SISTEMAS INTEGRADOS CON ARDUINO

JOSÉ RAFAEL LAJARA VIZCAÍNO
JOSÉ PELEGRÍ SEBASTIÀ

Alfaomega

marcombo
ediciones técnicas

Sistemas integrados con Arduino

José Rafael Lajara Vizcaíno

José Pelegrí Sebastiá

Alfaomega

marcombo
ediciones técnicas

Datos catalográficos
Lajara, José Rafael y Pelegrí, José
Sistemas integrados con Arduino
Primera Edición

Alfaomega Grupo Editor, S.A. de C.V., México

ISBN: 978-607-622-046-7

Formato: 17 x 23 cm Páginas: 320

Sistemas integrados con Arduino
José Rafael Lajara Vizcaíno y José Pelegrí Sebastiá
ISBN: 978-84-267-2093-1, edición en español publicada por MARCOMBO, S.A., Barcelona, España

Primera edición: Alfaomega Grupo Editor, México, enero 2014

Pitágoras 1139, Col. Del Valle, 03100, México D.F.

Miembro de la Cámara Nacional de la Industria Editorial Mexicana
Registro No. 2317

Pág. Web: **http://www.alfaomega.com.mx**
E-mail: **atencionalcliente@alfaomega.com.mx**

ISBN: 978-607-622-046-7

Nota importante:
La información contenida en esta obra tiene un fin exclusivamente didáctico y, por lo tanto, no está previsto su aprovechamiento a nivel profesional o industrial. Las indicaciones técnicas y programas incluidos, han sido elaborados con gran cuidado por el autor y reproducidos bajo estrictas normas de control. ALFAOMEGA GRUPO EDITOR, S.A. de C.V. no será jurídicamente responsable por: errores u omisiones; daños y perjuicios que se pudieran atribuir al uso de la información comprendida en este libro, ni por la utilización indebida que pudiera dársele.

Edición autorizada para venta en México y todo el continente americano.

Impreso en México. Printed in Mexico.

Empresas del grupo:

México: Alfaomega Grupo Editor, S.A. de C.V. – Pitágoras 1139, Col. Del Valle, México, D.F. – C.P. 03100.
Tel.: (52-55) 5575-5022 – Fax: (52-55) 5575-2420 / 2490. Sin costo: 01-800-020-4396
E-mail: atencionalcliente@alfaomega.com.mx

Colombia: Alfaomega Colombiana S.A. – Calle 62 No. 20-46, Barrio San Luis, Bogotá, Colombia,
Tels.: (57-1) 746 0102 / 210 0415 – E-mail: cliente@alfaomega.com.co

Chile: Alfaomega Grupo Editor, S.A. – Av. Providencia 1443. Oficina 24, Santiago, Chile
Tel.: (56-2) 2235-4248 – Fax: (56-2) 2235-5786 – E-mail: agechile@alfaomega.cl

Argentina: Alfaomega Grupo Editor Argentino, S.A. – Paraguay 1307 P.B. Of. 11, C.P. 1057, Buenos Aires,
Argentina, – Tel./Fax: (54-11) 4811-0887 y 4811 7183 – E-mail: ventas@alfaomegaeditor.com.ar

Los autores queremos agradecer la colaboración en la revisión del libro, así como las sugerencias y mejoras propuestas, en especial de Juanjo Pérez.

También queremos agradecer a David Aparisi, Borja Fuster y Tomás Sogorb por la ayuda prestada en la realización de este libro mediante revisiones, material prestado y colaboración en algunos programas.

Índice

1. Introducción 1

1.1 Ecosistema de Arduino 2

1.2 Software y hardware abierto 4

1.3 Sistemas embebidos 6

1.4 Componentes básicos en electrónica 7

1.5 Bibliografía 9

2. Hardware 11

2.1 Esquemático de Arduino UNO 11

2.2 Alimentación 12

2.3 Microcontrolador 14

2.4 Periféricos internos 17

2.4.1 Sistema de reloj 18

2.4.2 Temporizadores 19

2.4.3 PWM 19

2.4.4 I^2C 20

2.4.5 SPI 21

2.4.6 USART 22

2.5 CSP 23

2.6 USB 24

2.7 Conector de pines 25

2.8 Tipos de placas 27

2.9 Tipos de *shields* 30

2.10 Bibliografía 31

3. Programación 33

3.1 IDE 33

3.1.1 Instalación 33

3.1.2 Descripción del entorno 36

3.1.3 Otros entornos de desarrollo 40

3.2 Comenzando a programar con Arduino 41

3.3 Programación 43

3.3.1 Directivas del precompilador 44

3.3.2 Comentarios 47

3.3.3 Tipos de datos 48

3.3.4 Operadores 56

3.3.5 Estructuras de decisión 61

3.3.6 Estructuras de repetición 65

3.3.7 Funciones 69

3.4 Estudio del programa Blink 74

3.5 Depuración 79

3.6 Bibliografía 83

4. Acceso al hardware 85

4.1 Entradas y salidas digitales 85

4.1.1 Entrada 85

4.1.2 Salida 89

4.1.3 Puertos 90

4.1.4 Ejemplo: Push-button y LED 91

4.1.5 Ejemplo: secuencia de luces del coche fantástico 94

4.2 Conversión analógico-digital 96

4.2.1 Ejemplo: sensor de temperatura 99

4.3 Salida analógica 102

4.3.1 Ejemplo: control de iluminación de un LED *105*
4.4 Bibliografía 109
5. Comunicaciones 111
5.1 USART 111
5.1.1 Comunicación serie usando librería software *114*
5.1.2 Manejo de strings *115*
5.1.3 Ejemplo: GPS *117*
5.2 I^2C/SPI 126
5.2.1 Ejemplo: RTC *130*
5.3 Stream 137
5.4 Bibliografía 138
6. Periféricos del microcontrolador 139
6.1 Temporización 139
6.1.1 Ejemplo: cambiar la frecuencia de trabajo de la salida PWM *142*
6.2 Interrupciones 147
6.2.1 Externas *149*
6.2.2 Internas *151*
6.2.3 Watchdog *159*
6.3 Sleep 161
6.3.1 Ejemplo: Sleep y despertar con evento externo *163*
6.4 Memorias 165
6.4.1 RAM *165*
6.4.2 Flash *166*
6.4.3 EEPROM *168*
6.5 Bibliografía 173
7. Librerías 175
7.1 SD Card 177

