UNIDAD Nº 1

ESTRUCTURA DE LAS COMPUTADORAS Y LOS MICROCONTROLADORES

2021

I. GENERALIDADES DE UN EQUIPO DE CÓMPUTO Y ARQUITECTURA VON NEUMANN

1.0 INTRODUCCIÓN

Tanto en la ciencia, la tecnología, la ingeniería, el comercio, u otros campos de trabajo del hombre, constantemente se trabaja con **cantidades**. Las cantidades se pueden observar, medir, monitorear o incluso manipular de manera aritmética. Estas cantidades o valores numéricos se pueden representar de dos maneras: de forma **analógica** o de forma **digital**.

La representación analógica se hace cuando un indicador proporcional varía de forma continua, como por ejemplo el velocímetro de un auto que indica por medio de una aguja la velocidad a la que avanza el auto. Los termómetros que tienen encapsulada cierta cantidad de mercurio en una columna sirven para indicar la temperatura del cuerpo donde se coloque empleando la representación analógica. También los relojes antiguos de cuerda que cuentan con una aguja que gira en un eje son un ejemplo de representación de cantidades analógicas. Las cantidades analógicas tienen la característica de variar a través del tiempo un gran intervalo continuo de valores, por ejemplo en el auto que avanza de 0 a 100 km/hr no se puede precisar cuántos intervalos se pueden tener y de qué magnitud. Así también cuando el mercurio recorre la columna de la cápsula de vidrio no se puede precisar cuántos intervalos genera al variar entre un grado de temperatura a otro.

En la representación digital no se emplea un indicador que varía de forma continua sino uno que varía de acuerdo con cambios relacionados a cantidades previamente establecidas en magnitud por medio de símbolos llamados dígitos. Cuando se mide la velocidad de un auto Por medio de un indicador luminoso de leds se está haciendo la representación de la velocidad de forma digital. Un reloj digital moderno puede representar los cambios de tiempo en segundos y la suma de estos en minutos, los que a la vez representan horas y todos ellos los podemos ver a través de leds indicadores en una pantalla. Existen termómetros digitales que también indican a través de dígitos los cambios de temperatura corporal o ambiental de manera precisa por medio de intervalos regulares discretos y no continuos.

La principal diferencia entre cantidades analógicas y digitales se plantea como:

Analógica = Continua (sin valor definido de las cantidades, se da un rango)
Digital = Discreta (paso a paso, valor definido)

En la representación digital no existe ambigüedad cuando se lee el valor de una cantidad digital, mientras que al pretender leer una representación analógica se da pie a la interpretación personal y generalmente se redondea a un nivel de precisión conveniente. Por lo que podemos encontrar sistemas digitales y sistemas analógicos, donde la diferencia entre ellos es el tipo de información que manipulan, ya sea aquella representada de forma analógica o de forma digital. Sin embargo, por las múltiples ventajas económicas, de diseño y manejo de información que ofrece el diseño de sistemas digitales, la tendencia futura es **digital**, de donde surge el uso y la proliferación de los cada vez más sofisticados sistemas de cómputo digitales.

1.1 COMPUTADORA

Existen diversos tipos de sistemas de cómputo, que por la forma como procesan los datos encontramos las digitales, analógicas y las híbridas. El objetivo de este texto se enfoca al estudio de la arquitectura de las computadoras de tipo digital, que manejan datos codificados en un sistema binario con el que se hacen representaciones de cifras, letras o símbolos especiales. Bajo este enfoque definimos computadora digital como:

1.1.1 COMPUTADORA DIGITAL

Es una máquina electrónica de propósito general compuesta por varios elementos con una función específica, controlados por un núcleo central llamado microprocesador capaz de procesar grandes cantidades de datos a gran velocidad bajo la dirección de un programa almacenado en su memoria.

1.1.2 COMPUTADORA ANALÓGICA

Computadora analógica: esta procesa datos que están registrados en una escala continua, medidos bajo cierto grado de precisión. Se utilizan para reconstruir modelos que representan un sistema físico real, operando generalmente al mismo tiempo en el que se estudia dicho fenómeno o sea en tiempo real, creando con la información una analogía matemática que permite construir simuladores que reproducen el fenómeno físico estudiado con una alta fidelidad. Así se construyen modelos en la industria del petróleo, simuladores de redes eléctricas, simuladores de vuelo para la aviación, que consideran variables como la velocidad del aire, elevación, presión atmosférica, ángulo de ataque, etc. Y otras más donde es importante representar la variación en el tiempo de magnitudes continuas. Es una desventaja que solo sirvan para una aplicación, pero son muy importantes porque pueden establecer modelos de tamaño menor, manejable, con sus variables de manera proporcional al fenómeno real, lo que implica disminuciones de costo en el desarrollo de proyectos.

1.1.3 COMPUTADORA HÍBRIDA

Las computadoras híbridas son aquellas que utilizan simultáneamente los sistemas de información adquiridos de manera analógica y digital en sus componentes. Para usar ambos tipos de técnicas de información en la solución de un problema, se emplean convertidores "digital-analógico" y "analógico-digital", para hacer la conversión adecuada del tipo de información que emplea una computadora digital y viceversa. Una aplicación la podemos ver en los sistemas GPS.

1.2 GENERACIONES DE COMPUTADORAS DIGITALES

En 1944 John Von Neumann propuso ideas para la construcción de una computadora electrónica digital, basado en un concepto llamado de programa interno almacenado. Con estas ideas en 1946 se construyó la primera computadora electrónica digital, cuya lógica de operación se basó en circuitos combinatorios construidos con tubos de vacío, almace-nando datos en dispositivos llamados tambores magnéticos. El sistema construido con aproximadamente 18000 bulbos resultó ser un verdadero monstruo por su gran tamaño. Se programó en lenguaje de máquina y recibió el nombre de ENIAC (Electronic Numerical Integrator and Calculator). Con la misma tecnología en 1950 se contruyó la UNIVAC 1.

En 1948 se empezó a experimentar y hacer las primeras aplicaciones con el transistor. Hacia 1955 se empezó a idear cómo sustituir los circuitos lógicos discretos, construidos con bulbos de vacío de las computadoras, con nuevos construidos con el transistor, mientras que las memorias construidas con anillos de ferrita se empezaban a emplear con éxito. El desarrollo de la ingeniería electrónica es lo que ha permitido marcar con claridad la evolución de las computadoras digitales, por lo que se establecen las siguientes:

1.2.1 COMPUTADORAS DE PRIMERA GENERACIÓN (1954-1959)*

*(Las fechas indicadas de cada período generacional pueden variar de acuerdo con el autor, las indicadas en el texto, fueron tomadas de la Ing. Patricia Quiroga. Arquitectura de computadoras. ED. Alfaomega, 2010).

Las computadoras de esta generación se caracterizan porque están construidas con válvulas de vacío, también llamadas bulbos. Ocupaban un gran espacio y disipaban una gran cantidad de calor, por lo que consumían gran cantidad de energía eléctrica, además se requerían de sistemas de aire acondicionado. Los trabajos se ejecutaban de manera secuencial, o sea: se ejecutaban los programas instrucción por instrucción.

Los programas se almacenaban en tarjetas o cintas perforadas y se cargaban a la memoria principal por otro programa llamado cargador del sistema operativo. Los resultados se imprimían. Los procesos de entrada, proceso y salida de información se hacían de manera secuencial, lo que acumulaba tiempo al estar encadenado su desarrollo.

1.2.2 SEGUNDA GENERACIÓN (1959-1964)

Construidas a base de transistores y circuitos impresos, lo que provocó una gran disminución de su tamaño, hasta en 200 veces en relación la generación anterior, con el consecuente ahorro de energía eléctrica. La memoria principal se formaba con núcleos magnéticos. Se Posibilitó la ejecución de una operación de cálculo con una de E/S, empleando para ello computadoras auxiliares para el almacenamiento en unidades de cinta magnética como un medio intermedio a las perforadoras de tarjeta e impresoras, dando origen al procesamiento por lotes.

Se empezaron a usar lenguajes de programación de alto nivel como Fortran y Cobol, con lo que se incrementan el número de aplicaciones: científicas, comerciales, simulación, inventarios, nóminas, contables, reservación en líneas aéreas, control de tráfico aéreo.

1.2.3 TERCERA GENERACIÓN (1964-1971)

Entre 1964 y 1967 se tuvo éxito al empezar a sustituir los circuitos transistorizados discretos por aquellos integrados en los equipos de cómputo, que son pastillas de silicio en cuyo interior se concentran miles de componentes electrónicos con técnicas de fabricación SSI (Smal Scale Integration) y MSI (Medium Scale Integratión), con lo que se incrementó considerablemente la velocidad de ejecución y la disminución de gasto de energía eléctrica.

Surge el concepto de **multiprogramación** y **multiproceso**. Varios programas residen en forma simultánea en la memoria en estado de ejecución y varias computadoras relacionadas entre sí en ejecución de varios programas de forma simultánea. También se hace posible el teleproceso, que es el procesamiento a distancia.

Tiempo compartido o **time sharing** es la forma en que una computadora asigna de forma alterna una parte del tiempo del CPU a cada proceso que ejecuta, lo que aparenta simultaneidad de ejecución.

