un ancho de banda puede ser acomodado de acuerdo a la desviación de la señal FSK. La estructura del filtro es de un Butterworth paso de baja de 7º orden, con 40 dB a 2*BW.

- Filtrado de datos y recuperación de reloj: el filtrado de los datos de salida puede ser completado por un condensador externo o bien usando un filtrado digital de manera acorde a la aplicación final.

  - Filtrado analógico: se trata de un filtro paso de baja RC, seguido de un Smitch-Trigger. Una capacidad externa puede ser incluida, acorde al régimen binario real. La FIFO no puede ser usada con este filtro, además el reloj no es proporcionado para los datos modulados.

  - Filtrado digital: se usa un filtrado digital a una frecuencia 29 veces superior a la tasa de bits. En este modo, hay un circuito detector de reloj, usando este

reloj puede ir llenándose la FIFO. El CR (circuito recuperador) tiene tres modos de uso: rápido, lento y automático. En modo lento, la inmunidad al ruido es mayor, pero tiene un mayor tiempo de operación y requiere mayor precisión que en modo rápido. En automático, el CR cambia de lento a rápido, el CR empieza en modo rápido, y tras determinar el reloj pasa a modo lento.

- Bloques validadores de datos:

  - RSSI (Lectura de nivel de señal): una salida digital RSSI es proporcionada para el nivel de la señal de entrada. Pone una señal a nivel alto si el nivel de señal entrante detectado supera el programado. Un RSSI analógico también está disponible. La respuesta del RSSI depende de la capacidad externa del filtro. El pin 19 es la salida analógica del RSSI.

  - DQD: (Detector de calidad de datos) basado en la contabilización de la saltos abruptos en los datos recibidos y no filtrados. El umbral DQD es completado usando el comando de filtrado de datos.

  - AFC: (Control automático de frecuencia) el receptor puede minimizar el offset de TX/RX en paso discretos, usando:

    1. Cristales de baja precisión.
    2. Ancho de banda estrecha en el receptor.
    3. Elevada Tasa de Bits.

- Cristal oscilador. El RF12 posee un único pin con una salida, correspondiente con la salida de un circuito generador de reloj, que dota además de una señal de referencia 10 Mhz para el PLL. Puede dotar de señal de reloj para un microcontrolador, economizando los diseños.

- Detector de batería baja. El circuito detector del nivel de batería genera una petición de interrupción cuando el nivel de tensión baja del umbral programado.

- Despertador. Es un dispositivo de bajo consumo que puede temporizar desde 1ms a varios días con una precisión de un 5%.

- Manejador de eventos. Con motivo de economizar el consumo, el transceptor soporta varios modos de ahorro de energía. Se puede pasar al modo activo, por medio de eventos activadores (pulso negativo nINT, activación del despertador, FIFO llena ó por petición expresa a partir del interfaz serie).

  Si un evento de asociado a la temporización del reloj ocurre, la lógica del despertador genera una señal de petición de interrupción que podría ser usada para

despertar también al microcontrolador, reduciendo el tiempo que éste podría estar activo. La fuente de la interrupción puede ser leída por el microcontrolador a partir del pin SDO.

- Interfaz con el microcontrolador. Gracias a esta interfaz el micro puede seleccionar la banda de frecuencia, el centro del sintetizador, el ancho de banda en recepción, etc. Todos los parámetros son configurables tras encenderse el sistema, los valores programados son guardados en modo sleep. Se permite la lectura del estado de los registros así como de los datos entrantes. El bloque transmisor dispone de un registro de transmisión de 8 bits para ser emitidos. Dichos datos pueden ser transmitidos a un régimen binario determinado.

  También es posible guardar los datos en la FIFO y leerlos cuando esté llena.

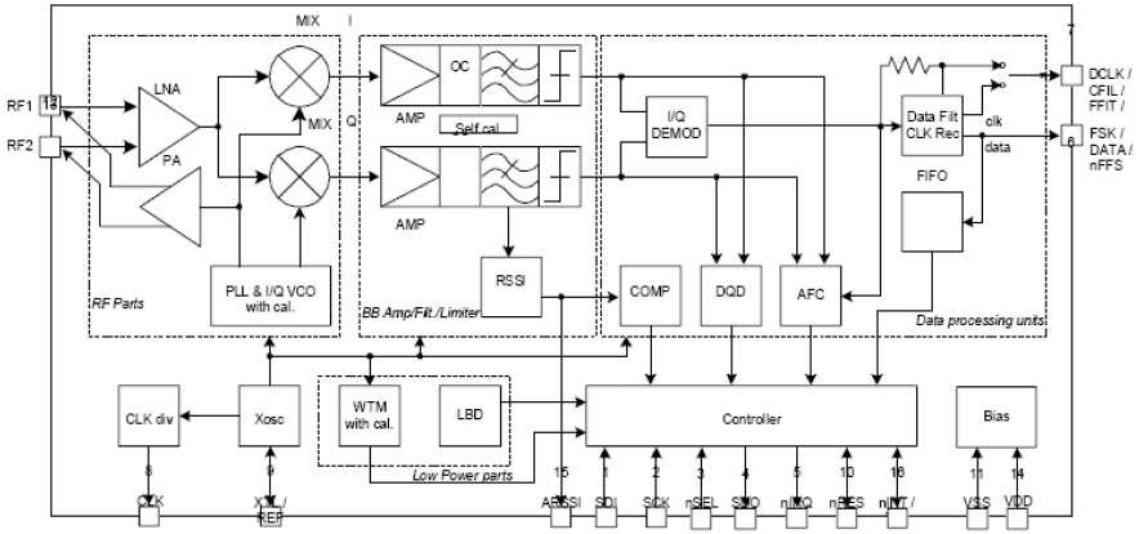


Figura 9. Diagrama de bloques del RF12.