7.1.1 Ejemplo: datalog *179*

7.2 Ethernet 182

7.2.1 Ejemplo: servidor web *187*

7.3 Comunicaciones inalámbricas 196

7.3.1 RFID *197*

7.3.2 Wifi *201*

7.3.3 ZigBee *209*

7.4 Pantalla TFT 224

7.4.1 Ejemplo: graficar señal analógica *226*

7.5 Motores 233

7.5.1 Motores paso a paso *233*

7.5.2 Servomotores *238*

7.6 Bibliografía 246

8. Comunicación con otras plataformas 249

8.1 Graficar valores con MakerPlot 249

8.2 Firmata 251

8.2.1 Ejemplo: programa en Arduino *254*

8.2.2 Ejemplo: aplicación en PC con .NET *256*

8.3 Intérprete de comandos 257

8.4 LabVIEW 261

8.5 MATLAB 263

8.6 Android 264

8.6.1 Ejemplo: programa en Arduino *267*

8.6.2 Ejemplo: programa en Android *267*

8.7 Servicios web 273

8.8 *Internet of things* 276

8.9 Bibliografía 282

9. Núcleo de Arduino 285

9.1 *Toolchain* 285

9.1.1 Preprocesador *286*

9.1.2 Compilador *287*

9.1.3 Linker *288*

9.1.4 Bootloader y programadores *289*

9.2 Core 290

9.3 Crear nuevas librerías 293

9.3.1 Ejemplo: conmutación de pin *297*

9.4 Crear nuevas placas 299

9.5 Bibliografía 302

10. Apéndices 303

1. Introducción

Un sistema empotrado, a veces también traducido como «embebido», es aquel sistema electrónico en el que confluyen *hardware* y *software*. Normalmente tiene un pequeño procesador que puede ser un microprocesador o un microcontrolador como parte central.

Los desarrolladores de sistemas electrónicos embebidos tienen la necesidad de elegir dicho procesador de entre la gran variedad de fabricantes que existen en el mercado: Microchip, Renesas, ARM, Atmel, Freescale, Philips, Sharp, etc. Además, para realizar una validación del diseño de forma rápida tienen que acudir a las placas de evaluación que suelen tener los fabricantes con un coste más o menos razonable. Por otro lado, dependiendo de la elección tendrán que disponer del *software* necesario y elegir uno de los diferentes compiladores existentes para la programación del micro usado. Esto conlleva la necesidad de estudiar cómo funciona este microcontrolador y compilador, porque aunque la mayoría se basa en el uso del lenguaje C, suelen existir discrepancias entre los distintos compiladores, aunque sean para el mismo micro. Además, en algunos casos hace falta diseñar un *hardware* específico, una tarea muy especializada y que requiere de herramientas complejas o contactar con terceros para realizar la fabricación.

Todo esto hace que sea complejo abordar aplicaciones sencillas para personas ajenas a este mundo tan apasionante, como profesionales del sector textil, del hostelero, el del diseño o cualquier otro sector donde se están introduciendo sencillos sistemas electrónicos embebidos. Incluso el precio de algunas aplicaciones y herramientas hace que, para gente aficionada que realiza diseños para su propio uso, sea complicado afrontar según qué proyectos.

La aparición de Arduino resuelve este problema. Arduino nace con la idea de hacer las cosas fáciles y sencillas a los profesionales de cualquier sector no

relacionado directamente con la tecnología electrónica. Incluso se ha extendido mucho entre los propios desarrolladores del área de la electrónica para reducir tiempos de desarrollo.

1.1 Ecosistema de Arduino

Arduino se basa en trabajos previos, el primero de ellos es el lenguaje de programación Processing y su entorno de desarrollo, diseñado especialmente para personas sin experiencia en programación. Posteriormente se creó la plataforma Wiring, era una plataforma *hardware* para realizar prototipos rápidos de aplicaciones. La placa de Wiring incluía un pequeño «ordenador» programable bajo Wiring. El proyecto Arduino fue una evolución de Wiring para hacerlo más sencillo y barato [1].

El lenguaje de programación y las herramientas derivan en último término de C, uno de los lenguajes más extendidos. Sobre este lenguaje se añaden elementos de Processing/Wiring para hacerlo lo más sencillo posible. Mediante este lenguaje se describirá el comportamiento del «computador» de la placa Arduino. Los ficheros que contienen el texto del programa se denominan Sketchs.

Arduino empieza con un proyecto de Massimo Banzi, un alumno en el Interaction Design Institute Ivrea (Italia) en el año 2005 y toma forma con la realización de una tesis sobre *hardware*. Finalmente un grupo de investigadores realiza una optimización de este para que sea económico y accesible a la comunidad [2][3]. En la actualidad el equipo principal de desarrollo de Arduino está compuesto por Massimo Banzi, David Cuartielles, Tom Igoe, David Mellis y Gianluca Martino.

Arduino se creó con un fuerte enfoque en el movimiento del código abierto, esto es, la compartición del conocimiento. Desde entonces Arduino ha ido creciendo sin parar gracias al aporte de investigadores, empresas y aficionados de todo el mundo.

El ecosistema Arduino está formado por tres elementos principales:

- El *hardware*: placa electrónica básica para hacer desarrollos rápidos y económicos. Este *hardware* es libre y permite una forma simple de aprender electrónica básica desmenuzando los componentes que forman sus circuitos. También permite ampliar el *hardware* mediante el uso de otras placas (*shields*) que se conectan a la del microprocesador.

- El *entorno de programación*: que permite de forma muy sencilla (teniendo conocimientos básicos de lenguaje de programación) la programación del *hardware* escogido. Además, este entorno es multiplataforma (Windows, Mac y Linux) y únicamente necesita un cable USB para conectarse a la placa.
- Y una *comunidad* que participa activamente en nuevos desarrollos, ideas y test. Se puede consultar a través del foro y de la wiki tanto el *hardware* como el *software* de forma libre, tanto en la web oficial de Arduino como de otros realizados por la comunidad.

El entorno integrado de desarrollo o IDE (*Integrated Development Environment*) es un *software*, disponible en la web de Arduino de forma gratuita, que permite realizar Sketchs y compilarlos. La compilación es una «traducción» a un formato entendible por el procesador que lleva la placa Arduino. Este *software* también permite realizar la programación del chip, en este caso un microcontrolador Atmel, que se estudiará en el próximo capítulo (o ARM en algunas placas). Para poder realizar la programación del chip sin la necesidad de *hardware* extra, el chip tiene que tener un *bootloader*, esto es, un pequeño programa ya incorporado que permite la comunicación con el IDE y el cambio del resto del código dentro del chip.

El éxito y repercusión que ha tenido se debe sobre todo a varios factores: muy bajo coste del material a utilizar, fácil de entender y manejar para crear aplicaciones concretas y finalmente se trata de un *software* y *hardware* abierto, con una licencia que permite su estudio, reproducción y modificación.

Existe una gran cantidad de aplicaciones realizadas por la comunidad Arduino muy interesantes. A continuación se listan a modo ilustrativo algunas de ellas:

- **XOscillo:** *software* que convierte la placa Arduino en un osciloscopio (instrumento que permite la captura de señales analógicas) o en un analizador lógico (instrumento que permite la captura de señales digitales), incluso puede realizar un procesado de la señal como un análisis en frecuencia mediante FFT (*Fast-Fourier Transform*) [4].
- **Arduinome:** es un controlador MIDI (*Musical Instrument Digital Inferface*). Instrumento musical con interfaz digital, que imita el Monome. Monome consiste en una matriz de botones con iluminación, permite controlar diferentes dispositivos como, por ejemplo, un secuenciador por pasos que hace sonar un kit de percusión [5].