Surgen los sistemas operativos que permiten la interactividad del usuario con los procesos que ejecuta el equipo de cómputo y la expansión de la industria del software con la capacidad de poder operar en máquinas futuras.

Un equipo da atención a diversos usuarios, lo que la hace **multiusuario**, y al ser multiusuario por lo tanto también se le clasifica como **multitárea**.

En esta etapa se desarrolla el concepto de **máquina virtual**, que simplifica la labor del programador, ya que solo interactúa con un equipo ficticio. Es creada y controlada por el sistema operativo que genera una actividad compartida por varios usuarios.

1.2.4 CUARTA GENERACIÓN (1971-1981)

Bajo el enfoque de equipos de cómputo digital, que en lo sucesivo llamaremos simplemente computadora o PC (computadora personal), encontramos a su vez diversas clasificaciones, de acuerdo con su tamaño, precio, potencia, aplicación, tecnología y otros aspectos de su implementación o uso.

La fabricación de circuitos integrados con tecnologías LSI o Large Scale Integration (gran escala de integración) permitió incluir en un solo circuito integrado una CPU completa, dándosele el nombre de microprocesador. Así se lograron obtener equipos más pequeños, accesibles económicamente, más rápidos. Un usuario podía comprarse un equipo de cómputo y llevárselo a su casa, naciendo el concepto de PC (computadora personal). Por lo que la atención se encauzó a mejorar el software para que permitiera mayores velocidades de proceso pensando que los avances de hardware se habían agotado.

En esta etapa los procesos se realizan en mayor medida en tiempo real. El proceso interactivo hace posible la consulta y actualización de datos de grandes bancos de información, teniendo para ello unidades distribuidas en un sistema de red, como el sistema bancario, consultas telefónicas con respuesta inmediata, regulación automática de semáforos, procesos en línea de producción manufacturera, paquetes integrados de software con herramientas poderosas de hoja de cálculo y procesadores de texto, de diseño CAD (asistido por computadora), desarrollo de sistemas operativos con interfaces gráficas, etc.

1.2.5 QUINTA GENERACIÓN Y LA INTELIGENCIA ARTIFICIAL (1982-1989)

La arquitectura de computadoras se encausa a nuevos conceptos de diseño, arquitecturas RISC, pipelines, superescalaridad, niveles de caché más densos, etc. Las supercomputadoras desarrollan funciones inteligentes basados en experiencias de inteligencia artificial encauzando proyectos al paradigma que propone que es necesario diseñar equipos con funciones inteligentes capaces de aprender.

Un avance tecnológico que sirve como parámetro de inicio a esta 5ª generación es la creación de la primer supercomputadora con capacidad de proceso en paralelo, diseñada por Seymoury Cray en 1982. Y otro hecho significativo en esta generación es el proyecto "Quinta Generación" por parte del gobierno japonés con la participación de las seis empresas más destacadas de cómputo y que debería terminar en 1992.

Los procesadores ejecutan varias etapas de las instrucciones en paralelo como ejemplo de avance con respecto a las generaciones anteriores, cuyo proceso era estrictamente instrucción por instrucción. Los objetivos tienden incluso el reconocimiento de un lenguaje natural con voz, la lectura de información escrita, en definitivo al reconocimiento de cualquier forma de comunicación humana a fin de ser capaces con el aprendizaje, la toma de decisiones similares a las del ser humano.

Se reconocen tres áreas fundamentales de desarrollo: integración de circuitos a una cada vez más alta escala de integración, nuevos métodos de procesamiento de información, nuevas arquitecturas y avances en el diseño de memorias de mayor capacidad y velocidad. Un equipo de quinta generación puede estar dividido en niveles de funcionalidad: una superior que se encarga de la interfaz hombre máquina en lenguaje natural, un procesador de conocimientos implementado en una máquina virtual con base de datos y red de comunicaciones, una configuración de procesadores dedicados, un procesador vectorial, uno de comunicaciones, uno de base de datos y uno de sistema operativo. De manera que las computadoras participan activamente en su propia evolución; los diseños de equipos de alta gama de rendimiento requieren cada vez más de su propia inteligencia.

El objetivo es que las computadoras tengan capacidades de aprendizaje superiores a las del ser humano, por lo tanto deben contar con bases de datos enormes con la información cultural, científica y tecnológica que posee el hombre, acumulado en los dispositivos de memoria masiva del equipo en cuestión contando además con nuevas tecnologías de proceso, almacenamiento y recuperación de información como lo hace la memoria humana a corto y largo plazo.

Independientemente de los "milagros" de la tecnología moderna, es difícil distinguir la brecha que separa la quinta y la sexta generación. El proyecto japonés parece detenido y momentáneamente sumido en el fracaso, quizás momentáneo, de la inteligencia artificial. El pronóstico hecho en esta generación de enlazar computadoras entre sí, sigue en auge desde 1994, con la expansión de la red Internet y del World Wide Web, y ha adquirido una importancia enorme en empresas grandes, medianas y pequeñas, y desde luego, entre los usuarios comunes de los equipos de cómputo.

1.2.6 SEXTA GENERACIÓN (1990 HASTA LA FECHA)

La sexta generación de computadoras, para la cual, como para las otras generaciones, difícilmente se puede establecer un punto de partida, sin embargo se considera que el inicio de esta es en el año de 1990, y la caracterizan arquitecturas combinadas paralelo/ vectorial con miles de procesadores vectoriales trabajando al mismo tiempo, con altas velocidades de operación, del orden de los petaflops (cuatrillones de operaciones de punto flotante por segundo); las redes WAN de área mundial (Wide Area Network) siguen creciendo de manera desorbitada utilizando medios de comunicación a través de fibra óptica, satelital, medios inalámbricos 3G, 4G, con anchos de banda impresionantes. Las tecnologías de esta generación siguen desarrollándose con un abaratamiento de costos, como es natural cuando surgen empresas competentes. Tecnologías basadas en inteligencia/ artificial distribuida; redes neuronales, teoría del caos, sistemas difusos, algoritmos genéticos, holografía, transistores ópticos, procesadores construidos con grafeno, computación cuántica, computación basada en ADN, etcétera. Microprocesadores con memoria caché interna de los tres niveles L 1, L2, L3, con tamaños mayores a decenas de megabytes. El acceso a Internet de todos los equipos de cómputo existentes en cada hogar, en cada escuela. Una computadora es capaz de jugar el ajedrez y ganarle a Gary Kasparov, campeón mundial en esta disciplina. Se inician los torneos de competencia en diversas disciplinas individuales y por equipos (de lucha, fútbol, etc.) con robots fabricados en diferentes países y, desde luego, las nuevas guerras con aviones no tripulados.

1.5 ARQUITECTURA VON NEUMANN

John Von Neumann (28 de diciembre 1903-8 de febrero de 1957). Nacido en Budapest (Hungría). Matemático, nacionalizado estadounidense, genio matemático desde niño, alumno de Albert Einstein, doctorado en Matemáticas en la Universidad de Budapest, también graduado en Ingeniería Química, desarrolló las matemáticas estadísticas a partir de juegos de estrategia desarrollada después en su obra *Theory of games and economic behavior*, publicada en 1944.

Participó en el proyecto Manhattan estadounidense en 1943 para la construcción de bombas atómicas, posteriormente también en la bomba de hidrógeno y en el desarrollo de misiles balísticos.

Su contribución a las computadoras en el ejército fue la concepción de una memoria que actúa secuencialmente, almacenando un programa con instrucciones para la solución de un problema y no solo registrando los datos numéricos del problema.

Interesado en la robótica, en 1952 propuso dos modelos de máquinas autoreproductoras, una con reproducción parecida a los cristales y otra con similitud a la de los animales. Nombrado miembro en 1955 en la Comisión de Energía Atómica del gobierno estadounidense, padece un cáncer que lo retira de la actividad científica hasta su muerte.

John Von Neumann observó que manejar operaciones aritméticas en sistema decimal en las computadoras como la ENIAC generaba muchos problemas, por lo que ideó la forma de emplear aritmética binaria. Diseño un sistema básico de cómputo, al que se le denominó Máquina de Von Neumann y se empleó para construir el siguiente modelo de computadora, la EDVAC, que fue la primera que podía almacenar un programa.