2.2.1.2.2 Comunicación SPI con el RF12.

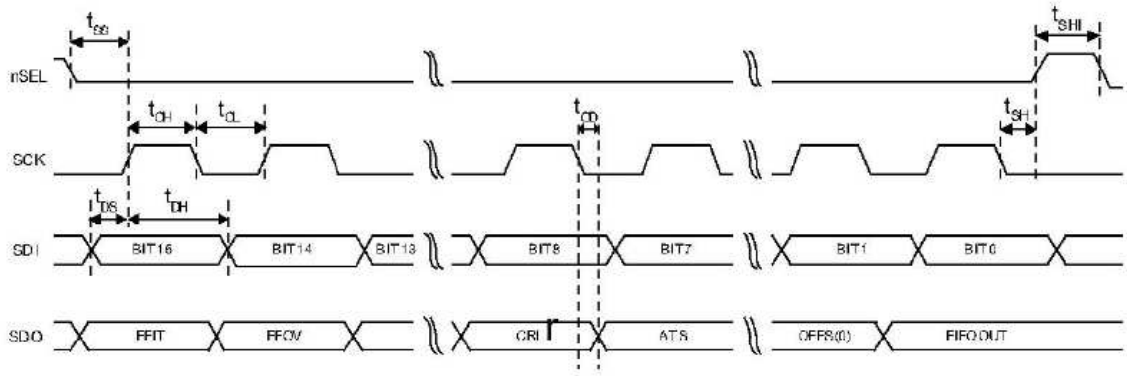


Figura 10. Diagrama de tiempos de la comunicación SPI.

Las señales que van a usarse en esta comunicación las que se detallan a continuación:

- nSEL: se trata de una señal digital activa a nivel bajo usada para seleccionar al dispositivo (ver figura 10).

- SCK: es la señal de reloj que sincroniza la comunicación entre ambos elementos, los datos de entrada son tomados en los flancos de subida de dicha señal (ver figura 10).

- SDI: (data input of the serial control interface), es una entrada desde el punto de vista del RF12, cada bit es tomado en el flanco de subida de SCK.

- SDO: (serial data output), al transmitir un comando, el RF12 suele ir devolviendo a la par información del dispositivo, tal y como se muestra en la figura 10, posteriormente se discutirá qué información se devuelve.

2.2.1.2.2 Comandos.

Los comandos son palabras de 16 bits, son los siguientes:

- Configuration setting (**80xxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	0	0	0	0	0	el	ef	b1	b0	x3	x2	x1	x0

El comando lo marca el byte superior, es decir, el valor 0x80 en este caso. El resto de bits vienen a ser los siguientes datos:

- el: habilita el registo interno de datos, si el este registro es usado deberá ponerse el pin FSK del RF12 a nivel alto.
- ef: activa el modo FIFO.
- b0-b1: permiten elegir la banda de la transmisión-recepción:

B1	B0	Frecuencia (MHz)
0	0	315
0	1	433
1	0	868
1	1	915

- Los 4 últimos bits marcan la capacidad con la que se va a cargar la antena: en nuestro caso seleccionamos 12 pF (0111).

- Administración de potencia (**82xxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	0	0	0	1	0	eb	ebb	et	es	ex	et	ew	dc

- er: activa el receptor, es decir, la cadena de recepción
- ebb: activa zona de banda base del receptor
- et: activa el PLL, el PA y comienza la transmisión si el registro TX está activado.
- es: activa el sintetizador.
- ex: enciende el cristal oscilador
- eb: activa el detector de nivel bajo de batería.
- ew: activa el temporizador despertador
- dc: deshabilita la salida del reloj.

- Ajuste de frecuencia (**Axxxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	0	F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	F0

Se trata de un parámetro de 12 bits que debe dar lugar a un valor correspondido entre 96 y 3903. Si el valor es excedido de ese rango, el valor anterior es guardado. La frecuencia central de la banda es calculada según la siguiente fórmula:

$$f_o = 10 * C_1 * \left(C_2 + \frac{F}{4000} \right) [MHz]$$

Donde F es el valor dado por los 12 bits y, C_1 y C_2 vienen dados por la banda de frecuencias según la tabla:

Frecuencia (MHz)	C_1	C_2
315	1	31
433	1	43
868	2	43
915	3	30

- Régimen binario (**Cxxxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	0	1	1	0	cs	r6	r5	r4	r3	r2	r1	r0

La tasa binaria real en modo de transmisión, así como la tasa esperada en el receptor es determinada por los 7 bits rx y el bit cs, según la fórmula:

$$BR = \frac{\frac{10000}{29}}{\frac{R+1}{(1+7*cs)}}$$

Donde BR es el régimen binario deseado, en el receptor por tanto sacamos el valor R despejando la ecuación:

$$R = \left(\frac{\frac{10000}{29}}{\frac{(1+7*cs)}{BR}} \right) - 1$$

Donde BR está dado en Kbps.

- Control del receptor (**9xxxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	1	0	p20	d1	d0	I2	I1	I0	G1	G0	r2	r1	r0

- P20: función del pin 20, a 1 es para VDI y a 0 para interrupción de entrada.

- VDI (Valid data indicador) los bits d1 y d0 muestran, el tipo de respuesta frente a la llegada de un dato:

d1	d0	Respuesta
0	0	Rápida
0	1	Media
1	0	Lenta
1	1	Siempre on

- I2,I1,I0. Sirven para elegir el ancho de banda de la recepción en banda base:

I2	I1	I0	BW [Khz]
0	0	0	reservado
0	0	1	400
0	1	0	340
0	1	1	270
1	0	0	200
1	0	1	134
1	1	0	67
1	1	1	reservado

- G1, G0: Ganancia del LNA relativa al máximo.

G1	G0	Ganacia, respecto al máximo en dB
0	0	0
0	1	-6
1	0	-14
1	1	-20

- R2,R1,R0: umbral de detección RSSI (potencia recibida)

R2	R1	R0	RSSI umbral
0	0	0	-103
0	0	1	-97
0	1	0	-91
0	1	1	-85
1	0	0	-79
1	0	1	-73
1	1	0	-67
1	1	1	-55

- Filtrado de datos (**C2xxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	0	0	1	0	al	ml	1	s	1	F2	F1	F0

- Al: Recuperación del reloj (clock recovery CR), si está a 1. La recuperación empezará en modo rápido, tras la recuperar para a modo lento.

- Ml: Bloqueo de la recuperación del reloj.

  1: Modo rápido, rápido ataque y rápida liberación, se recomienda un preámbulo de 6 a 8 bits (101010101...)
  0: Modo lento, rento ataque y lenta liberación, se recomienda un preámbulo de 10 a 14 bits (101010101010...)

  El modo lento requiere una mayor precisión en los tiempos de bit, ver comando de Régimen binario.

- S: Selecciona el tipo de filtro de datos, 0 se trata de un filtrado digital, mientras que 1 usa un filtro analógico.

- F2-F0: DQD umbral. Muestra el parámetro de buena calidad de señal. Debería ser menor que 4 en el caso de que la tasa de bits está cerca de la desviación.