- **Impresora 3D PrintrBot:** impresora en 3D basada en la plataforma Arduino Mega 2560 [6].
- **Ardupilot:** consiste en un sistema automático con GPS y sensores inerciales construido con arduiIMU+ para la estabilización y para pilotar aviones de aeromodelismo y drones de forma remota [7].
- **Video Game Shield:** sistema que permite convertir a Arduino en un sistema de videojuegos. Reproduce gráficos, textos, música y efectos sonoros, además de permitir la conexión con TVs [8].
- **DIY Cellphone:** teléfono móvil con teclado, pantalla, micrófono, altavoces, capacidad para hacer llamadas a través de la línea GSM, mandar mensajes y mantener una agenda [9].
- **Ardusat:** es un nanosatelite de código abierto basado en Arduino. Equipado con paneles solares, cámaras, sensores y transmisores UHF. Utiliza 17 placas Arduino. Fue lanzado con éxito al espacio el 3 de agosto de 2013 [10].

1.2 Software y hardware abierto

El concepto de *software* abierto (*Open Source Software*, OSS) tuvo sus orígenes en los primeros años de la era de la computación; cuando solo había unas docenas de ordenadores los programadores compartían su código y conocimiento. Con la comercialización de aplicaciones esta tendencia se fue reduciendo. Algunos desarrolladores pensaron que era una tendencia negativa y comenzaron a aparecer los primeros movimientos de *software* libre y, de él, el *software* abierto.

El *software* abierto busca ofrecer a todo el mundo el acceso al código, permitir su modificación y redistribución. La principal herramienta para lograr este objetivo es el mecanismo de licencias de código abierto.

Entre las licencias más populares en el mundo del código libre/abierto cabe destacar:

- GPL (*GNU General Public License*): licencia creada por el grupo GNU (*GNU's Not UNIX*) que busca asegurar los derechos de usar, estudiar, compartir y modificar un código. A su vez añade la obligación de que el código derivado utilice la misma licencia.
- CC (*Creative Common*): es una familia de licencias creada por la organización homónima. El autor de una obra selecciona una licencia CC

de acuerdo con los derechos que quiera otorgar a sus usuarios. Algunas de estas licencias permiten o restringen el uso comercial o permiten o restringen modificaciones y cambios de licencia en trabajos derivados.

- BSD: es una licencia muy permisiva que permite trabajos derivados sin restricción alguna. El autor simplemente mantiene el reconocimiento de sus contribuciones.

En Arduino se emplean principalmente las licencias GPL y Creative Common CC BY-SA. Esta última permite obras derivadas siempre que se cite al autor original y se utilice la misma licencia o similar.

Las licencias anteriores están pensadas para *software*. Para el *hardware* en algunos casos se pueden aplicar directamente las licencias anteriores, como en el código de dispositivos programables o reconfigurables. En otros casos el *hardware* necesita algunas particularidades ya que es de naturaleza diferente al *software*: hay elementos físicos, costes asociados, etc. En estos casos el *hardware* abierto simplemente puede consistir en proporcionar la documentación necesaria. En el caso de Arduino se tiene tanto la documentación como los ficheros de diseño.

Como consecuencia de lo anterior, alrededor de un proyecto de código abierto se puede formar una comunidad de personas que mantienen, mejoran, corrigen fallos, documentan o traducen el proyecto. Estas comunidades pueden ser más o menos organizadas y suelen utilizar diversas herramientas en Internet para comunicarse y compartir información. En el caso de Arduino se dispone de foros y wikis para compartir ideas y realizar preguntas; en la web oficial hay abundante documentación del *software* y *hardware*; y además se dispone de acceso a repositorios públicos (lugares de almacenaje) de los sistemas de control de versiones que alojan las distintas partes de Arduino; así desde estos repositorios no solo se pueden descargar los programas binarios sino también los códigos fuentes.

Un movimiento que puede salir muy beneficiado del código abierto es el DIY (*Do It Yourself*) o «hazlo tú mismo». Este movimiento aboga por que las personas creen sus propias cosas, desde objetos domésticos a industriales. Hay diferentes enfoques del DIY, que van desde movimientos políticos, económicos a educativos. Desde el punto de vista educativo y *hobbyist* se ha adoptado Arduino como una plataforma de referencia y como base para construir nuevos sistemas.

1.3 Sistemas embebidos

Un sistema embebido es el nombre que recibe un sistema electrónico diseñado específicamente para realizar unas tareas concretas. Normalmente lleva un microcontrolador o microprocesador y además tiene unos periféricos para interconectarse con otros dispositivos.

En un sistema embebido se tiene una parte *hardware* que consiste en toda la electrónica necesaria para operar adecuadamente. Dentro de esa electrónica, como se ha comentado antes, suele haber un sistema programable. El sistema programable puede ser un microcontrolador, microprocesador, DSP (*Digital Signal Processor*) o FPGA (*Field-Programmable Gate Array*). Estos elementos ejecutan un programa llamado *firmware*. Este programa implementaría parte de la funcionalidad del sistema.

El dispositivo programable normalmente requiere comunicarse con otros dispositivos. Para ello emplea diferentes medios de comunicación, por lo que los sistemas embebidos también pueden incorporar puertos serie, buses SPI o USB, wi-fi, etc.

Los sistemas embebidos también tienen un sistema de alimentación para proporcionar energía al sistema. Esta energía puede venir de la red eléctrica, baterías, energías renovables, etc.

Otros elementos que a veces pueden aparecer son sistemas de interfaz hombre-máquina como pantallas o teclados y sistemas de digitalización para convertir señales físicas en información para el procesador, o al revés: sistemas para generar señales.

Los sistemas embebidos están prácticamente omnipresentes en la sociedad actual. Ejemplos de sistemas embebidos son: ordenador de a bordo de vehículos, reproductores MP3, despertadores, etc.

Arduino permite implementar sistemas embebidos de una forma muy sencilla. Utilizándolo se dispone de un *hardware* genérico, el diseño del *hardware* se reduce a conectar placas de expansión o *shields* a la placa base de Arduino. De esta forma se elimina una de las partes que requiere conocimiento más específico y que supone fabricación. Así se permite que el desarrollador se concentre en la programación del *firmware* y en la funcionalidad del sistema.

El *firmware* es el código de los elementos programables, ya sea el microcontrolador, DSP o FPGA. Se escribe como texto en un determinado lenguaje, por ejemplo, en el caso de microcontroladores suele ser C. El código del *firmware* es traducido por un programa llamado compilador a un segundo código, denominado código máquina, que es entendido por el procesador. Este código es una secuencia de instrucciones que ejecutará el procesador. La combinación de estas instrucciones determina la forma en que el dispositivo funciona e interactúa con los demás elementos.