El modelo Von Neumann lo integraban los siguientes módulos principales:

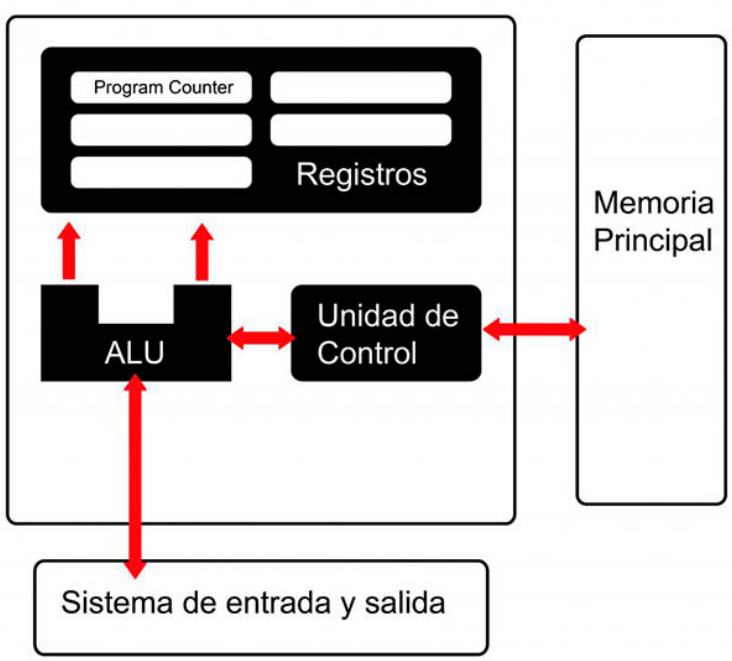


Figura 1.3 Módulos básicos de la Arquitectura Von Neumann

Memoria principal. Formada por 4096 palabras cada una de 40 bits. Donde en cada palabra se podían almacenar 2 instrucciones de 20 bits o números enteros de 39 bits cada uno con signo, de manera temporal. Se establecen diferencias entre "memoria permanente" y "memoria de trabajo", con lo que se concibe lo que será hasta la actualidad la memoria RAM, destinada al almacenamiento temporal de datos e instrucciones.

Unidad de Control. Es la unidad que supervisa, gestiona y concentra el sistema completo, encargada de controlar que la transferencia de información se haga de manera correcta e indicarle a la Unidad Aritmética Lógica las operaciones a realizar. Una parte importante en este módulo es el puntero de instrucción que guarda la dirección de la próxima instrucción a leer y ejecutar. Junto con la Unidad Aritmética conformarán en la actualidad la unidad central de proceso o microprocesador.

Unidad Aritmética. Es aquella unidad responsable de ejecutar las operaciones aritméticas (matemáticas) y lógicas (de comparación) necesarias para poder completar la ejecución de una instrucción. Los datos los recibe de la memoria principal, donde también almacenará los resultados. A modo de memoria dispone de registros internos donde también almacena datos.

Unidad de Entrada/Salida. Es el sistema de comunicación del usuario, donde se puede interactuar y permite recibir y enviar datos al exterior por medio de los dispositivos periféricos que se encuentran conectados al sistema a través de los buses.

Periférico de Entrada. Se le llamó de esta manera a todo dispositivo que permitiera alimentar de información al sistema básico de cómputo.

Periférico de Salida. Así se le llamó desde entonces a cualquier dispositivo que permitiera dar a conocer la información procesada por el equipo de cómputo a un usuario.

Este modelo fue básicamente para ejecutar un programa, es decir conjuntos de instrucciones elementales establecidos en un orden preestablecido. A través del puntero contador de programa incluido en la unidad de control, que siempre apunta a la dirección de la siguiente instrucción a ejecutar, se toma dicha instrucción, se analiza y si es preciso se leen de la memoria los operandos necesarios. El siguiente paso es la ejecución de la instrucción registrada, que puede ser de menor o mayor complejidad, guardándose los resultados obtenidos en los registros o incluso en la memoria principal del sistema, finalmente se genera la iteración, o sea que se regresa al apuntador de programa para tomar la siguiente dirección de memoria, donde se encuentra almacenada la siguiente instrucción del programa que se está ejecutando, repitiendo nuevamente el proceso, al que se le de-nomina ciclo de FETCH.

Este tipo de arquitectura, como podremos, ver fue base de desarrollo para las futuras computadoras hasta la actualidad, aunque actualmente han surgido nuevos conceptos de arquitectura por la forma como se ejecutan las instrucciones con más velocidad, como la RISC o EPIC.

Las primeras computadoras que operaban con sistema numérico decimal eran construidas con una circuitería electrónica muy complicada y susceptible a fallas con sistemas cableados para la programación, por lo que Von Neumann propuso dos conceptos básicos que revolucionarían la incipiente informática:

a) El uso de un sistema numérico binario, que reducía de manera considerable los problemas y fallas que se generaban durante la ejecución de operaciones y funciones lógicas.

b) Almacenamiento del programa con la secuencia de instrucciones en una memoria interna, accesible y con los datos que se empleaban, incrementándose con esto la velocidad de ejecución , ya que anteriormente tanto las instrucciones como los datos se introducían de manera alambrada en un circuito impreso en el mejor de los casos.

Por lo tanto los modelos de computadora que aparecieron posteriormente, considerando las anteriores características, tienen una estructura básica compuesta por las siguientes partes:

1.6 ARQUITECTURA BÁSICA DE UNA COMPUTADORA

En lo sucesivo haremos una revisión de los componentes de una computadora personal ampliando el análisis en algunos aspectos donde se hace más destacado el concepto de arquitectura, lo que ha hecho evolucionar un sistema de cómputo hasta nuestros días.

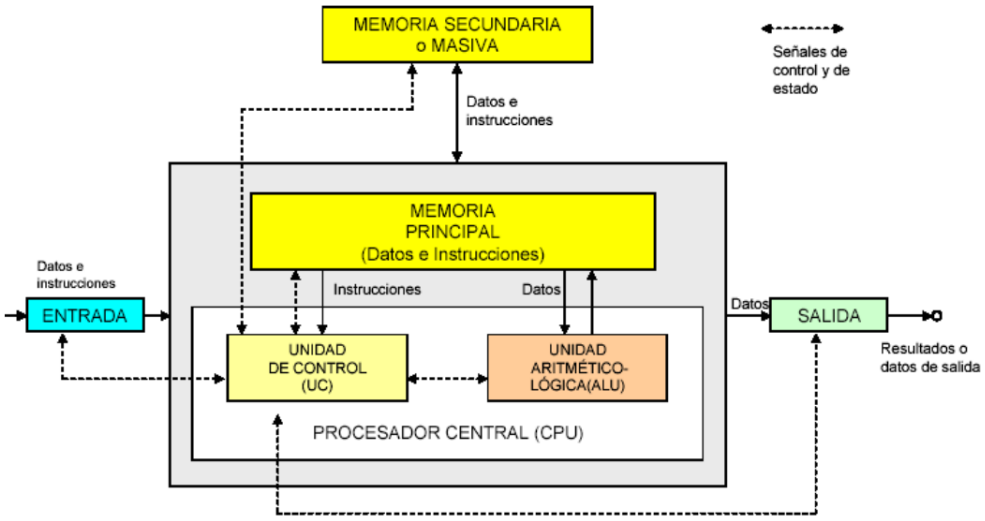
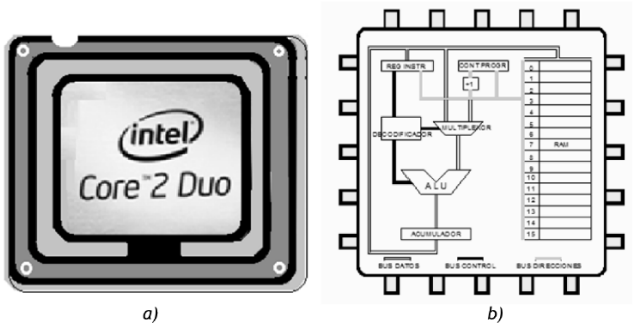


Figura 1.4 Arquitectura básica de una computadora

1.6.1 UNIDAD CENTRAL DE PROCESO (CPU)

Es la Unidad Central de Proceso de un sistema de cómputo, contiene aquellos componentes (circuitos y chips de memoria en su mayoría) que interpretan, controlan y ejecutan instrucciones. Es el componente del computador y otros equipos con dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos.



a) *b)*

Figura 1.5 *a)*.Vista externa de un CPU, *b)* Vista interna de un CPU o microprocesador básico

El CPU es un microprocesador fabricado en un chip, un único trozo de silicio que contiene millones de componentes electrónicos. El microprocesador del CPU está formado por una unidad aritmético-lógica (ALU) que realiza cálculos, comparaciones y toma decisiones lógicas (determina si una afirmación es cierta o falsa) por una serie de registros donde se almacena información temporalmente, y por una unidad de control que interpreta y ejecuta las instrucciones. Para aceptar órdenes del usuario, acceder a los datos y presentar los resultados, el CPU se comunica a través de un conjunto de circuitos o conexiones llamado bus. El bus conecta el CPU a los dispositivos de almacenamiento, los dispositivos de entrada y los dispositivos de salida.

1.6.2 UNIDAD DE CONTROL

Coordina las operaciones del CPU, decodificando el contenido del registro de instrucción y, mediante pulsos de reloj, coordina las acciones necesarias para ejecutar de manera secuencial un proceso lógico. Es decir dirige de manera secuencial los elementos lógicos de la ALU para que ejecute instrucciones que sabe hacer el microprocesador tales como: leer y escribir datos, ya sea en sus registros de memoria internos, a la ALU o a dispositivos externos también llamados periféricos de entrada y salida (E/S) intercambiando información a través del bus de datos.

1.6.3 UNIDAD ARITMÉTICA-LÓGICA

Compuesta por varios registros cuya función es realizar las decisiones de la lógica de Boole, suma binaria, multiplicación y división en múltiplos de dos, complementar una palabra. También se le conoce como ALU, por sus siglas en inglés, Arithmetic Logic Unit. Es la parte matemática del CPU.

HARDWARE ARDUINO

¿QUÉ ES UN SISTEMA ELECTRÓNICO?

