Computadoras, microcomputadoras y microcontroladores

La amplia diseminación de estas tecnologías y sus aplicaciones en diferentes áreas de conocimiento tiende a confundir los alcances de cada una, por ello es importante empezar con la definición de Nuñez Rodríguez J. A. (2012).

> Computadora. Dispositivo capaz de aceptar información, almacenarla, aplicarle un proceso y registrar los resultados de este proceso.
>
> Microcomputadora. Computadora que utiliza un microprocesador como CPU (*Central Process Unity*), es pequeña en comparación con los servidores, estaciones de trabajo o *mainframes*.
>
> Microcontrolador. Circuito integrado programable capaz de ejecutar las instrucciones grabadas en su memoria, por lo general se utiliza para realizar tareas repetitivas.
>
> Debido a que la funcionalidad principal de una computadora es procesar datos para convertirlos en información, se puede tomar la definición de una computadora para los dispositivos Arduino y Raspberry Pi.

Otros microcontroladores

Existen diversas opciones de plataformas de microcontroladores como: Parallax Basic Stamp, Netmedia´s BX-24, Phidgets, MIT´s Handyboard y otras más, lo cual se considera una respuesta positiva en este ámbito de desempeño, no obstante obliga a los usuarios a realizar una evaluación sobre la mejor opción a partir del proyecto que desarrollan. En la siguiente tabla se presentan varios de los microcontroladores señalados anteriormente junto con el lenguaje que utilizan:

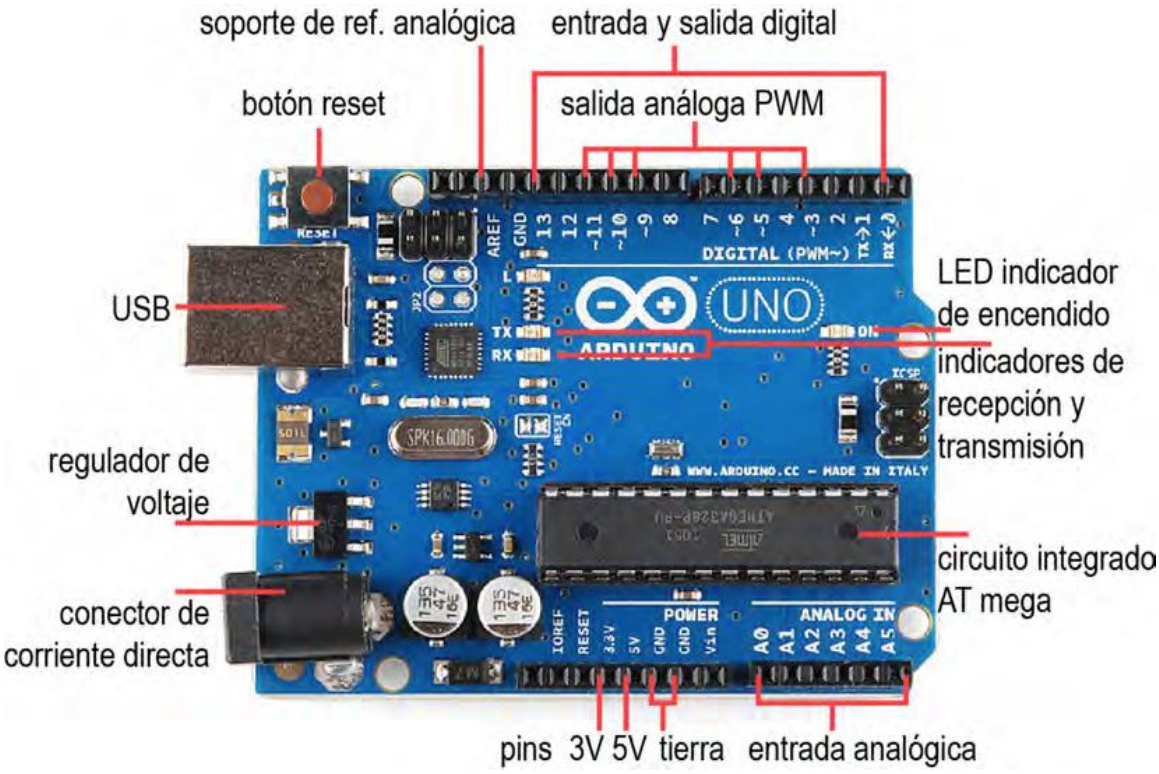
Microcontroller	Lenguaje
Parallax Basic Stamp	BASIC
Netmedia´s BX-24	PBASIC
MIT´s Handyboard	Interactive C
Phidgets	C, C/C++, Phyton, Java, Visual Basic .NET, Visdual Basic 6.0, Cocoa, Android Java, IOS, Applescript, Autoit, Ruby, LabVIEW, MATLAB, Flash AS3, Max/MSP, Adobe Director, Live Cocoa, Delphi.
Arduino	IDE Arduino