1.4 Componentes básicos en electrónica

Como se acaba de comentar, el diseño *hardware* del sistema se reduce enormemente al utilizar una placa ya hecha. No obstante siempre es conveniente tener ciertas nociones para saber qué está pasando, para conectar otros sistemas o para encontrar posibles problemas. Con esta excusa se expondrán muy brevemente algunos conceptos básicos sobre electrónica que se utilizarán en capítulos posteriores.

En electrónica las magnitudes físicas principales son la diferencia de potencial o tensión (V) y la intensidad de corriente (I). La tensión es la resta del potencial que existe entre dos puntos medida en voltios; normalmente se elige un punto como referencia de tensiones y se le asigna el valor 0, se le denomina masa. La corriente es la cantidad de carga eléctrica que se desplaza por un material en un tiempo determinado, medido en amperios.

Las dos grandes reglas que rigen el análisis de circuitos son las leyes de Kirchhoff. La primera ley de Kirchhoff indica que la suma de las corrientes en un nodo (punto del circuito donde se unen varias ramas) es igual a cero, es decir, las corrientes entrantes son iguales a las corrientes de salida. La segunda indica que en un tramo cerrado de un circuito la suma de las diferencias de tensión de todos los elementos es igual a cero, es decir, las caídas de tensión en todos sus elementos es igual a la tensión suministrada por las fuentes.

Los componentes básicos en electrónica son la resistencia (R), el condensador (C) y la bobina o inductancia (L). La resistencia es un elemento que se opone al paso de una corriente eléctrica, lo que provoca una caída de tensión en un circuito cerrado, de acuerdo con la ley de Ohm $V = I \cdot R$. El condensador se utiliza habitualmente como elemento de almacenamiento de tensión y la bobina de

corriente. Estos dos últimos presentan un comportamiento variable en la frecuencia, por lo que también son empleados para favorecer o rechazar (filtrar) unas frecuencias respecto a otras en una señal eléctrica.

Otros componentes son los diodos, transistores BJT, MOSFET y los amplificadores operacionales (AO). Estos elementos se fabrican utilizando materiales semiconductores. Con estos circuitos se pueden diseñar circuitos electrónicos básicos como puedan ser etapas amplificadoras, interruptores o conmutadores, etc. El funcionamiento básico del diodo consiste en ofrecer poca oposición al paso de corriente en un sentido, pero mucha en el sentido contrario. Los transistores suelen utilizarse como amplificadores o como interruptores.

También existen otros tipos de componentes, que permiten la realización de aplicaciones, como son reguladores para asegurar un nivel de tensión estable y conocido, LED para emitir luz (un tipo particular de diodo luminiscente), diferentes tipos de sensores (temperatura, humedad, presión, infrarrojos, etc.), diferentes tipos de chips o circuitos integrados para aplicaciones más completas como un reloj en tiempo real, una memoria EEPROM o el mismo microcontrolador, etc.

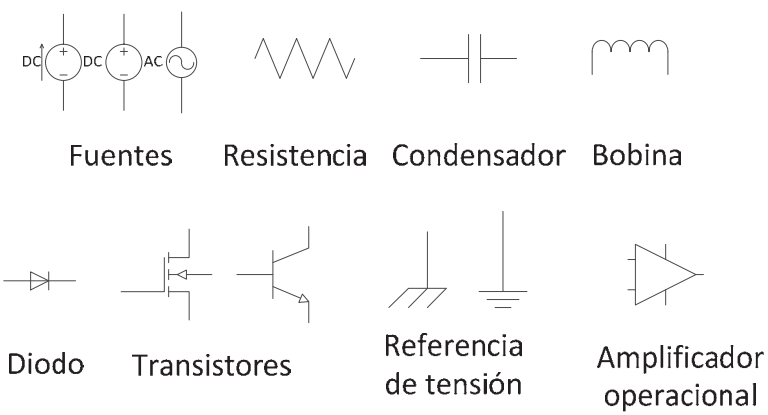

Figura 1.1. Representación esquemática de los dispositivos electrónicos básicos.

En el próximo capítulo se verá el *hardware* en detalle para quien esté interesado en conceptos relacionados con la electrónica. Los siguientes capítulos están enfocados en la programación del *firmware*. Finalmente el último capítulo explica cómo funcionan las herramientas que utiliza Arduino para «traducir» el código y grabarlo en la placa.

1.5 Bibliografía

[1] KUSHNER, DAVID. *The Making of Arduino. How five friends engineered a small circuit board that's taking the DIY world by storm*. IEEE Spectrum, octubre 2011.

[2] ARDUINO. *Arduino website*. [Fecha de consulta: agosto 2013]. Disponible en *http://www.arduino.cc*

[3] WIKIPEDIA CONTRIBUTORS. *Arduino*. [Fecha de consulta: agosto 2013]. Disponible en *http://en.wikipedia.org/wiki/Arduino*

[4] AGUAVIVA, RAÚL. *XOscillo*. [Fecha de consulta: agosto 2013]. Disponible en *https://code.google.com/p/xoscillo/*

[5] HOCHENBAUM, JORDAN; VALLIS, OWEN. *Arduinome*. [Fecha de consulta: agosto 2013]. Disponible en *http://flipmu.com/work/arduinome/*

[6] REPRAP. *Printrbot*. [Fecha de consulta: agosto 2013]. Disponible en *http://reprap.org/wiki/Printrboard*

[7] ANDERSON, CHRIS. *ArduPilot*. [Fecha de consulta: agosto 2013]. Disponible en *http://www.diydrones.com/notes/ArduPilot*

[8] *WAYNE AND LAYNE LLC*. *Video Game Shield*. [Fecha de consulta: agosto 2013]. Disponible en *http://www.wayneandlayne.com/projects/video-game-shield/*

[9] MELLIS, DAVID. *DIY Cellphone*. [Fecha de consulta: agosto 2013]. Disponible en *http://web.media.mit.edu/~mellis/cellphone/index.html*

[10] NANOSATISFI. *ArduSat*. [Fecha de consulta: agosto 2013]. Disponible en *https://ardusat.org/*

2. *Hardware*

En este capítulo se dará una visión general de las placas de Arduino y del microcontrolador. Servirá para presentar los recursos disponibles, los cuales se utilizarán en los capítulos posteriores.

Al final del capítulo se verán las diferentes placas y *shields* Arduino que se pueden utilizar.

2.1 Esquemático de Arduino UNO

Los bloques básicos del *hardware* Arduino vienen dados por la Figura 2.1, en este caso se representa la placa Arduino UNO, la más utilizada y que sirve como referencia para las otras, que se tratarán al final del capítulo.