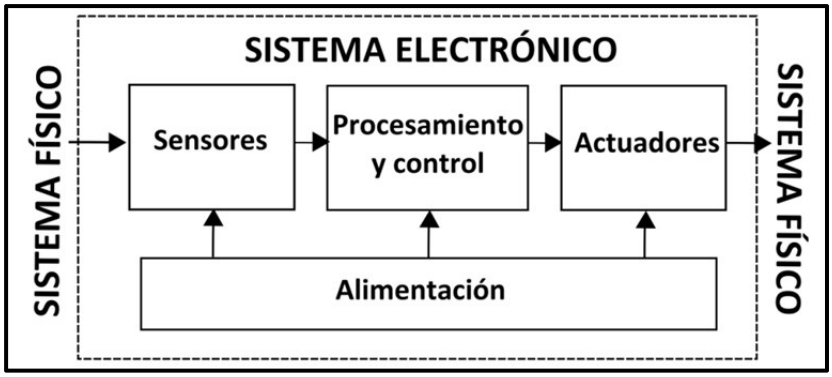
Un sistema electrónico es un conjunto de: sensores, circuitería de procesamiento y control, actuadores y fuente de alimentación.

Los sensores obtienen información del mundo físico externo y la transforman en una señal eléctrica que puede ser manipulada por la circuitería interna de control. Existen sensores de todo tipo: de temperatura, de humedad, de movimiento, de sonido (micrófonos), etc.

Los circuitos internos de un sistema electrónico procesan la señal eléctrica convenientemente. La manipulación de dicha señal dependerá tanto del diseño de los diferentes componentes hardware del sistema, como del conjunto lógico de instrucciones (es decir, del "programa") que dicho hardware tenga pregrabado y que sea capaz de ejecutar de forma autónoma.

Los actuadores transforman la señal eléctrica acabada de procesar por la circuitería interna en energía que actúa directamente sobre el mundo físico externo. Ejemplos de actuadores son: un motor (energía mecánica), una bombilla (energía lumínica), un altavoz (energía acústica), etc.

La fuente de alimentación proporciona la energía necesaria para que se pueda realizar todo el proceso descrito de "obtención de información del medio <-> procesamiento <-> actuación sobre el medio". Ejemplos de fuentes son las pilas, baterías, adaptadores AC/DC, etc.



¿QUÉ ES UN MICROCONTROLADOR?

Un microcontrolador es un circuito integrado o "chip" (es decir, un dispositivo electrónico que integra en un solo encapsulado un gran número de componentes) que tiene la característica de ser programable. Es decir, que es capaz de ejecutar de forma autónoma una serie de instrucciones previamente definidas por nosotros. En el diagrama anterior, representativo de un sistema electrónico, el microcontrolador sería el componente principal de la circuitería de procesamiento y control.

Por definición, un microcontrolador (también llamado comúnmente “micro”) ha de incluir en su interior tres elementos básicos:

CPU (Unidad Central de Proceso): es la parte encargada de ejecutar cada instrucción y de controlar que dicha ejecución se realice correctamente. Normalmente, estas instrucciones hacen uso de datos disponibles previamente (los “datos de entrada”), y generan como resultado otros datos diferentes (los “datos de salida”), que podrán ser utilizados (o no) por la siguiente instrucción.

Diferentes tipos de memorias: son en general las encargadas de alojar tanto las instrucciones como los diferentes datos que estas necesitan. De esta manera posibilitan que toda esta información (instrucciones y datos) esté siempre disponible para que la CPU pueda acceder y trabajar con ella en cualquier momento. Generalmente encontraremos dos tipos de memorias: las que su contenido se almacena de forma permanente incluso tras cortes de alimentación eléctrica (llamadas “persistentes”), y las que su contenido se pierde al dejar de recibir alimentación (llamadas “volátiles”). Según las características de la información a guardar, esta se grabará en un tipo u otro de memoria de forma automática, habitualmente.

Diferentes patillas de E/S (entrada/salida): son las encargadas de comunicar el microcontrolador con el exterior. En las patillas de entrada del microcontrolador podremos conectar sensores para que este pueda recibir datos provenientes de su entorno, y en sus patillas de salida podremos conectar actuadores para que el microcontrolador pueda enviarles órdenes e así interactuar con el medio físico. De todas formas, muchas patillas de la mayoría de microcontroladores no son exclusivamente de entrada o de salida, sino que pueden ser utilizados indistintamente para ambos propósitos (de ahí el nombre de E/S).

Es decir, un microcontrolador es un computador completo (aunque con prestaciones limitadas) en un solo chip, el cual está especializado en ejecutar constantemente un conjunto de instrucciones predefinidas. Estas instrucciones irán teniendo en cuenta en cada momento la información obtenida y enviada por las patillas de E/S y reaccionarán en consecuencia. Lógicamente, las instrucciones serán diferentes según el uso que se le quiera dar al microcontrolador, y deberemos de decidir nosotros cuáles son.

Cada vez existen más productos domésticos que incorporan algún tipo de microcontrolador con el fin de aumentar sustancialmente sus prestaciones, reducir su tamaño y coste, mejorar su fiabilidad y disminuir el consumo. Así, podemos encontrar microcontroladores dentro de multitud de dispositivos electrónicos que usamos en nuestra vida diaria, como pueden ser desde un simple timbre hasta un completo robot pasando por juguetes, frigoríficos, televisores, lavadoras, microondas, impresoras, el sistema de arranque de nuestro coche, etc.

¿QUÉ ES ARDUINO?

Arduino (http://www.arduino.cc) es en realidad tres cosas:

Una placa hardware libre que incorpora un microcontrolador reprogramable y una serie de pines-hembra (los cuales están unidos internamente a las patillas de E/S del microcontrolador) que permiten conectar allí de forma muy sencilla y cómoda diferentes sensores y actuadores.

Cuando hablamos de "placa hardware" nos estamos refiriendo en concreto a una PCB (del inglés "printed circuit board", o sea, placa de circuito impreso). Las PCBs son superficies fabricadas de un material no conductor (normalmente resinas de fibra de vidrio reforzada, cerámica o plástico) sobre las cuales aparecen laminadas ("pegadas") pistas de material conductor (normalmente cobre). Las PCBs se utilizan para conectar eléctricamente, a través de los caminos conductores, diferentes componentes electrónicos soldados a ella. Una PCB es la forma más compacta y estable de construir un circuito electrónico (en contraposición a una breadboard, perfboard o similar) pero, al contrario que estas, una vez fabricada, su diseño es bastante difícil de modificar. Así pues, la placa Arduino no es más que una PCB que implementa un determinado diseño de circuitería interna.

No obstante, cuando hablamos de "placa Arduino", deberíamos especificar el modelo concreto, ya que existen varias placas Arduino oficiales, cada una con diferentes características (como el tamaño físico, el número de pines-hembra ofrecidos, el modelo de microcontrolador incorporado –y como consecuencia, entre otras cosas, la cantidad de memoria utilizable–, etc.). Conviene conocer estas características para identificar qué placa Arduino es la que nos convendrá más en cada proyecto.

De todas formas, aunque puedan ser modelos específicos diferentes (tal como acabamos de comentar), los microcontroladores incorporados en las diferentes placas Arduino pertenecen todos a la misma "familia tecnológica", por lo que su funcionamiento en realidad es bastante parecido entre sí. En concreto, todos los microcontroladores son de tipo AVR, una arquitectura de microcontroladores desarrollada y fabricada por la marca Atmel (http://www.atmel.com). Es por eso que, en este libro seguiremos nombrando "placa Arduino" a cualquiera de ellas mientras no sea imprescindible hacer algún tipo de distinción.

El diseño hardware de la placa Arduino está inspirado originalmente en el de otra placa de hardware libre preexistente, la placa Wiring (http://www.wiring.co). Esta placa surgió en 2003 como proyecto personal de Hernando Barragán, estudiante por aquel entonces del Instituto de Diseño de Ivrea (lugar donde surgió en 2005 precisamente la placa Arduino).

Un software (más en concreto, un "entorno de desarrollo") **gratis, libre y multiplataforma** (ya que funciona en Linux, MacOS y Windows) que debemos instalar en nuestro ordenador y que nos permite escribir, verificar y guardar ("cargar") en la memoria del microcontrolador de la placa Arduino el conjunto de instrucciones que deseamos que este empiece a ejecutar. Es decir: nos permite programarlo. La manera estándar de conectar nuestro computador con la placa Arduino para poder enviarle y grabarle dichas instrucciones es mediante un simple cable USB, gracias a que la mayoría de placas Arduino incorporan un conector de este tipo.

Los proyectos Arduino pueden ser autónomos o no. En el primer caso, una vez programado su microcontrolador, la placa no necesita estar conectada a ningún computador y puede funcionar autónomamente si dispone de alguna fuente de alimentación. En el segundo caso, la placa debe estar conectada de alguna forma permanente (por cable USB, por cable de red Ethernet, etc.) a un computador ejecutando algún software específico que permita la comunicación entre este y la placa y el intercambio de datos entre ambos dispositivos. Este software específico lo deberemos programar generalmente nosotros mismos mediante algún lenguaje de programación estándar como Python, C, Java, Php, etc., y será independiente completamente del entorno de desarrollo Arduino, el cual no se necesitará más, una vez que la placa ya haya sido programada y esté en funcionamiento.