- FIFO y modo de RESET (**CAxxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	1	0	1	0	F3	F2	F1	F0	1	al	ff	dr

  - F3-F0: nivel de interrupción de la FIFO, se genera una interrupción cuando el nivel de bits alcanzados es recibido.
  - Al: establece la entrada de la condición de FIFO llena.

    0: por patrón de sincronización
    1: siempre llena.

    El patrón de sincronización es 2DD4h.
  - FF: la FIFO será habilitada tras la recepción del patrón de recepción, la FIFO llena se parará tras poner a cero este bit.
  - Dr: sensibilidad al RESET, el sistema se resetea al observar un determinado voltaje umbral. Si este bit está a 0 un glitch de 200mV puede provocar un reset.

- Lectura de la FIFO del receptor (**8000h**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Con este comando se leer 8 bits desde la FIFO del receptor. El bit 6 ef, debe estar a en el comando de configuración. Tras transmitir ese comando al RF12, se recibe el dato recibido, es el propio RF12 quién avisa de que hay un dato en la FIFO.

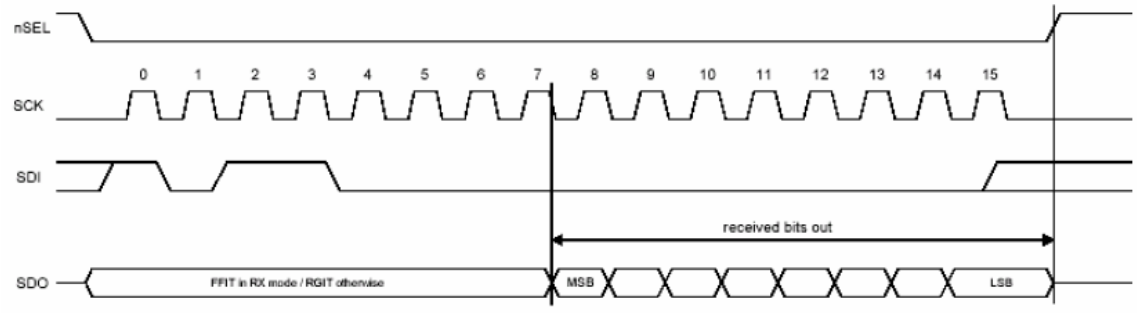


Figura 11. Diagrama de tiempos

En la figura 11 se muestra como el RF12 escribe en SDO el byte recibido.

- Control y configuración del Transmisor (**98xxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	1	1	0	0	mp	M3	M2	M1	M0	0	P2	P1	P0

Mp,m3-m1: parametros de la modulación FSK. La frecuencia de salida resultante puede ser calculada como:

$$f_{out} = f_o + (-1)^{SIGN} * (M + 1) * (15Khz)$$

donde f_o es la frecuencia central de la banda, M el valor binario obtenido de los 4 bits, y SIGN = mp XOR (FSK input).

- P2-p1-p0: es la potencia de salida del transmisor. El valor de salida expresado en relación ala salida de la antena, la cual depende de la impedancia real de la misma.

P2	P1	P0	Potencia de salida relativa dB
0	0	0	0
0	0	1	-3
0	1	0	-6
0	1	1	-9
1	0	0	-12
1	0	1	-15
1	1	0	-18
1	1	1	-21

- Control del registro de transmisión (**B8xxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	1	1	0	0	0	T7	T6	T5	T4	T3	T2	T1	T0

Con este comando se mandan los 8 bits hacia el registro de transmisión, para poder ser enviados el bit E1 del registro de configuración debe de estar a 1.

- Temporizador despertador (**Exxxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	R4	R3	R2	R1	R0	M7	M6	M5	M4	M3	M2	M1	M0

- Detector de batería baja y divisor del reloj (**C0xxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	0	0	0	0	D2	D1	D0	V4	V3	V2	V1	V0

Los 5 bits V4-V0, marcan el umbral para el cual se considera que la batería e está baja, este valor se calcula:

$$v_{lb} = 2.2 + v * 0.1[V]$$

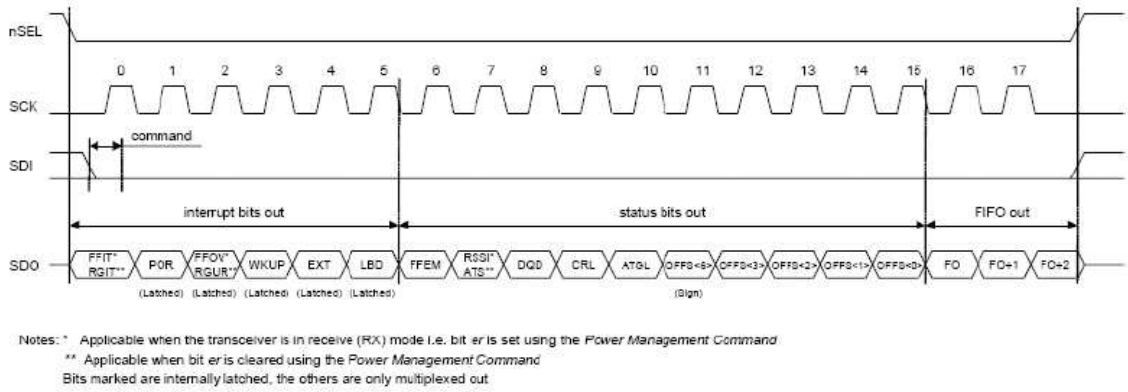
El divisor de reloj se rige por la tabla:

D2	D1	D0	Divisor del reloj [MHz]
0	0	0	1
0	0	1	1.25
0	1	0	1.66
0	1	1	2
1	0	0	2.5
1	0	1	3.33
1	1	0	5
1	1	1	10

- Lectura del registro de estado (**00xxh**):

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Se envía un 0x0000 al RF12 y el dispositivo envía una respuesta como la del siguiente diagrama de tiempos:



Notes: * Applicable when the transceiver is in receive (RX) mode i.e. bit *er* is set using the *Power Management Command*
** Applicable when bit *er* is cleared using the *Power Management Command*
Bits marked are internally latched, the others are only multiplexed out

Figura 12. Detalle de la transmisión de un comando.