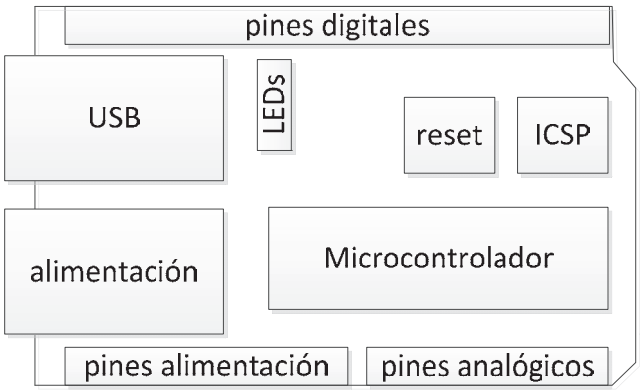

Figura 2.1. Diagrama de bloques que conforman la placa Arduino UNO.

Los bloques principales son: alimentación, microcontrolador, el programador ICSP, USB y los conectores de pines. A continuación se van a tratar cada uno de estos

bloques viendo los componentes que lo forman y su funcionalidad. En el apéndice puede consultarse el esquemático completo.

Cabe destacar que todas las placas vienen con el anagrama Arduino y el nombre de cada una, tal como se puede ver en la Figura 2.2 para la placa Arduino UNO. Esta información es necesaria porque en el entorno de programación hay que indicar explícitamente qué placa se va a programar.

Figura 2.2. Anagrama de la placa Arduino UNO.

Como curiosidad se puede comentar que la serigrafía y su calidad, como indican en el blog de Arduino, es una de las formas de diferenciar las placas oficiales respecto a las clónicas, ya que estas últimas tienen peor calidad y las imágenes se ven menos definidas.

En [1] se pueden descargar los ficheros de diseño del esquema electrónico y del rutado de la placa. Estos ficheros están hechos con el programa EAGLE [2], un programa de diseño electrónico que dispone de versiones gratuitas para usos no comerciales. A partir de ellos se pueden realizar modificaciones o generar los ficheros de fabricación para construir placas propias.

2.2 Alimentación

La alimentación de la placa puede venir de una fuente externa o a través del cable USB. Si no se emplea el USB se utilizará la fuente de la Figura 2.3, en ella se puede observar el esquema electrónico y la parte correspondiente a la placa.

Al conector se enchufaría una tensión DC entre 7 y 12 V. Esta tensión puede provenir de un inversor/transformador.

Este bloque se basa en el regulador NCP1117ST50T3G. Este regulador, U1, se encarga de mantener una tensión estable de 5 V a su salida y puede suministrar hasta 1 A.

También hay dos condensadores electrolíticos de 47 µF y uno de 10 nF. Los condensadores electrolíticos de 47 µF se encargan de estabilizar la tensión y suministrar corriente rápidamente ante demandas instantáneas; el otro condensador de 10 nF sirve para filtrar ruidos de alta frecuencia. El diodo rectificador M7 se utiliza para evitar que la corriente fluya en sentido contrario al deseado. Y finalmente se tiene el conector donde se enchufa la fuente externa.

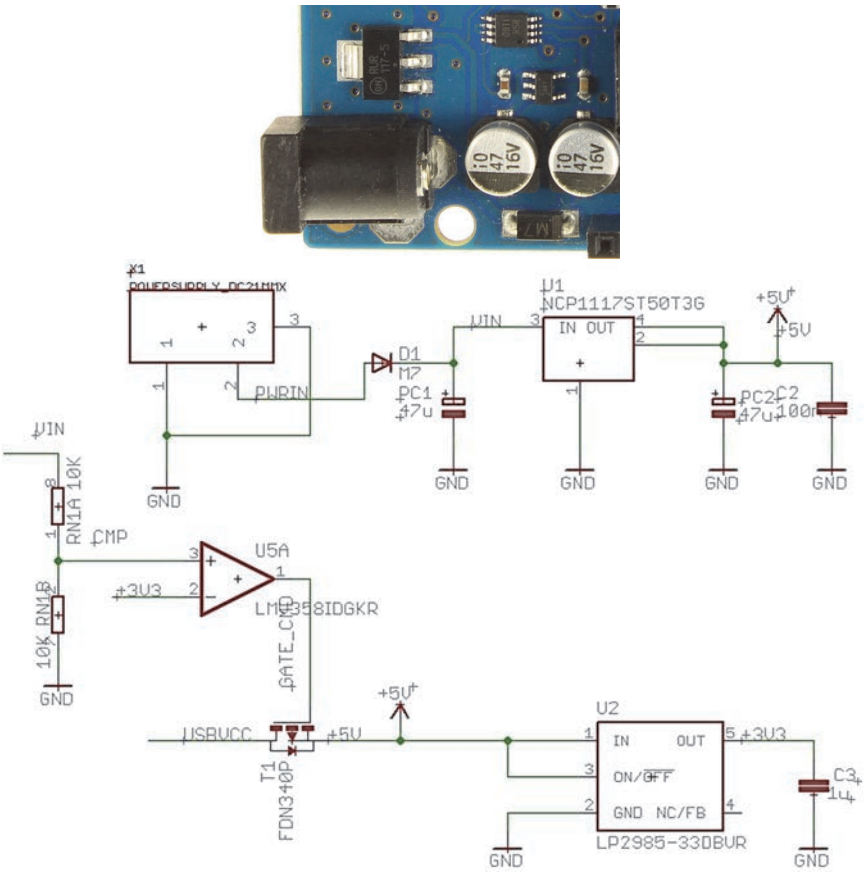

Figura 2.3. Fuente de alimentación de Arduino UNO rev3.

Desde el nodo +5V se utiliza otro regulador, U2, en este caso un LP2985-33DBVR. Su función es conseguir una señal de 3,3 V y hasta 150 mA. Esta tensión de alimentación es utilizada por muchos chips en lugar de la de 5 V. Típicamente se puede utilizar en dispositivos externos y/o *shields*.

El amplificador operacional U5A funciona en modo comparador. Activa su salida si se ha conectado una alimentación externa (concretamente si Vin es superior a 6,6 V). Esta salida sirve para activar o desactivar el transistor T1, que funciona como interruptor. Si hay alimentación externa se utiliza esta y el transistor bloquea la alimentación del USB para que no haya conflictos entre ellas; si no hay alimentación externa el transistor permitirá que la alimentación de la placa provenga del USB.

2.3 Microcontrolador

El microcontrolador usado por Arduino UNO es el ATmega328P-PU [3] de la compañía Atmel. Casi todos los Arduino utilizan microcontroladores de la familia ATmega, la placa UNO usa el modelo 328; las dos últimas letras indican el encapsulado utilizado, en este caso el -PU indica un encapsulado PDIP (de patillas que atraviesan la placa a través de agujeros).

Un microcontrolador es un chip programable, parecido a un microprocesador como el utilizado en los ordenadores personales, pero de mucha menor potencia. A cambio incorpora en un único chip todos los elementos necesarios para funcionar, tales como memoria, reloj y algunos periféricos.

El ATmega es un microcontrolador basado en una arquitectura Harvard modificada de 8 bits con tecnología RISC (*Reduced Instruction Set Computing*), es decir, con memorias separadas para programa y datos y un conjunto de instrucciones simples para realizar la programación.