Un lenguaje de programación libre. Por "lenguaje de programación" se entiende cualquier idioma artificial diseñado para expresar instrucciones (siguiendo unas determinadas reglas sintácticas) que pueden ser llevadas a cabo por máquinas. Concretamente dentro del lenguaje Arduino, encontramos elementos parecidos a muchos otros lenguajes de programación existentes (como los bloques condicionales, los bloques repetitivos, las variables, etc.), así como también diferentes comandos –asimismo llamados "órdenes" o "funciones" – que nos permiten especificar de una forma coherente y sin errores las instrucciones exactas que queremos programar en el microcontrolador de la placa. Estos comandos los escribimos mediante el entorno de desarrollo Arduino.

Tanto el entorno de desarrollo como el lenguaje de programación Arduino están inspirado en otro entorno y lenguaje libre preexistente: Processing (http://www.processing.org), desarrollado inicialmente por Ben Fry y Casey Reas. Que el software Arduino se parezca tanto a Processing no es casualidad, ya que este está especializado en facilitar la generación de imágenes en tiempo real, de animaciones y de interacciones visuales, por lo que muchos profesores del Instituto de Diseño de Ivrea lo utilizaban en sus clases. Como fue en ese centro donde precisamente se inventó Arduino es natural que ambos entornos y lenguajes guarden bastante similitud. No obstante, hay que aclarar que el lenguaje Processing está construido internamente con código escrito en lenguaje Java, mientras que el lenguaje Arduino se basa internamente en código C/C++.

Con Arduino se pueden realizar multitud de proyectos de rango muy variado: desde robótica hasta domótica, pasando por monitorización de sensores ambientales, sistemas de navegación, telemática, etc. Realmente, las posibilidades de esta plataforma para el desarrollo de productos electrónicos son prácticamente infinitas y tan solo están limitadas por nuestra imaginación.

¿CUÁL ES EL ORIGEN DE ARDUINO?

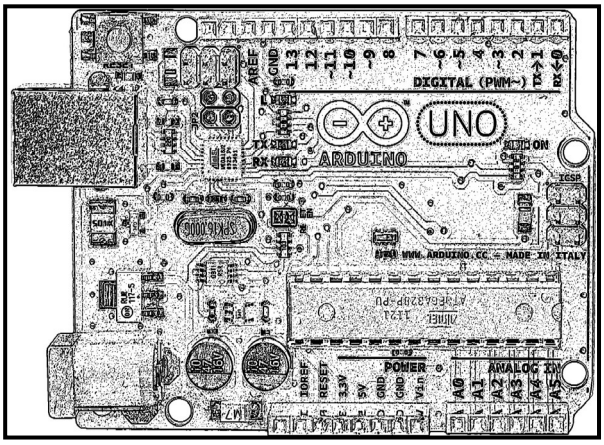
Arduino nació en el año 2005 en el Instituto de Diseño Interactivo de Ivrea (Italia), centro académico donde los estudiantes se dedicaban a experimentar con la interacción entre humanos y diferentes dispositivos (muchos de ellos basados en microcontroladores) para conseguir generar espacios únicos, especialmente artísticos. Arduino apareció por la necesidad de contar con un dispositivo para utilizar en las aulas que fuera de bajo coste, que funcionase bajo cualquier sistema operativo y que contase con documentación adaptada a gente que quisiera empezar de cero. La idea original fue, pues, fabricar la placa para uso interno de la escuela.

No obstante, el Instituto se vio obligado a cerrar sus puertas precisamente en 2005. Ante la perspectiva de perder en el olvido todo el desarrollo del proyecto Arduino que se había ido llevando a cabo durante aquel tiempo, se decidió liberarlo y abrirlo a "la comunidad" para que todo el mundo tuviera la posibilidad de participar en la evolución del proyecto, proponer mejoras y sugerencias y mantenerlo "vivo". Y así ha sido: la colaboración de muchísima gente ha hecho que Arduino poco a poco haya llegado a ser lo que es actualmente: un proyecto de hardware y software libre de ámbito mundial.

El principal responsable de la idea y diseño de Arduino, y la cabeza visible del proyecto es el llamado "Arduino Team", formado por Massimo Banzi (profesor en aquella época del Instituto Ivrea), David Cuartielles (profesor de la Escuela de Artes y Comunicación de la Universidad de Mälmo, Suecia), David Mellis (por aquel entonces estudiante en Ivrea y actualmente miembro del grupo de investigación High-Low Tech del MIT Media Lab), Tom Igoe (profesor de la Escuela de Arte Tisch de Nueva York), y Gianluca Martino (responsable de empresa fabricante de los prototipos de las placas, cuya web oficial es: http://www.smartprojects.it).

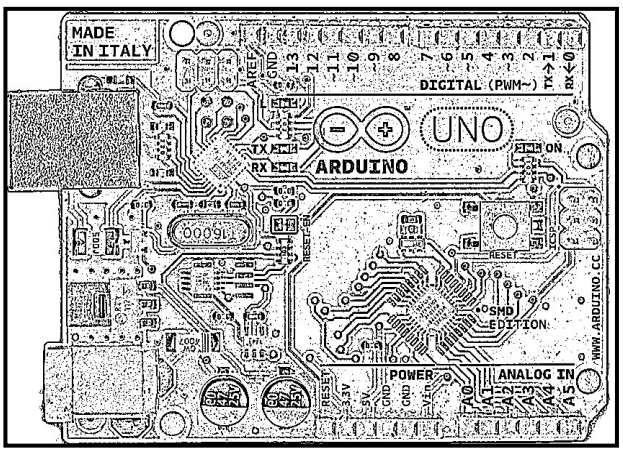
CARACTERÍSTICAS DEL MICRO DE LA PLACA ARDUINO UNO

Ya se ha comentado anteriormente que existen varios tipos de placas Arduino, cada una con características específicas que hay que conocer para poder elegir el modelo que más nos convenga según el caso. No obstante, existe un modelo "estándar" de placa, que es el más utilizado con diferencia y que es el que utilizaremos también nosotros en este libro en todos los proyectos: la placa Arduino UNO. Desde que apareció en 2010 ha sufrido tres revisiones, por lo que el modelo actual se suele llamar UNO Rev3 o simplemente UNO R3.



El encapsulado del microcontrolador

La imagen anterior muestra la placa Arduino UNO en su variante convencional. La imagen siguiente muestra la variante llamada Arduino UNO SMD. La única diferencia entre ambas placas es el encapsulado físico del microcontrolador incorporado: ambas tienen el mismo modelo, pero la placa convencional lo lleva montado en formato DIP ("Dual In-line Package") y la placa SMD lo lleva en formato SMD ("Surface Mount Device"). Tal como se puede apreciar en ambas figuras, el formato DIP (visualmente, un gran rectángulo en el centro-inferior-derecha de la placa) es mucho más grande que el formato SMD (visualmente, un pequeño cuadrado ubicado en diagonal en el centro-inferior-derecha de la placa).



Una diferencia importante entre el formato SMD y el DIP es que el primero está soldado a la superficie de la placa (mediante una tecnología llamada precisamente "de montaje superficial" –en inglés, SMT, de "surface mount technology" –), mientras que el segundo está conectado a la placa mediante una serie de patillas metálicas (las cuales son, de hecho, las patillas de E/S del microcontrolador) que se pueden separar fácilmente y que permiten la substitución del microcontrolador por otro si fuera necesario. En la práctica, esto no nos debería importar demasiado a no ser que deseemos separar y reutilizar el microcontrolador de nuestra placa en otras placas o montajes; en ese caso, deberíamos optar por el formato DIP.

Aunque en la placa Arduino solamente dispongamos para nuestro microcontrolador de estas dos alternativas de encapsulado (DIP o SMD), el mundo de los chips en general no es tan sencillo, ya que existen muchas variantes de los dos encapsulados anteriores, además de otros tipos de encapsulados diferentes. Esto es debido a las diferentes necesidades que existen respecto la disponibilidad de espacio físico y la disposición de los conectores en las PCBs que utilicemos. Si se desea consultar qué encapsulados son los más importantes en el mundo electrónico, se puede descargar de http://goo.gl/OU47S un breve documento que los resume. Si aún se desea obtener una información más exhaustiva, recomiendo consultar la página http://www.siliconfareast.com/ic-package-types.htm.

De todas formas, en los proyectos de este libro no nos preocuparemos demasiado por los encapsulados de los chips utilizados, ya que para trabajar con ellos haremos uso de placas "breakout". Una placa "breakout" es una placa PCB que lleva soldado un chip (o más) junto con los conectores y la circuitería necesaria para permitir enchufar a ella dispositivos externos de una forma fácil y rápida. Por tanto, cuando necesitemos en nuestros proyectos algún chip en particular, recurriremos a alguna placa breakout que lo incorpore y entonces simplemente deberemos utilizar los conectores ofrecidos por ella para poner el chip en comunicación con el resto del circuito. Así pues, solo en el caso de necesitar soldar un chip individual a una placa es cuando su tipo de encapsulado lo deberíamos tener en cuenta, pero esto es un tema avanzado que no abordaremos.