Se van leyendo en los flancos de subida los siguientes parámetros:

- RGIT: registro TX listo para recibir un nuevo byte.
- FFIT: el número de bits de datos en RX FIFO ha alcanzado el límite.
- POR: Power on-reset.
- RGUR: el registro TX está siendo usado, evitar sobre escribir.
- FFOV: RX FIFO overflow.
- WKUP: despertador overflow.
- EXT: Pin 16, a nivel bajo, posible interrupción.
- LBD: detección de batería baja.
- FFEM: FIFO vacía.
- ATS: sintonizador de antena detecta un fuerte nivel de señal.
- RSSI: la cantidad de señal esperada está por debajo de lo pre-programado.
- DQD: detector de la calidad a la salida.
- CRL: circuito de detección del reloj bloqueado.
- ATGL: conmuta en cada ciclo del AFC
- OFFS(6): MSB del offset de frecuencia (signo)
- OFFS(3)-OFFS(0): valor del offset añadido al parámetro de control de frecuencia.

2.3 Servidor Web.

2.3.2 Ethernet y TCP/IP.

Ethernet es el nombre de una tecnología de redes de computadoras de área local (LANs) basada en tramas de datos. El nombre viene del concepto físico de ether. Ethernet define las características de cableado y señalización de nivel físico y los formatos de trama del nivel de enlace de datos del modelo OSI. Ethernet se refiere a las redes de área local y dispositivos bajo el estándar IEEE 802.3 que define el protocolo CSMA/CD. Actualmente, existe una gran incidencia de redes basadas en la norma IEEE 802.3, desarrollar un sistema empotrado que pueda comunicarse usando dicha norma es una buena idea, puesto que permite dar un acceso potencialmente grande a dicho sistema.

La familia de protocolos de Internet, es un conjunto de protocolos de red que implementa la pila de protocolos en la que se basa Internet, y que permiten la transmisión de datos entre redes de computadoras. En ocasiones se la denomina conjunto de protocolos TCP/IP, en referencia a los dos protocolos más importantes que la componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP), que fueron los dos primeros en definirse, y que son los más utilizados de la familia. Existen tantos protocolos en ese conjunto que llegan a ser más de 100 diferentes, entre ellos se encuentra el HTTP (HyperText Transfer Protocol), que es el que se utiliza para acceder a las páginas web, además de otros como el ARP (Address Resolution Protocol) para la resolución de direcciones, el FTP (File Transfer Protocol) para transferencia de archivos, y el SMTP (Simple Mail Transfer Protocol) y el POP (Post Office Protocol) para correo electrónico, TELNET para acceder a equipos remotos, entre otros.

El TCP/IP es la base de Internet, y sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN).

La familia de protocolos de Internet puede describirse por analogía con el modelo OSI, que describe los niveles o capas de la pila de protocolos, aunque en la práctica no corresponde exactamente con el modelo de Internet. En una pila de protocolos, cada nivel soluciona una serie de problemas relacionados con la transmisión de datos, y proporciona un servicio bien definido a los niveles más altos. Los niveles superiores son los más cercanos al usuario y tratan con datos más abstractos, dejando a los niveles más bajos la labor de traducir los datos de forma que sean físicamente manipulables.

2.3.1 Servidores web.

Un servidor web es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para lo que llamamos hipertextos, páginas web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

Sin embargo, el hecho de que HTTP y HTML estén íntimamente ligados no debe dar lugar a confundir ambos términos. HTML es un lenguaje de programación y un formato de archivo y HTTP es un protocolo

Cabe destacar el hecho de que la palabra servidor identifica tanto al programa como a la máquina en la que dicho programa se ejecuta. Existe, por tanto, cierta ambigüedad en el término, aunque no será difícil diferenciar a cuál de los dos nos referimos en cada caso. Nosotros nos centramos siempre a la aplicación.

Un servidor web se encarga de mantenerse a la espera de peticiones HTTP llevada a cabo por un cliente http, que solemos conocer como navegador. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita. A modo de ejemplo, al teclear www.google.com en nuestro navegador, éste realiza una petición HTTP al servidor de dicha dirección. El servidor responde al cliente enviando el código HTML de la página; el cliente, una vez recibido el código, lo interpreta y lo muestra en pantalla. Como vemos con este ejemplo, el cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página; el servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma.

Sobre el servicio web clásico podemos disponer de aplicaciones web. Éstas son fragmentos de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Hay que distinguir entre:

- Aplicaciones en el lado del cliente: el cliente web es el encargado de ejecutarlas en la máquina del usuario. Son las aplicaciones tipo Java o Javascript: el servidor proporciona el código de las aplicaciones al cliente; y éste, mediante el navegador, las ejecuta. Es necesario, por tanto, que el cliente disponga de un navegador con capacidad para ejecutar aplicaciones (también llamadas scripts). Normalmente, los navegadores permiten ejecutar aplicaciones escritas en lenguaje javascript y java, aunque pueden añadirse más lenguajes mediante el uso de plugins

- Aplicaciones en el lado del servidor: el servidor web ejecuta la aplicación; ésta, una vez ejecutada, genera cierto código HTML; el servidor toma este código recién creado y lo envía al cliente por medio del protocolo HTTP.

Las aplicaciones de servidor, suelen ser la opción por la que se opta en la mayoría de ocasiones para realizar aplicaciones web. La razón es que, al ejecutarse ésta en el servidor y no en la máquina del cliente, éste no necesita ninguna capacidad adicional, como sí ocurre en el caso de querer ejecutar aplicaciones javascript o java. Así pues, cualquier cliente dotado de un navegador web básico puede utilizar este tipo de aplicaciones.

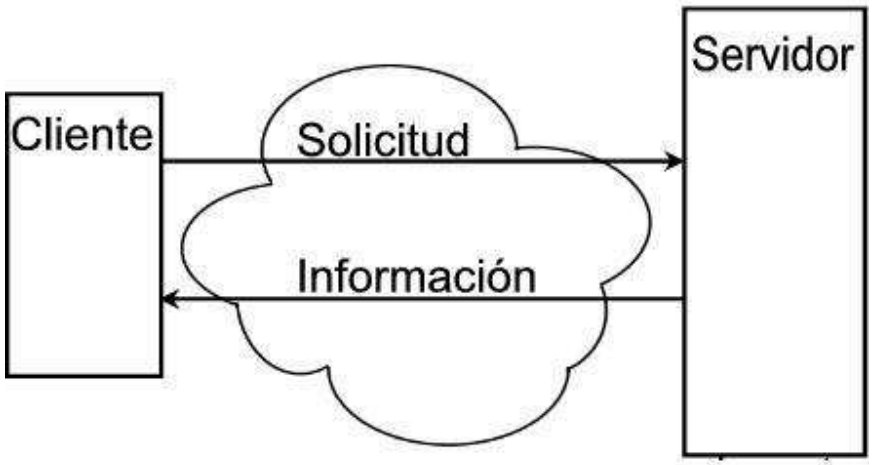


Figura 13. Esquema donde se muestra una comunicación cliente-servidor.