ARDUINO

Arduino es una plataforma de microntroladores *open-source* con capacidades de lectura para datos de entrada, los cuales convierten en salida. Los datos entrantes pueden registrarse a través de la luz, los sensores, los botones de presión, mensajes de Twitter, etcétera; y estos a su vez pueden activar motores, luminarias o publicar algún dato específico en línea.

En general, la información que se obtiene corresponde a datos del mundo físico que son registrados mediante sensores, de forma tal que dependiendo de los alcances de cada proyecto habrá que evaluar los componentes que se añaden para que el sistema funcione de forma óptima. De forma adicional, Arduino puede trabajar con software como Flash, Processing, MaxMSP, PureData, entre otros.

Las instrucciones se ejecutan mediante una interfaz IDE que utiliza un lenguaje propio de Arduino. Asimismo, es compatible con las plataformas Windows, Mac OSX y Linux.



Las características relacionadas con el funcionamiento de cada uno de los modelos de placa Arduino son diferentes dependiendo de los distintos procesadores y componentes, en el caso de la placa Arduino UNO, estos serían los requerimientos para lograr una óptima función:

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
0 Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Componentes Arduino

ATmega16U2
Es una memoria FLASH programable, con una capacidad de 16K, se identifica como un chip AVR, se puede considerar el disco duro del microcontrolador con excepción de que solo se lee. Estos componentes trabajan a 10MHz, con la capacidad de una memoria RAM a 1KB y 10 KB de almacenamiento. Es ideal para muchos proyectos por su tamaño y su bajo costo.

ATMega328P-PU
El chip microcontrolador RISC AVR es de 8 bits, con una memoria FLASH de 32KB lectura/escritura, EEPROM 1024KB, 23 conectores de entrada y salida de uso general, tres *timer/counters* con capacidades de comparación, interruptores internos y externos, conectores para comunicación serial SPI en seis canales que convierten señal análoga en digital, *watchdog timer* y oscilador interno. Funciona con una alimentación entre 1.8 a 5.5 volts.

Otros componentes
- Amplificadores operacionales a (+3.3V -3.3V) y (+5V -5V)
- Reguladores de voltaje
- Varistor
- Convertidores análogos/digitales

Características entre las diferentes placas de Arduino

Dependiendo de la naturaleza del proyecto desarrollado es importante analizar las características de cada uno de los microcontroladores Arduino; en específico al revisar las particularidades del procesador ATmega se puede tener un dato preciso de la capacidad existente para almacenar código.

En los casos señalados en rojo se observa que la disponibilidad de pines digitales en el modelo UNO y MEGA se triplica, lo cual quiere decir que se pueden conectar más dispositivos.

Arduino Yùn	An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
Arduino/Genuino Uno	An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
Arduino Diecimila or Duemilanove w/ ATmega168	*An ATmega168 running at 16 MHz with auto-reset.*
Arduino Nano w/ ATmega328	An ATmega328 running at 16 MHz with auto-reset. Has eight analog inputs.
Arduino/Genuino Mega 2560	An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
Arduino Mega	An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
Arduino Mega ADK	An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
Arduino Leonardo	An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In,