Aclaremos que otros componentes simples que no son circuitos integrados (tales como resistencias, condensadores, diodos, fusibles, etc.) también pueden estar encapsulados en formato SMD para optimizar al máximo el espacio físico ocupado. En estos casos, la variedad de formas y tamaños, aunque estandarizada, es inmensa, y se sale de los objetivos de este libro profundizar en este tema. Baste decir que muchos de estos encapsulados se suelen distinguir por un código de cuatro dígitos, los dos primeros de los cuales indican la longitud del componente y los dos últimos su anchura, en centésimas de pulgadas. Por ejemplo, el encapsulado (por otra parte muy común) "0603" indicaría que el componente en cuestión tiene un tamaño de 0,06" x 0,03". Otros encapsulados comunes son el "0805" y el "0402".

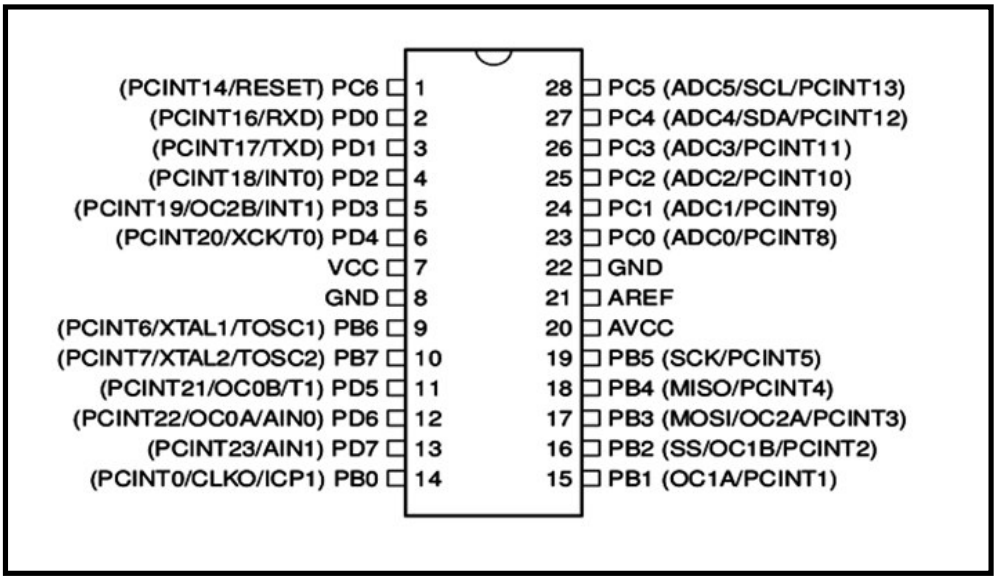
El modelo del microcontrolador

El microcontrolador que lleva la placa Arduino UNO es el modelo ATmega328P de la marca Atmel. La "P" del final significa que este chip incorpora la tecnología "Picopower" (propietaria de Atmel), la cual permite un consumo eléctrico sensiblemente menor comparándolo con el modelo equivalente sin "Picopower", el Atmega328 (sin la "P"). De todas formas, aunque el ATmega328P pueda trabajar a un voltaje menor y consumir menos corriente que el Atmega328 (especialmente en los modos de hibernación), ambos modelos son funcionalmente idénticos.

Al igual que ocurre con el resto de microcontroladores usados en otras placas Arduino, el ATmega328P tiene una arquitectura de tipo AVR, arquitectura desarrollada por Atmel y en cierta medida "competencia" de otras arquitecturas como por ejemplo la PIC del fabricante Microchip. Más concretamente, el ATmega328P pertenece a la subfamilia de microcontroladores "megaAVR". Otras subfamilias de la arquitectura AVR son la "tinyAVR" (cuyos microcontroladores son más limitados y se identifican con el nombre de ATtiny) y la "XMEGA" (cuyos microcontroladores son más capaces y se identifican con el nombre de ATxmega), pero no las estudiaremos ya que las placas Arduino no incorporan microcontroladores de esas familias.

De todas formas, si se desea saber más sobre la arquitectura AVR y los modelos y características que ofrecen los microcontroladores construidos de esta forma, no hay nada mejor como consultar la web de la propia empresa fabricante: http://www.atmel.com/products/microcontrollers/avr/default.aspx). Y más en concreto, si se desea consultar la especificación técnica del ATmega328P (aunque para los proyectos de este libro no será necesario) se puede descargar aquí: http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf .

Lo que sí nos puede venir bien es conocer la disposición concreta de las patillas (llamadas también "pines") de entrada/salida del microcontrolador, ya que, aunque hemos dicho anteriormente que en general todos los pines de E/S sirven para comunicar el microcontrolador con el mundo exterior, es cierto que cada pin suele tener una determinada función específica. Como cada modelo de microcontrolador tiene un número y ubicación de pines diferente, en nuestro caso concreto deberemos tener a mano la disposición de pines del ATmega328P. La figura siguiente muestra esta disposición en el encapsulado de tipo DIP, y ha sido obtenida de la especificación técnica mencionada en el párrafo anterior. Nota: el circulito que aparece en la parte superior de la figura indica el lugar donde existe una muesca en el encapsulado real, de manera que así sea fácil distinguir la orientación de los pines.



Observando la imagen se puede saber qué pin es el que recibe la alimentación eléctrica (señalado como "VCC"), qué dos pines están conectados a tierra (los señalados como "GND"), qué pines son los de E/S (señalados como PBx, PCx o PDx) y la existencia de otros pines más específicos como el AVCC (donde se recibe la alimentación suplementaria para el convertidor analógico-digital interno del chip) o el AREF (donde se recibe la referencia analógica para dicho convertidor –esto lo estudiaremos más adelante–). También se puede observar que junto el nombre de los pines de E/S se indica entre paréntesis las funciones especializadas que cada uno de ellos tiene en particular (además de su función genérica de entrada/salida). Algunas de estas funciones específicas las iremos estudiando a lo largo del libro, como por ejemplo la función de "reset" del microcontrolador, o la comunicación con el exterior usando el protocolo serie o el SPI o el I^2C, o el uso de interrupciones, o el de las salidas PWM, etc.

Las memorias del microcontrolador

Otra cosa que hay que saber de los microcontroladores son los tipos y cantidades de memoria que alojan en su interior. En el caso del ATmega328P tenemos:

Memoria Flash: memoria persistente donde se almacena permanentemente el programa que ejecuta el microcontrolador (hasta una nueva reescritura si se da el caso). En el caso del ATmega328P tiene una capacidad de 32KB.

Breve nota sobre las unidades de medida de la información:

Es buen momento para repasar cómo se mide la cantidad de datos almacenados en una memoria (o transferidos por algún canal). El sistema de medición de la información es el siguiente:

1 bit es la unidad mínima, y puede valer 0 o 1.
1 byte es un grupo de 8 bits.
1 kilobyte (a veces escrito como KB) es un grupo de 1024 bytes (es decir, 8192 bits). 1 megabyte (MB) es un grupo de 1024 KB (es decir, 1048576 bytes).
1 gigabyte (GB) es un grupo de 1024 MB.

Observar que los múltiplos "kilo", "mega" y "giga" hacen referencia a agrupaciones de 1024 (2^{10}) elementos, en vez de lo que suele ocurrir con el resto de unidades de medida del mundo físico, donde son agrupaciones de 1000 elementos. Esta discrepancia es debe a la naturaleza intrínsecamente binaria de los componentes electrónicos.

Observar también que no es lo mismo Kbit ("kilobit") que KB ("kilobyte"). En el primer caso estaríamos hablando de 1024 bits, y en el segundo de 1024 bytes (es decir, 8192 bits, ocho veces más).

En los microcontroladores que vienen incluidos en la placa Arduino no podemos usar toda la capacidad de la memoria Flash porque existen 512 bytes (el llamado "bootloader block") ocupados ya por un código preprogramado de fábrica (el llamado "bootloader" o "gestor de arranque"), el cual nos permite usar la placa Arduino de una forma sencilla y cómoda sin tener que conocer las interioridades electrónicas más avanzadas del microcontrolador. Los ATmega328P que podamos adquirir individualmente normalmente no incluyen de fábrica este pequeño programa, por lo que sí que ofrecen los 32 KB íntegros, pero a cambio no podremos esperar conectarlos a una placa Arduino y que funcionen sin más ya que les faltará tener grabada esa "preconfiguración". De los gestores de arranque, de su uso y su importancia hablaremos en un próximo apartado.

> **Memoria SRAM**: memoria volátil donde se alojan los datos que en ese instante el programa (grabado separadamente en la memoria Flash, recordemos) necesita crear o manipular para su correcto funcionamiento. Estos datos suelen tener un contenido variable a lo largo del tiempo de ejecución del programa y cada uno es de un tipo concreto (es decir, un dato puede contener un valor numérico entero, otro un número decimal, otro un valor de tipo carácter… también pueden ser cadenas de texto fijas u otros tipos de datos más especiales). Independientemente del tipo de dato, su valor siempre será eliminado cuando se deje de alimentar eléctricamente al microcontrolador. En el caso del ATmega328P esta memoria tiene una capacidad de 2KB.