2.3.3 Acceso a Ethernet para Sistemas Empotrados.

Mediante la UART, el SPI, el bus I2C, etc, podemos comunicar un sistema empotrado con otro dispositivo para una determinada función. Sin embargo, dichos interfaces de comunicación son muy limitados. Es interesante por tanto, poder ampliar la capacidad de comunicación de los diseños. Ello podría permitir redistribuir de manera más ambiciosa los elementos que fuesen a integrar nuestro sistema. Por ejemplo, añadiendo una etapa de adaptación adecuada, podríamos usar el interfaz RS232, con lo que se conseguiría transmitir información a una distancia de una decena de metros. Si por el contrario, nos basamos en la norma RS485, se alcanzaría la centena de metros.

Se podría mejorar aún más el planteamiento, podríamos integrar en nuestro sistema un dispositivo acogido a un protocolo de comunicaciones determinado, se conseguiría por tanto, que nuestro sistema pudiese interactuar con los desarrollos de otros fabricantes, con lo que ello supone.

Nuestro objetivo, es poder acceder a servidor web mediante HTTP, que se apoya en TCP/IP y que a su vez se vale de Ethernet.

Hay varias soluciones para hacer esto, de entre ellas destacamos:

- ECN28J60 (Microchip): se trata de un microcontrolador que lleva implementada una pila TCP/IP en su interior. Mediante el interfaz I2C podemos comunicarnos con éste para poder acceder a la LAN. Se trata de una solución económica y simple para integrar en una placa.

- XPORT (Lantronix): es un dispositivo que permite dar acceso a una LAN implementando un puerto serie virtual, es decir, a efectos prácticos nos comunicamos con un puerto serie pero el dispositivo está incluido en una LAN.

- EM202 (Tibbo): es muy similar al anterior, hace lo mismo, nos permite acceder a una LAN tratándolo como si fuese un puerto serie más. Sin embargo hay una característica que lo hace más atractivo aún, es posible instalarle un sistema operativo para el que pueden desarrollarse aplicaciones, en un lenguaje similar al Basic.

Existe otra solución más, consiste en implementar una pila TCI/IP en un micro, añadiéndole el hardware que le permite acceder a la LAN. Es una solución barata, aunque el coste de desarrollo es alto.

2.3.4 EM202.

De las soluciones anteriores hemos elegido el EM202, desde el punto de vista económico es la solución intermedia, sin embargo desde el punto de vista de las prestaciones es la mejor. La posibilidad de usar un sistema compacto con potencialidad de alojar una aplicación, permite rebajar los tiempos de desarrollo, ya sea en el aspecto del hardware como en el del software.

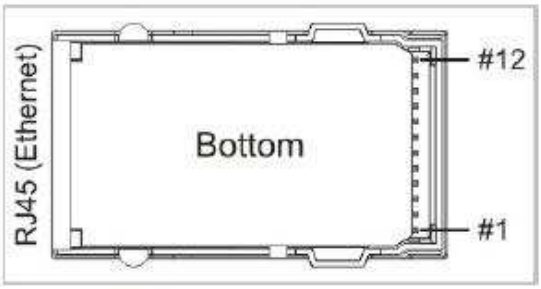
Se trata de un dispositivo muy compacto, cerrado por una carcasa metálica que le aísla de interferencias, con un conector RJ45 integrado así como 4 LEDs (dos de propósito general y otros 2 para el estado de la LAN).

Como se he destacado antes, es dual, es decir, se puede tratar como un dispositivo que permite el acceso a una LAN desde un puerto serie virtual (Serial-Ethernet converter), ó como un pequeño sistema empotrado capaz de albergar aplicaciones.

2.3.4.1 Características hardware.

- Dimensiones: 32,3x19x16mm
- Consumo: 230mA a 5VDC.
- Un puerto ethernet 100BaseT.
- Integrado circuito de acceso a LAN.
- Un puerto serie (TTL) con las señales Tx, Rx, RTS, CTS, DTR y DSR).
- 4 LEDs
- Firmware interno actualizable desde el puerto serie o bien desde la LAN.
- Certificados CE y FCC aprobados.

EM202 pin assignment is shown below.



#1	**MD (MD)**	Input	Mode selection pin
#2	**RST**	Input	Reset, active high
#3	**P3 (DTR)**	Input/output (output)	General-purpose input/output line Data terminal ready output
#4	**P2 (DSR)**	Input/output (input)	General-purpose input/output line Data set ready input
#5	**L3 (SG)**	Output (output)	LED output 3 (Green status LED)
#6	**L4 (SR)**	Output (output)	LED output 4 (Red status LED)
#7	**VCC**		Positive power input, 5V nominal, +/- 5%, app. 220mA
#8	**GND**		Ground
#9	**RX**	Input	Serial receive line
#10	**TX**	Output	Serial transmit line
#11	**P4 (CTS/SEL)**	Input/output (input)	General-purpose input/output line Clear to send input Full-/half-duplex selection input
#12	**P5 (RTS/DIR)**	Input/output (output)	General-purpose input/output line Request to send output (full-duplex mode) Data direction control output (half-duplex mode)

*Line functions defined by the application firmware are shown in **blue***

Figura 14. Asignación de PINES.

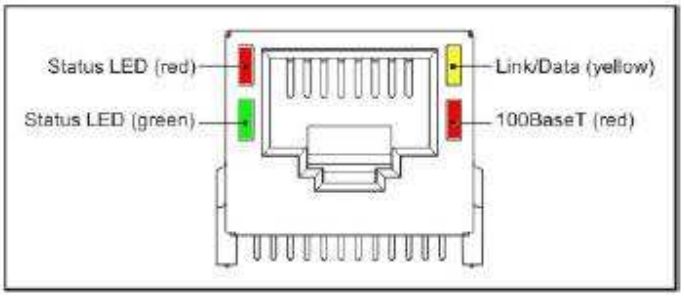


Figura 15. Significado de LEDs.