	20 Digital I/O and 7 PWM.
Arduino Micro	An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
Arduino Esplora	An ATmega32u4 running at 16 MHz with auto-reset.
Arduino Mini w/ ATmega328	An ATmega328 running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.
Arduino Ethernet	Equivalent to Arduino UNO with an Ethernet shield: An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
Arduino Fio	An ATmega328 running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ATmega328, 6 Analog In, 14 Digital I/O and 6 PWM.
Arduino BT w/ ATmega328	ATmega328 running at 16 MHz. The bootloader burned (4 KB) includes codes to initialize the on-board bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM..
LilyPad Arduino USB	An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.
LilyPad Arduino	An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328	An ATmega328 running at 16 MHz with auto-reset. Equivalent to Arduino Duemilanove or Nano w/ ATmega328; 6 Analog In, 14 Digital I/O and 6 PWM.
Arduino NG or older w/ ATmega168	An ATmega168 running at 16 MHz *without* auto-reset. Compilation and upload is equivalent to Arduino Diecimila or Duemilanove w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.
Arduino Robot Control	An ATmega328 running at 16 MHz with auto-reset.
Arduino Robot Motor	An ATmega328 running at 16 MHz with auto-reset.
Arduino Gemma	An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

Propiedades de los PINES digitales

- INPUT

En Arduino los pines declarados como inputs con pinMode() tienen un estado de alta impedancia (corriente baja–voltaje alto). Los pines de entrada demandan una corriente muy pequeña en el circuito, lo que favorece cambiar de un estado a otro con gran facilidad como ocurre cuando se implementan un sensor capacitivo, la lectura de *LED* o los sensores análogos.

Los pines digitales poseen una configuración PULLUP, para activarla se escribe el siguiente código:

```
pinMode(pin, INPUT); // configura pin de entrada
digitalWrite(pin, HIGH); //enciende el resistor pullup
```

NOTA.
El pin 13 no se debe usar como pin de entrada digital, porque está conectado a una resistencia y a un LED en la placa, el voltaje que maneja ese pin es de 1.7V en vez de 5V debido a su configuración.

- OUTPUT

Los pines configurados como de salida se consideran en un estado de baja impedancia (corriente alta–voltaje bajo), lo cual implica que alimentan de corriente a otros circuitos. Los pines de Atmega proveen corriente por encima de los 40 mA, suficiente para hacer brillar un LED o activar sensores, por ello es conveniente utilizar resistores de 470Ω a 1K, que controlen la corriente requerida para la aplicación que se realiza.

Propiedades de los PINES análogos

- Convertidor análogo digital

En Arduino los pines declarados como análogos son seis y poseen la propiedad de leer la señal análoga y convertirla en digital, con una resolución de 10 bits, regresando datos de 0 a 1023. La principal función de estos pines es la de leer señales análogas, sin embargo, también se pueden usar como salidas y entradas generales GPIO.

Al referirse en el código a estos puertos se deben usar A0, A1, A2, A3, A4, por ejemplo:

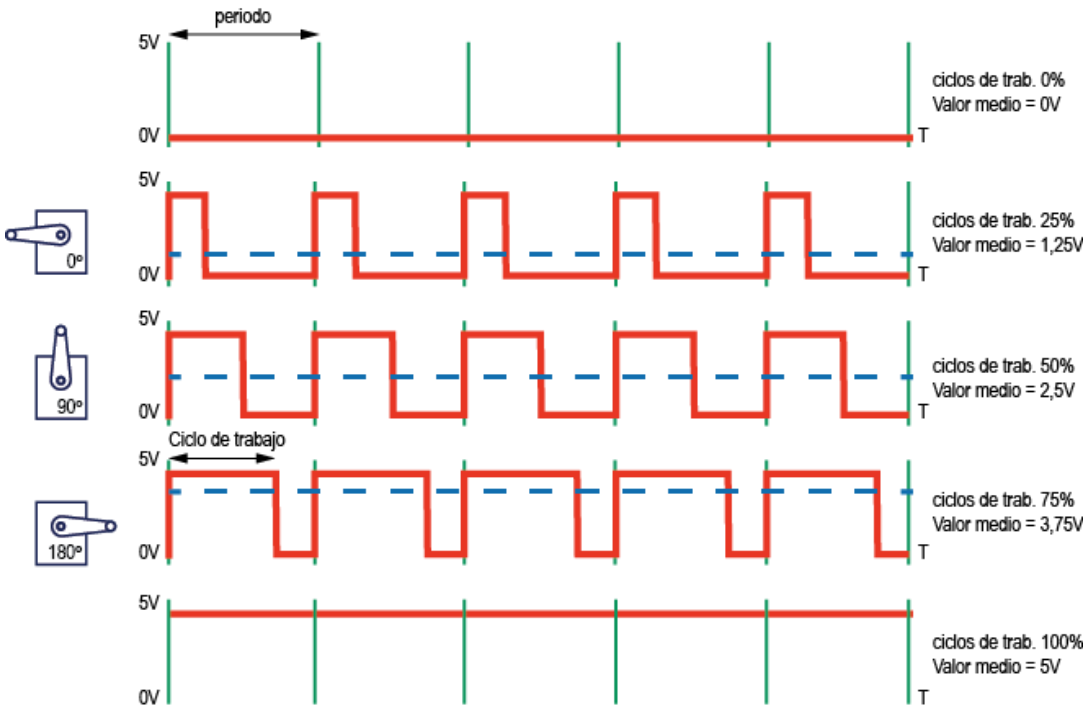
```
pinMode(A0, OUTPUT); // configura pin análogo de salida
digitalWrite(A0, HIGH); // Determina el estado del pin
```