Si necesitáramos ampliar la cantidad de memoria SRAM disponible, siempre podríamos adquirir memorias SRAM independientes y conectarlas al microcontrolador utilizando algún protocolo de comunicación conocido por este (como SPI o I^2C, de los cuales hablaremos enseguida); no obstante, esto no será necesario en los proyectos de este libro.

> **Memoria EEPROM**: memoria persistente donde se almacenan datos que se desea que permanezcan grabados una vez apagado el microcontrolador para poderlos usar posteriormente en siguientes reinicios. En el caso del ATmega328P esta memoria tiene una capacidad de 1 KB, por lo que se puede entender como una tabla de 1024 posiciones de un byte cada una.

Si necesitáramos ampliar la cantidad de memoria EEPROM disponible, siempre podemos adquirir memorias EEPROM independientes y conectarlas al microcontrolador utilizando algún protocolo de comunicación conocido por este (como SPI o I^2C, de los cuales hablaremos enseguida). O bien, alternativamente, adquirir tarjetas de memoria de tipo SD ("Secure Digital") y comunicarlas mediante un circuito específico al microcontrolador. Las memorias SD son en realidad simples memorias Flash, encapsuladas de una forma concreta; son ampliamente utilizadas en cámaras digitales de foto/vídeo y en teléfonos móviles de última generación, ya que ofrecen muchísima capacidad (varios gigabytes) a un precio barato. La razón por la cual este tipo de tarjetas son capaces de ser reconocidas por el ATmega328P es porque pueden funcionar utilizando el protocolo de comunicación SPI.

Podemos deducir de los párrafos anteriores que la arquitectura a la que pertenece el chip ATmega328P (y en general, toda la familia de microcontroladores AVR) es de tipo Harvard. En este tipo de arquitectura, la memoria que aloja los datos (en nuestro caso, la SRAM o la EEPROM) está separada de la memoria que aloja las instrucciones (en nuestro caso, la Flash), por lo que ambas memorias se comunican con la CPU de forma totalmente independiente y en paralelo, consiguiendo así una mayor velocidad y optimización. Otro tipo de arquitectura (que es la que vemos en los PCs) es la arquitectura Von Neumann, en la cual la CPU está conectada a una memoria RAM única que contiene tanto las instrucciones del programa como los datos, por lo que la velocidad de operación está limitada (entre otras cosas) por el efecto cuello de botella que significa un único canal de comunicación para datos e instrucciones.

Los registros del microcontrolador

Los registros son espacios de memoria existentes dentro de la propia CPU de un microcontrolador. Son muy importantes porque tienen varias funciones imprescindibles: sirven para albergar los datos (cargados previamente desde la memoria SRAM o EEPROM) necesarios para la ejecución de las instrucciones previstas próximamente (y así tenerlos perfectamente disponibles en el momento adecuado); sirven también para almacenar temporalmente los resultados de las instrucciones recientemente ejecutadas (por si se necesitan en algún instante posterior) y sirven además para alojar las propias instrucciones que en ese mismo momento estén ejecutándose.

Su tamaño es muy reducido: tan solo tienen capacidad para almacenar unos pocos bits cada uno. Pero este factor es una de las características más importantes de cualquier microcontrolador, ya que cuanto mayor sea el número de bits que “quepan” en sus registros, mayores serán sus prestaciones, en cuanto a poder de cómputo y velocidad de ejecución. En efecto, es fácil ver (simplificando mucho) que un microcontrolador con registros el doble de grandes que otro podrá procesar el doble de cantidad de datos y por tanto, trabajar el “doble de rápido” aun funcionando los dos al mismo ritmo. De hecho, es tan importante esta característica que cuando escuchamos que un microcontrolador es de “8 bits” o de “32 bits”, nos estamos refiriendo precisamente a este dato: al tamaño de sus registros.

Dependiendo de la utilidad que vayamos a darle al microcontrolador, será necesario utilizar uno con un tamaño de registros suficiente. Por ejemplo, el control de un electrodoméstico sencillo como una batidora no requiere más que un microcontrolador de 4 u 8 bits. En cambio, el sistema de control electrónico del motor de un coche o el de un sistema de frenos ABS se basan normalmente en un microcontrolador de 16 o 32 bits.

El chip ATmega328P es de 8 bits. De hecho, todos los microcontroladores que incorporan las diferentes placas Arduino son de 8 bits excepto el incorporado en la placa Arduino Due, que es de 32 bits.

Los protocolos de comunicación I2C/TWI y SPI

Cuando se desea transmitir un conjunto de datos desde un componente electrónico a otro, se puede hacer de múltiples formas. Una de ellas es estableciendo una comunicación “serie”; en este tipo de comunicación la información es transmitida bit a bit (uno tras otro) por un único canal, enviando por tanto un solo bit en cada momento. Otra manera de transferir datos es mediante la llamada comunicación “paralela”, en la cual se envían varios bits simultáneamente, cada uno por un canal separado y sincronizado con el resto.

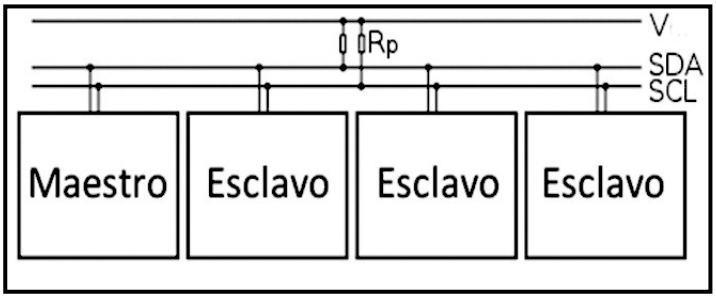
El microcontrolador, a través de algunos de sus pines de E/S, utiliza el sistema de comunicación serie para transmitir y recibir órdenes y datos hacia/desde otros componentes electrónicos. Esto es debido sobre todo a que en una comunicación serie solo se necesita en teoría un único canal (un único "cable"), mientras que en una comunicación en paralelo se necesitan varios cables, con el correspondiente incremento de complejidad, tamaño y coste del circuito resultante.

No obstante, no podemos hablar de un solo tipo de comunicación serie. Existen muchos protocolos y estándares diferentes basados todos ellos en la transferencia de información en serie, pero implementando de una forma diferente cada uno los detalles específicos (como el modo de sincronización entre emisor y receptor, la velocidad de transmisión, el tamaño de los paquetes de datos, los mensajes de conexión y desconexión y de dar paso al otro en el intercambio de información, los voltajes utilizados, etc.). De entre el gran número de protocolos de comunicación serie reconocidos por la inmensa variedad de dispositivos electrónicos del mercado, los que nos interesarán conocer son los que el ATmega328P es capaz de comprender y por tanto, los que podrá utilizar para contactar con esa variedad de periféricos. En este sentido, los estándares más importantes son:

> **I²C** (Inter-Integrated Circuit, también conocido con el nombre de **TWI** –de "TWo-wIre", literalmente "dos cables" en inglés–): es un sistema muy utilizado en la industria principalmente para comunicar circuitos integrados entre sí. Su principal característica es que utiliza dos líneas para transmitir la información: una (llamada línea "SDA") sirve para transferir los datos (los 0s y los 1s) y otra (llamada línea "SCL") sirve para enviar la señal de reloj. En realidad también se necesitarían dos líneas más: la de alimentación y la de tierra común, pero estas ya se presuponen existentes en el circuito.

Por "señal de reloj" se entiende una señal binaria de una frecuencia periódica muy precisa que sirve para coordinar y sincronizar los elementos integrantes de una comunicación (es decir, los emisores y receptores) de forma que todos sepan cuándo empieza, cuánto dura y cuándo acaba la transferencia de información. En hojas técnicas y diagramas a la señal de reloj en general se le suele describir como CLK (del inglés "clock").

Cada dispositivo conectado al bus I²C tiene una dirección única que lo identifica respecto el resto de dispositivos, y puede estar configurado como "maestro" o como "esclavo". Un dispositivo maestro es el que inicia la transmisión de datos y además genera la señal de reloj, pero no es necesario que el maestro sea siempre el mismo dispositivo: esta característica se la pueden ir intercambiando ordenadamente los dispositivos que tengan esa capacidad.



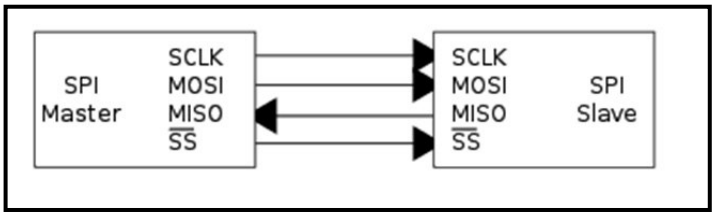
Tal como se muestra en el diagrama anterior, para funcionar correctamente tanto la línea "SDA" como la "SCL" necesitan estar conectadas mediante una resistencia "pull-up" a la fuente de alimentación común, la cual puede proveer un voltaje generalmente de 5 V o 3,3 V (aunque sistemas con otros voltajes pueden ser posibles).