El microcontrolador combina una memoria Flash interna de 32 kB para almacenar el programa, una memoria EEPROM de 1 kB para los datos no volátiles y 2 kB de memoria SRAM para datos volátiles. Tiene 23 pines de entrada/salida digital, 3 timers/contadores, interrupciones internas y externas, soporta comunicación serie tipo USART, I^2C y SPI, tiene 6 canales de entrada a un convertidor de analógico/digital de 10 bits, tiene un timer *watchdog* con oscilador interno y 5 modos de ahorro de energía. Puede funcionar en un rango de tensión de 1,8 V a 5,5 V y de -40 °C a 85 °C de temperatura.

Del *datasheet* [3] se pueden extraer las características más importantes de este microcontrolador que se resumen en la Tabla 2.1.

Parámetro	Valor
Memoria Flash	32 kB
Memoria SRAM	2 kB
Memoria EEPROM	1024 B
Pines totales	32
Pines I/O	23
Máxima frecuencia de operación	20 MHz
CPU	AVR, 8 bits, RISC
Interrupciones	24
SPI,	2
TWI (I^2C)	1
UART	1
Canales ADC / resolución / velocidad	8 canales / 10 bits / 15 kbps
Comparadores analógicos	1
Timers	3 (2 de 8 bits y 1 de 16 bits)
Tensión de alimentación	1,8 - 5,5 V
Canales PWM	6
RTC	
Watchdog	
Sensor de temperatura	

Tabla 2.1. Características del microcontrolador ATmega328.

Figura 2.4. Microcontrolador ATMega328.

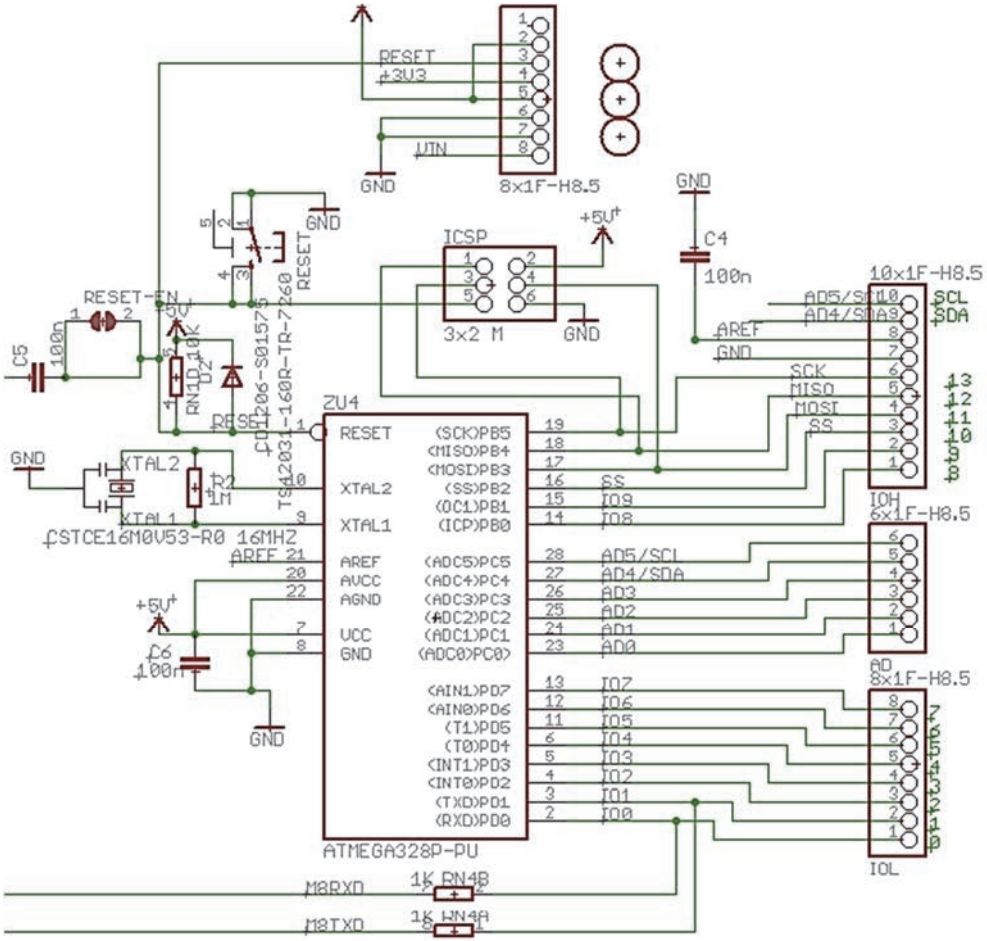

Figura 2.5. Esquema electrónico correspondiente al microcontrolador ATMega328 y sus conexiones.

En el esquema el microcontrolador es el elemento llamado ZU4.

En la Figura 2.5 se aprecia el circuito de *reset* en la parte superior izquierda del microcontrolador. El *reset* reiniciará el programa del microcontrolador, igual que si se le quitara la alimentación y se volviera a conectar. Cuando se presiona el pulsador, en el pin RESET habrá 0 V y el microcontrolador se reseteará. También se producirá un *reset* desde el USB a través del condensador C5. Si no se produce un *reset* por alguna de las dos causas anteriores, la resistencia RN1 fija un valor de 5 V en la línea y permite el funcionamiento normal del microcontrolador.

Debajo de este circuito está el reloj. Este es un resonador de frecuencia de 16 MHz. Otras placas tienen cristales de cuarzo, pero casi todas mantienen esta frecuencia. El reloj le sirve al microcontrolador para marcar el ritmo al cual se van ejecutando las instrucciones del código máquina.

Los elementos rectangulares de la parte derecha y superior son los diferentes conectores.

2.4 Periféricos internos

Dentro del microcontrolador hay varios subsistemas, a los que se denominará periféricos internos.

En la Figura 2.6 se observa el diagrama de bloques interno de un ATmega328, donde se pueden ver diferentes bloques, que se tratarán en los siguientes apartados.

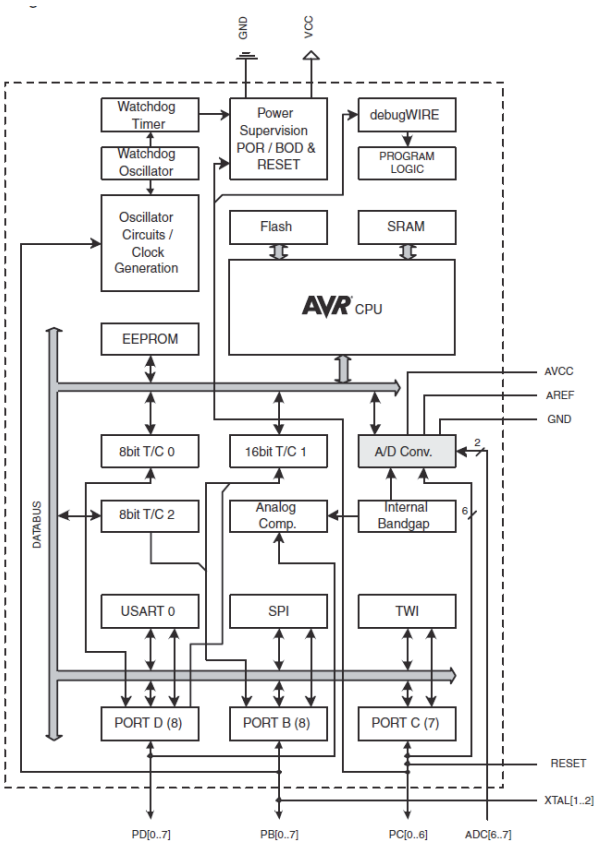

Figura 2.6. Diagrama de bloques del interior de un microcontrolador ATMega328 [3].