Figura 16. Aspecto del EM202

2.3.4.2 Características del sistema como autónomo.

- La aplicaciones se escriben en Tibbo BASIC.
- Flexibilidad de aplicación.
- Disponibilidad de entorno de desarrollo de aplicaciones (gratis).
- Integrado un servidor web (HTTP) con generación dinámica de contenido.
- El Tibbo Basic dispone de una serie de objetos de alto nivel, que permiten simplificar los desarrollos:

    - **Sock**: hasta 16 simultáneas conexiones TCP/IP.
    - **Ser**: UART comunicaciones.
    - **Io**: control de líneas de entrada/salida.
    - **Stor**: lectura y escritura en la EEPROM.
    - **Romfile**: añadir ROM estática al proyecto.

2.3.4.3 Entorno de desarrollo.

Taiko viene a ser el sistema que permite emplear el EM202 como un dispositivo autónomo, para que el EM202 pueda operar de ésta manera, es necesario instalar un firmware que viene a ser un sistema operativo conocido como TiOS System.

Desde el entorno de desarrollo (TIDE), podremos editar, programar, borrar, testear y arrancar nuestra aplicación.

En la figura 17 se muestra una pantalla del entorno de programación, su aspecto es similar al de otros programas de edición de código.

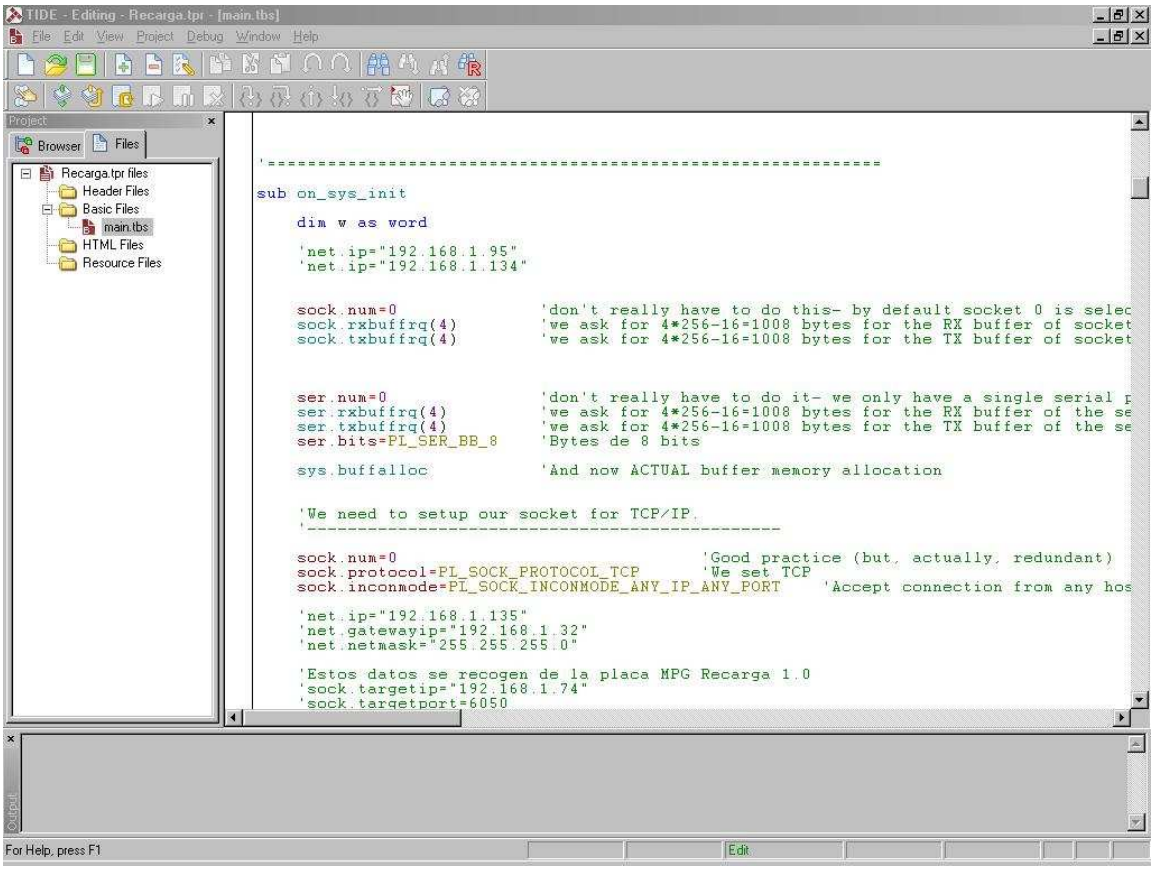


Figura 17. Screenshot del entorno.

El entorno dispone:

- Ventana de edición.
- Barra de menús con su estructura: Archivo, Edición, Ver, etc.
- Explorador de proyectos y ficheros.

En la Ayuda se especifican los objetos que componen el Tibbo Basic.

2.3.4.4. Tibbo Basic.

Guarda las estructuras típicas del Basic, es decir, IF-THEN, DO WHILE, SELECT CASE, etc. Al mismo tiempo también se conserva los tipos de variables, existen enteros, caracteres, cadenas, arrays, etc.

Sin embargo, esta especificación del Basic tiene un aspecto que le aproxima más a Visual Basic, se trata de los Objetos. Mediante dichos objetos, podremos controlar hardware de manera simple.

Los Objetos disponen de propiedades, eventos y métodos, mediante los cuales podremos acceder, controlar y cambiar las peculiaridades de los mismos, los objetos de Tibbo Basic son:

- Ser. Controla la UART, permite variar las especificaciones de dicha comunicación, dicho objeto dispone de un evento que "salta" ante la llegada de un byte.

- Sock. Controla la comunicación mediante los sockets empleados en las comunicaciones a través de una LAN. En dichas comunicaciones hay que indicar el destino de la comunicación, a quién se atiende en caso de una petición, el establecimiento o liberación de una comunicación, etc. Tiene asociado un evento a la llegada de tramas por la red.

- Sys. Es un objeto asociado a la inicialización y arranque del sistema. Se puede configurar los buffers de la UART, de los sockets, régimen binario, etc. Su evento salta al inicializarse el sistema.

- Pat. Permite controlar los LEDs de propósito general (Rojo y Verde) que el EM202 tiene, cada vez que se actúa sobre un LED, salta un evento.

- Io. No dispone de evento

- Stor. Controla una memoria no volátil, permitiendo guardar y leer registros.