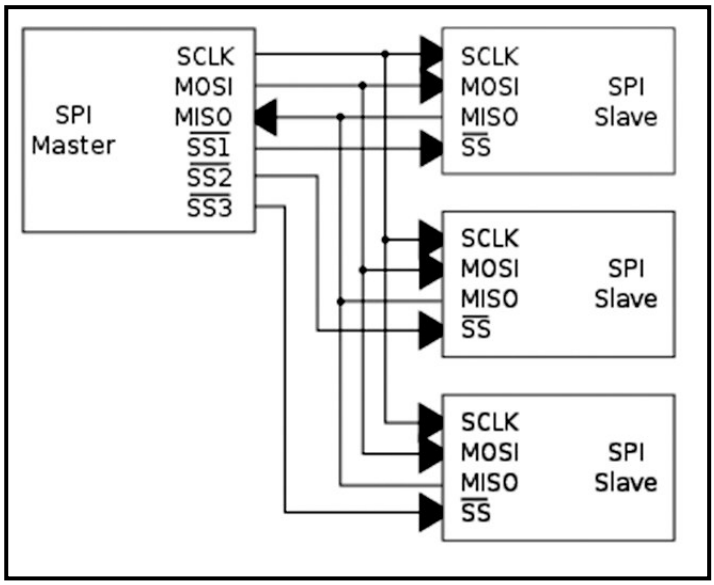
La velocidad de transferencia de datos es de 100 Kbits por segundo en el modo estándar (aunque también se permiten velocidades de hasta 3,4 Mbit/s). No obstante, al haber una única línea de datos, la transmisión de información es "half duplex" (es decir, la comunicación solo se puede establecer en un sentido al mismo tiempo) por lo que en el momento que un dispositivo empiece a recibir un mensaje, tendrá que esperar a que el emisor deje de transmitir para poder responderle.

> **SPI** (Serial Peripheral Interface): al igual que el sistema I²C, el sistema de comunicación SPI es un estándar que permite controlar (a cortas distancias) casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie sincronizado (es decir, regulado por un reloj). Igualmente, un dispositivo conectado al bus SPI puede ser "maestro" –en inglés, "master" – o "esclavo" –en inglés, "slave"–, donde el primero es el que inicia la transmisión de datos y además genera la señal de reloj (aunque, como con I²C , con SPI tampoco es necesario que el maestro sea siempre el mismo dispositivo) y el segundo se limita a responder.

La mayor diferencia entre el protocolo SPI y el I²C es que el primero requiere de cuatro líneas ("cables") en vez de dos. Una línea (llamada normalmente "SCK") envía a todos los dispositivos la señal de reloj generada por el maestro actual; otra (llamada normalmente "SS") es la utilizada por ese maestro para elegir en cada momento con qué dispositivo esclavo se quiere comunicar de entre los varios que puedan estar conectados (ya que solo puede transferir datos con un solo esclavo a la vez); otra (llamada normalmente "MOSI") es la línea utilizada para enviar los datos –0s y 1s– desde el maestro hacia el esclavo elegido; y la otra (llamada normalmente "MISO") es la utilizada para enviar los datos en sentido contrario: la respuesta de ese esclavo al maestro. Es fácil ver que, al haber dos líneas para los datos la transmisión de información es "full duplex" (es decir, que la información puede ser transportada en ambos sentidos a la vez).

En las siguientes figuras se muestra el esquema de líneas de comunicación existentes entre un maestro y un esclavo y entre un maestro y tres esclavos respectivamente. Se puede observar que, para el caso de la existencia de varios esclavos es necesario utilizar una línea "SS" diferente por cada uno de ellos, ya que esta línea es la que sirve para activar el esclavo concreto que en cada momento el maestro desee utilizar (esto no pasa con las líneas de reloj, "MOSI" y "MISO", que son compartidas por todos los dispositivos).Técnicamente hablando, el esclavo que reciba por su línea SS un valor de voltaje BAJO será el que esté seleccionado en ese momento por el maestro, y los que reciban el valor ALTO no lo estarán (de ahí el subrayado superior que aparece en la figura).





Como se puede ver, el protocolo SPI respecto el I²C tiene la desventaja de exigir al microcontrolador dedicar muchos más pines de E/S a la comunicación externa. En cambio, como ventaja podemos destacar que es más rápido y consume menos energía que I²C.

Tal como se puede observar en la figura que muestra la disposición de pines del microcontrolador ATmega328P (página 75), los pines correspondientes a las líneas I^2C SDA y SCL son los números 27 y 28, respectivamente, y los pines correspondientes a las líneas SPI SS, MOSI, MISO y SCK son los números 16, 17, 18 y 19, respectivamente. Si se necesitaran más líneas SS (porque haya más de un dispositivo esclavo conectado en nuestro circuito), se podría utilizar cualquier otro pin de E/S siempre que respete el convenio de poner el valor de su voltaje de salida a BAJO cuando se desee trabajar con el dispositivo esclavo asociado y poner a ALTO el resto de pines SS.

El gestor de arranque del microcontrolador

Ya hemos comentado anteriormente que dentro de la memoria Flash del microcontrolador incluido en las placas Arduino viene pregrabado de fábrica un pequeño programa llamado “bootloader” o “gestor de arranque”, que resulta imprescindible para un cómodo y fácil manejo de la placa en cuestión. Este software (también llamado “firmware”, porque es un tipo de software que raramente se modifica) ocupa, en la placa Arduino UNO, 512 bytes de espacio en un apartado especial de la memoria Flash, el llamado “bootloader block”, pero en otros modelos de placas Arduino puede ocupar más (por ejemplo, en el modelo Leonardo ocupa 4 Kilobytes).

La función de este firmware es gestionar de forma automática el proceso de grabación en la memoria Flash del programa que queremos que el microcontrolador ejecute. Lógicamente, el bootloader realizará esta grabación más allá del "bootloader block" para no sobrescribirse a sí mismo.

Más concretamente, el bootloader se encarga de recibir nuestro programa de parte del entorno de desarrollo Arduino (normalmente mediante una transmisión realizada a través de conexión USB desde el computador donde se está ejecutando dicho entorno hasta la placa) para proceder seguidamente a su correcto almacenamiento en la memoria Flash, todo ello de forma automática y sin que nos tengamos que preocupar de las interioridades electrónicas del proceso. Una vez realizado el proceso de grabación, el bootloader termina su ejecución y el microcontrolador se dispone a procesar de inmediato y de forma permanente (mientras esté encendido) las instrucciones recientemente grabadas.

En la placa Arduino UNO, el bootloader siempre se ejecuta durante el primer segundo de cada reinicio. Durante esos instantes, el gestor de arranque se espera a recibir una serie de instrucciones concretas de parte del entorno de desarrollo para interpretarlas y realizar la correspondiente carga de un posible programa. Si esas instrucciones no llegan pasado ese tiempo, el bootloader termina su ejecución e igualmente se empieza a procesar lo que haya en ese momento en la memoria Flash.

Estas instrucciones internas de programación de memorias Flash son ligeramente diferentes según el tipo de bootloader que tenga la placa, pero casi todas son variantes del conjunto de instrucciones ofrecido oficialmente por Atmel para la programación de sus microcontroladores, el llamado protocolo STK500 (http://www.atmel.com/tools/STK500.aspx). Un ejemplo es el bootloader que tiene pregrabado el ATmega328P del Arduino UNO, basado en un firmware libre llamado Optiboot (http://code.google.com/p/optiboot), el cual logra una velocidad de grabación de 115 kilobits de programa a cargar por segundo gracias al uso de instrucciones propias derivadas del "estándar" STK500. Otro ejemplo de bootloader derivado del protocolo STK500 es el bootloader "wiring", grabado de fábrica en el microcontrolador de la placa Arduino Mega. El bootloader que viene en la placa Leonardo (llamado "Caterina") es diferente, ya que entiende otro conjunto de instrucciones independiente llamado AVR109 (también oficial de Atmel). Toda esta información se puede obtener consultando el contenido del fichero llamado "boards.txt", descargado junto con el entorno de desarrollo de Arduino.

Por conveniencia, dentro del paquete instalador del entorno de desarrollo de Arduino (descargable de su web oficial, para más detalles ver el capítulo siguiente), se distribuyen además copias exactas bit a bit de los bootloaders oficiales que vienen grabados en los diferentes microcontroladores Arduino. Estas copias exactas son ficheros con extensión ".hex" que tienen un formato interno llamado "Intel Hex Format". Para el uso normal de nuestra placa no necesitamos para nada estos ficheros ".hex", pero si dispusiéramos de un programador ISP y en algún momento tuviéramos que "reponer" un bootloader dañado (o bien grabar un bootloader a algún microcontrolador que no tuviera ninguno), Arduino nos ofrece estos ficheros para cargarlos en la memoria Flash de nuestro microcontrolador siempre que queramos.

El formato Intel Hex Format es el utilizado por todos los chips AVR para almacenar contenido en sus memorias Flash, por lo que hay que aclarar que no solamente los bootloaders son alojados internamente de esta forma en la memoria Flash, sino que todos nuestros propios programas que escribamos en el entorno de desarrollo también serán alojados allí en formato ".hex" (aunque de estos detalles no nos debemos preocupar por ahora).

Evidentemente, los bootloaders Arduino también son software libre, por lo que al igual que ocurre con el entorno de programación Arduino, siempre tendremos disponible su código fuente (escrito en lenguaje C) para poder conocer cómo funciona internamente e incluso para poderlo modificar, si así se estima oportuno.