2.4.1 Sistema de reloj

Como se acaba de comentar, el reloj determina la velocidad a la que se ejecutan las tareas en el microcontrolador. Con 16 MHz se ejecutará una instrucción máquina en 1/16 M = 62,5 ns.

El microcontrolador tiene diferentes opciones de origen de reloj. Estas son: oscilador de cristal de bajo consumo, oscilador de cristal a pleno funcionamiento, oscilador de cristal a baja frecuencia, oscilador interno RC a 128 kHz, oscilador interno RC calibrado y reloj externo. Estas opciones son seleccionables a través de un registro interno del propio dispositivo que controla el multiplexor de la Figura 2.7 (*Clock Multiplexer*).

De este multiplexor sale la señal de reloj, que pasa a través de un *prescaler*. El *prescaler* se puede utilizar para reducir la frecuencia, reducir el consumo y mejora la estabilidad de la señal de reloj. El factor del *prescaler* puede ir desde 1 a 256 en potencias de 2. En Arduino por defecto está desactivado, por lo que el reloj del núcleo es igual al del resonador externo.

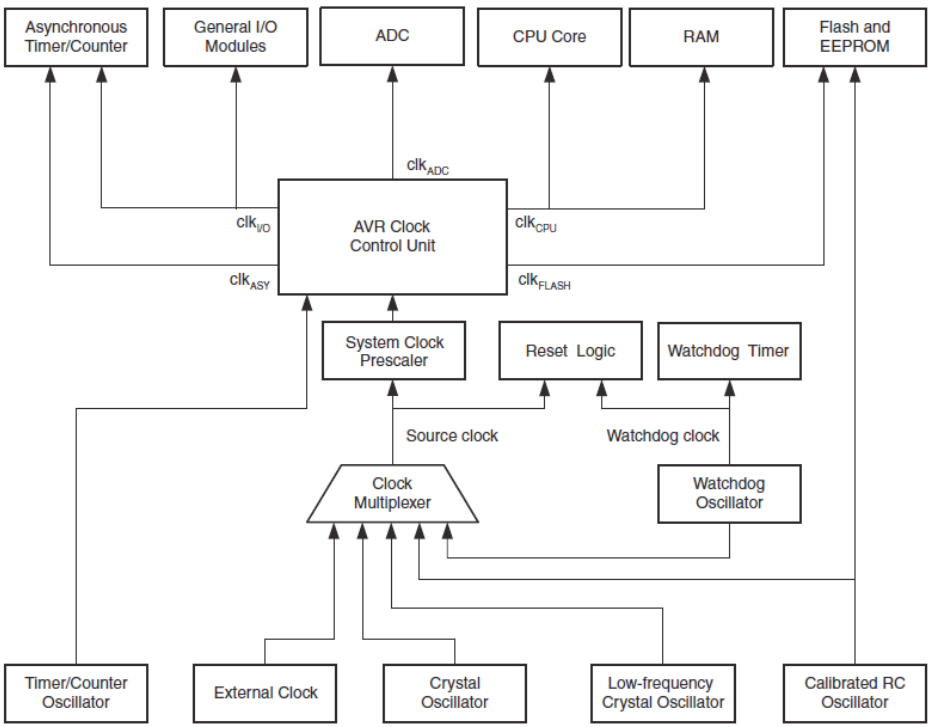

Figura 2.7. Diagrama del sistema de reloj del microcontrolador [3].

La fuente de reloj es distribuida por una unidad de control que gestiona el reloj para los diferentes bloques existentes, como pueda ser por un lado la CPU y la RAM (clk_{CPU}) y los pulsos de reloj que controlan las operaciones de las memorias Flash y EEPROM (clk_{FLASH}), este se suele activar conjuntamente con el clk_{CPU}.

Por otro lado hay una línea de reloj que se usa para la mayoría de los módulos de entrada/salida ($clk_{I/O}$), como puedan ser los contadores/timers, el SPI y la USART.

El conversor analógico/digital tiene una línea de reloj dedicada (clk_{ADC}), al separar esta señal del resto se puede reducir el ruido generado por la circuitería digital y de esta forma tener una conversión ADC (*Analog to Digital Converter*) más exacta.

Finalmente se tiene una línea de reloj para el *timer* asíncrono. Este puede tener una señal de reloj externo con el uso de un cristal de cuarzo de 32 kHz y usar el *timer* como un reloj en tiempo real que sigue funcionando incluso cuando se está en modo dormido (de bajo consumo).

2.4.2 Temporizadores

El microcontrolador que usa la placa UNO, tiene tres *timers* (timer0, timer1 y timer2) que también se pueden usar como contadores. Los *timers* 0 y 2 son de 8 bits y el *timer* 1 de 16. Estos *timers* tienen un módulo de preescalado para su propia señal de reloj. Esta señal de reloj puede venir tanto desde el sistema de reloj ya visto anteriormente (modo *timer*), como desde pines externos (modo contador).

Son módulos que funcionan en paralelo a la CPU y de forma independiente a ella. El funcionamiento básico consiste en aumentar el valor del registro del contador al ritmo que marca su señal de reloj.

Usando el reloj interno o un cristal externo puede ser utilizado para medir tiempos puesto que utiliza una señal periódica, precisa y de frecuencia conocida; mientras que si la señal viene de un pin externo puede contar eventos que se produzcan en el exterior y que se reflejen en cambios de nivel de la tensión de los pines.

Estos contadores también forman parte del generador de señales PWM y permiten configurar tanto la frecuencia como el ciclo de trabajo.

2.4.3 PWM

La modulación por anchura de pulso (PWM) es una técnica de modulación que se basa en la variación de la anchura del pulso de una señal digital en base a una

señal analógica dada. Cuando la señal analógica varía su amplitud, la anchura del pulso de la señal digital cambia.