- Net. Representa el interfaz Ethernet del dispositivo, sólo controla parámetros relacionados con la comunicación, pero sin tener capacidad de trasmitir o recibir datos.

- Button. Se asocia al botón, pues se le reserva una línea, salta un evento cuando el botón es presionado.

- Romfile. Objeto que controla un fichero en el que se pueden guardar datos de configuración, dicho fichero es leído desde el árbol de proyecto.

2.3.4.4. Páginas Web sobre el EM202.

Una de las principales características del EM202 es que permite alojar páginas web. Debido a la limitación de tamaño de la aplicaciones (64 Kbytes), las páginas que puede albergar son muy simples.

La principal aplicación de una página web sobre un EM202, es la de poder controlar un hardware determinado desde una web, veamos un ejemplo:

Vamos a implementar el control de uno de los LEDs de los que dispone el EM202 desde una web. Para llevar a cabo este proyecto se requieren los siguientes ficheros:

- global.tbh: donde se definen las funciones, constantes y variables globales a usar.
- main.tbs: donde se configura el hardware a usar.
- index.html: página web que se guarda en el EM202.
- post.html: se guarda el script basic que efectúa acciones relacionadas con los objetos de los que dispone la página web.

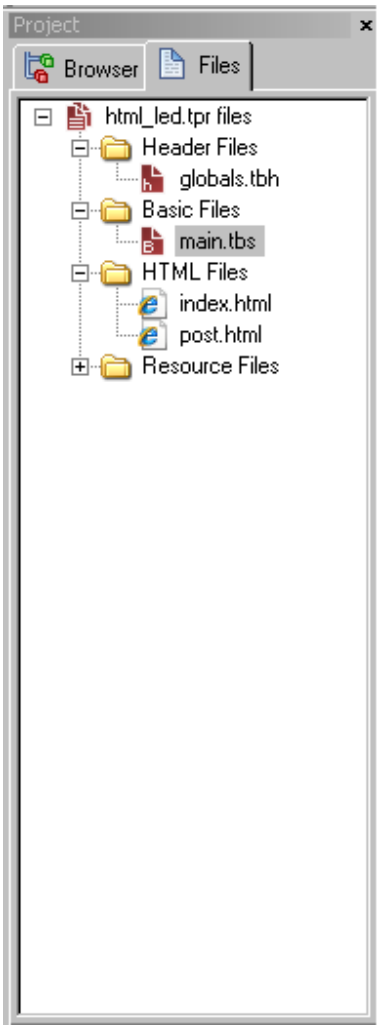


Figura 18. Árbol de proyecto de la aplicación.

El código de la página web está en el fichero **index.html**. Es el siguiente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
</HEAD>

<BODY>

<b>LED Control</b>
<br>
<div class="border1">
<table border="2" cellspacing="0" cellpadding="9" >
<form action="post.html" method="get">


<tr>
<td>Color</td>
<td><input type="checkbox" name="red">Red </td>
<td><input type="checkbox" name="green">Green</td>
</tr>

<tr>
<td><input type="text" name="pattern" size="10" class="input100"></td>
<td><input type="submit" value="Enter" tabindex="2"></td>
</tr>

</table>
<br>
<br>
</form>
</div>

</BODY>
</HTML>
```

Si vemos este código desde un programa explorador, tendremos:

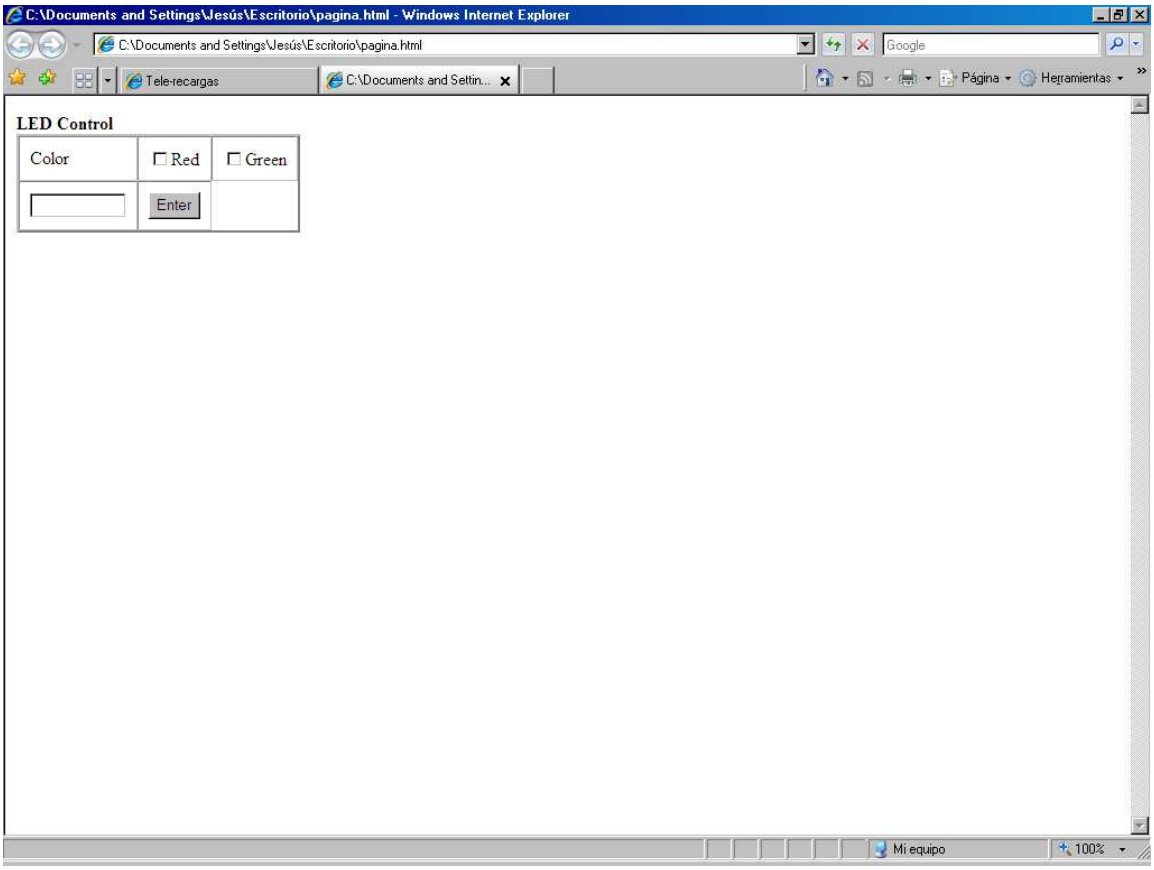


Figura 19. Vista la página asociada al código html.

El conjunto de acciones a desarrollar frente a la interacción de con dicha página es la siguiente:

```
<!DOCTYPE HTML PUBLIC “-//W3C//DTD HTML 4.0 Transitional//EN”>
<HTML>
<HEAD>
<META HTTP-EQUIV=”Refresh”
    CONTENT=”1; URL=index.html”>

</HEAD>

<BODY>

<?  include “globals.tbh”

        dim red_on as byte
        dim green_on as byte

        red_on = instr(1,sock.httprqstring,”red=on”,1)
```

```
        green_on = instr(1,sock.httprqstring,"green=on",1)
        pattern = get_http_argument(sock.httprqstring, "pattern=")

        if red_on > 0 then
                if green_on > 0 then
                        pat.play("B",PL_PAT_CANINT)
                else
                        pat.play("R",PL_PAT_CANINT)
                end if
        else
                if green_on > 0 then
                        pat.play("G",PL_PAT_CANINT)
                else
                        pat.play("-",PL_PAT_CANINT)
                end if
        end if

¿>


</BODY>
</HTML>
```

Cabe destacar de ese script, del empleo de variables asociadas a dicho dispositivo, cosa que no es posible en los scripts ordinarios. Se puede observar también el empleo del objeto pat, que permite controlar los LEDs.

Para poder actuar sobre los LEDs, se usan también funciones que están definidas en el fichero main.tbs, funciones que recogen las cadena de texto que se generan al hacer clic en cada uno de los objetos que integran la página web.

2.4 TRIACs

El TRIAC es un dispositivo semiconductor que pertenece a la familia de los dispositivos de control por tiristores. El TRIAC es en esencia la conexión de dos tiristores en paralelo pero conectados en sentido opuesto y compartiendo la misma compuerta. (ver figura 20).

Figura 20. Símbolo del TRIAC.

El TRIAC sólo se utiliza en corriente alterna y al igual que el tiristor, se dispara por la compuerta. Como el TRIAC funciona en corriente alterna, habrá una parte de la onda que será positiva y otra negativa.

La parte positiva de la onda (semiciclo positivo) pasará por el TRIAC siempre y cuando haya habido una señal de disparo en la compuerta, de esta manera la corriente circulará de arriba hacia abajo (pasará por el tiristor que apunta hacia abajo).

La parte negativa de la onda (semiciclo negativo) pasará por el TRIAC siempre y cuando haya habido una señal de disparo en la compuerta, de esta manera la corriente circulará de abajo hacia arriba (pasará por el tiristor que apunta hacia arriba) .

Para ambos semiciclos, la señal de disparo se obtiene de la misma patilla (la puerta o compuerta).

Lo interesante es, que se puede controlar el momento de disparo de esta patilla y así, controlar el tiempo que cada tiristor estará en conducción. Recordar que un tiristor sólo conduce cuando ha sido disparada (activada) la compuerta y entre sus terminales hay un voltaje positivo de un valor mínimo para cada tiristor.

Entonces, si se controla el tiempo que cada tiristor está en conducción, se puede controlar la corriente que se entrega a una carga y por consiguiente la potencia que consume. Gracias a esto podemos usar el “control dimming”, control que permite regular la potencia entregada a cargas resistivas, de manera que no se incremente el ruido por conmutación.

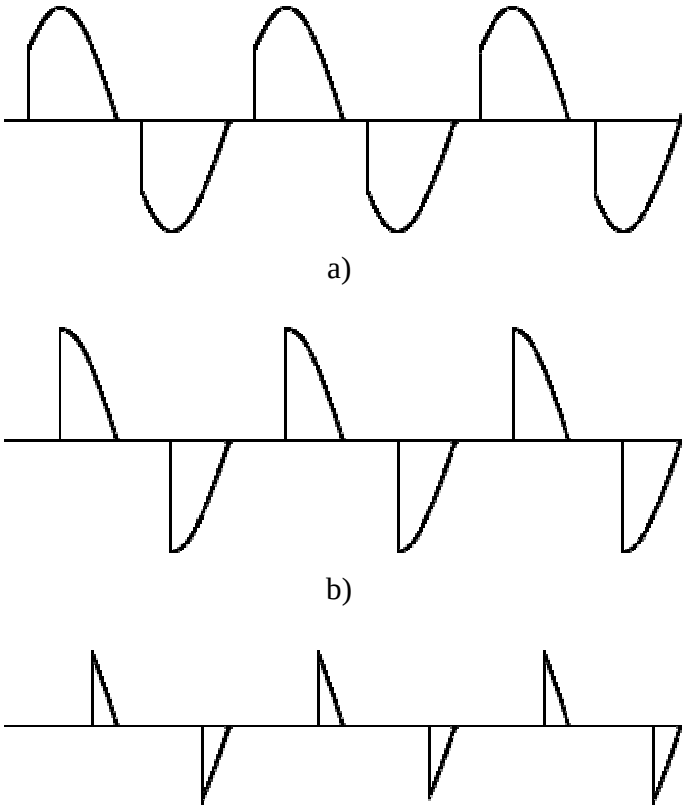
2.4.1 Control Dimming.

Aunque a priori, una de las aplicaciones del TRIAC es el desarrollo de relés de estado sólido para corriente alterna, es decir, cortando y abriendo la intensidad que cruza el dispositivo, existe una aplicación que permite regular el control de la potencia consumida por cargas resistivas. Hablamos del control dimming.

Mediante el control dimming se efectúa un control PWM aunque para corriente alterna. Es decir, si nos centramos en una línea monofásica doméstica de 220 VAC, los sistemas que usan dicha señal para obtener la energía que requieren, consumen la totalidad de la misma. Si tenemos una carga resistiva (brasero, bombilla, calentador, etc), podemos permitirnos cortar dentro de cada periodo de la onda de 50 Hz, la conducción de TRIAC.

Un tiempo estimado, podría ser 10%,20%, 50%, ... del intervalo, el efecto conseguido es la reducción del consumo en el porcentaje elegido.

En la figura 21, se muestra el efecto de dicha reducción sobre la onda de potencia:



a)

b)

c)

Figura 21. Control dimming a 75%, 50% y 25%.

El efecto que se observaría sobre una bombilla, sería la reducción de la intensidad lumínica.