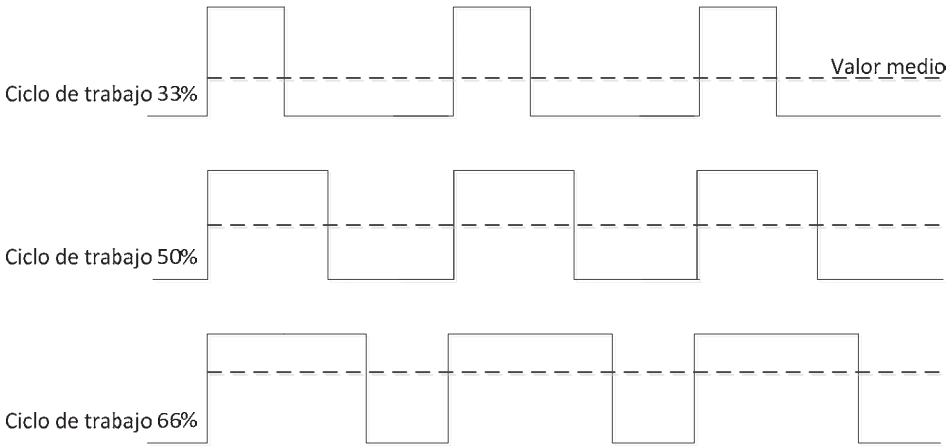

Figura 2.8. Señales PWM con pulsos de distinta anchura.

Habitualmente se usa para el control de motores, regulación de potencia, rectificación y como una forma sencilla de conversión digital-analógica.

La placa UNO soporta el PWM, para ello utiliza los *timers* y permite configurar varios modos de trabajo. La forma de onda PWM, en el modo de trabajo más sencillo (fastPWM), se genera de la siguiente forma:

1. El registro del contador se pone en marcha desde cero y cuenta de forma ascendente. En el momento de empezar la cuenta se activa el pin de salida del PWM.
2. Cuando el valor del registro del contador es igual al valor en otro registro del contador, se conmuta el pin de salida, por lo que pasa a valer 0. De esta forma, el valor de este segundo registro fija el ancho del pulso. El registro del contador sigue contando de forma normal.
3. Cuando el valor del registro del contador llega al final se pasa al primer punto. El pin de salida se vuelve a activar y la cuenta se reinicia. El tiempo que tarda el contador en llegar al final fija el período de la señal.

2.4.4 I^2C

El bus de dos hilos con interfaz serie I^2C (*Inter-Integrated Circuit*) se diseñó específicamente para aplicaciones con microcontroladores. Debido a su extenso

uso, la mayoría de los microcontroladores incluye un *hardware* que se encarga de toda la gestión del bus.

Permite la interconexión de hasta 128 dispositivos diferentes usando solo dos líneas bidireccionales de bus, una para el reloj (SCK) y otra para datos (SDA). Para la implementación del bus únicamente son necesarias dos resistencias de *pull-up*, una para cada línea (algunos chips las pueden tener integradas).

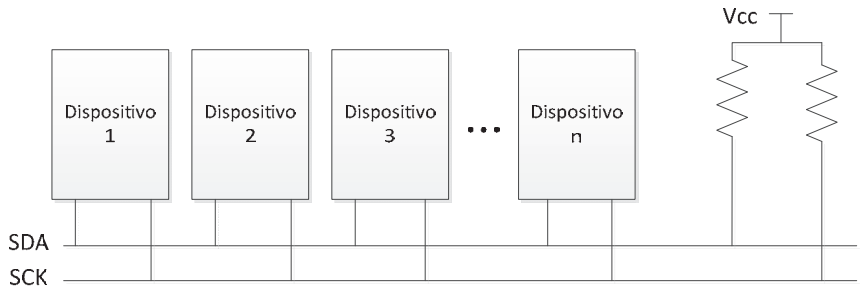

Figura 2.9. Conexión de diferentes dispositivos a través del bus I^2C.

Cada uno de los dispositivos conectados al bus tiene una dirección única, esta dirección se utiliza para seleccionar a un dispositivo en concreto del bus. La comunicación consistirá en que un dispositivo direcciona a otro, con lo que los siguientes eventos del bus solo los atenderá ese dispositivo. Con un dispositivo direccionado se pueden enviar comandos o acceder a sus registros para leerlos o escribirlos.

En la placa UNO los pines son AD5 para el SCK y el AD4 para el SDA.

2.4.5 SPI

Se denomina interfaz serie para periféricos (*Serial Peripheral Interface* – SPI) y consiste en una transmisión de datos síncrona de tres hilos punto a punto. Los microcontroladores normalmente también tienen un *hardware* específico para la gestión de este bus. En la Figura 2.10 se pueden ver las conexiones físicas del SPI. En la placa UNO estos pines son:

- **Pin 10 (SS):** es la línea de selección del esclavo que se va a comunicar con el maestro. Habrá tantos pines SS como número de periféricos

conectados al bus. En lugar —o además— del pin 10 se pueden utilizar otros pines digitales.

- **Pin 11 (MOSI):** es la salida de datos para el maestro y la entrada de datos para el esclavo.
- **Pin 12 (MISO):** es la entrada de datos para el maestro y la salida de datos para el esclavo.
- **Pin 13 (SCK):** es la salida de reloj del maestro y la entrada de reloj para el esclavo o esclavos.

Permite una transferencia de datos síncrona a alta velocidad entre el microcontrolador (como maestro) y dispositivos periféricos (como esclavos), como puedan ser un sensor digital, una memoria externa, otro dispositivo AVR, etc.

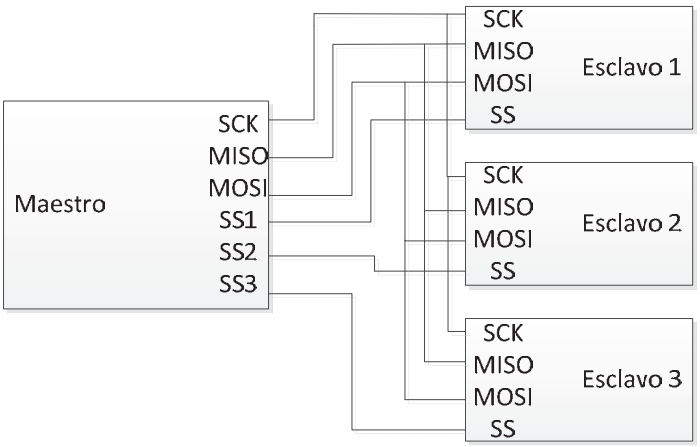

Figura 2.10. Ejemplo de conexión de un maestro y tres esclavos en un bus SPI.

Para seleccionar el dispositivo sobre el que se leerá o escribirá se utilizan las señales SSx. Hay una señal por cada dispositivo, se les suele denominar *chip select*. Los comandos enviados por el bus solo serán reconocidos por el dispositivo seleccionado mediante estas líneas. Al igual que con I^2C, sobre SPI se enviarán comandos para escribir o leer datos en los dispositivos.

2.4.6 USART

Se trata de un puerto de comunicaciones serie muy versátil denominado comúnmente USART (*Universal Synchronous and Asynchronous Serial Receiver and Transmitter*). Este se puede configurar como síncrono o como asíncrono. Los