NOTA.
Al definir un puerto análogo como OUTPUT, es probable que el comando analogRead no trabaje de forma correcta.
En la hoja de datos de Atmega se recomienda cerrar de forma temporal o cuando no haya actividad en los pines análogos porque la proximidad con el resto de estos causa ruido eléctrico y variación en el sistema análogo. Es suficiente con declararlos en estado OFF.

- Señales PWM

PWM (*Pulse Width Modulation*) o modulación por ancho de pulsos, es una técnica que logra producir el efecto de una señal análoga sobre una carga, a partir de la variación de frecuencia y el ciclo de trabajo de una señal digital. El ciclo de trabajo muestra la cantidad de tiempo que la señal se encuentra en un estado lógico alto o 5V.

La técnica consiste en proporcionar energía en forma de pulsos y no de manera continua. Este tipo de modulación genera una señal digital formada por ondas cuadradas de la misma frecuencia, en las que se modifica parte del periodo en el que la señal está activa.



En la gráfica se observa que el periodo de la señal varía en un lapso, mientras la señal está activa o alimentada por electricidad la podemos identificar como ancho de pulso o ciclo de trabajo. El valor de salida no es estrictamente 0V y 5V o 0 y 1 (valores binarios), sino la media de los valores obtenidos a lo largo de cada periodo de la señal, finalmente se obtendrán valores de tensión intermedios.

En Arduino la frecuencia es sobre los 500Hz y los intervalos entre las líneas verdes deben medir 2 milisegundos, analogWrite() con una escala de 0 a 255, las equivalencias con respecto a la gráfica que se muestra en la gráfica superior son:

Ciclo de trabajo 0%	analogWrite(0)
Ciclo de trabajo 25%	analogWrite(64)
Ciclo de trabajo 50%	analogWrite(127)
Ciclo de trabajo 75%	analogWrite(191)
Ciclo de trabajo 100%	analogWrite(255)

Memorias Arduino

Memoria FLASH. Almacena el sketch de Arduino.

SRAM (*Static Random Acces Memory*). Sitio donde el código de Arduino crea y manipula las variables cuando corren.

EEPROM. Espacio de memoria en el cual los programadores almacenan información.

La memoria Flash y EEPROM no son volátiles, es decir la información persiste aunque el dispositivo se apague, por ejemplo si el programa se encuentra listando

números pares y apaga en forma repentina, se verá que al encender muestra el último dato antes de que se desactivara y continuará su actividad. Por su parte, la SRAM es volátil y los datos se pierden cuando el código no se encuentra en ejecución.

Al ejecutar un *sketch* que excede las capacidades SRAM, el programa puede fallar de maneras diversas, podría simplemente no correr o hacerlo en forma extraña. Para checar si en realidad esto es lo que ocurre hay que intentar eliminar o resumir los comentarios, además de optimizar la estructura del código sin modificarlo, si se verifica que el problema es la capacidad de esta memoria, entonces se sugiere lo siguiente:

- Si el código llama a un programa de la computadora se puede intentar cambiar los datos y evitar que el procesador de Arduino realice cálculos, lo cual ayuda a optimizar los recursos de memoria.
- Si el código está revisando tablas u otros arreglos, se sugiere trabajar solamente con los datos necesarios, por ejemplo la variable int emplea dos bytes mientras que un byte utiliza solo uno.
- Si el código no incluye opciones para modificar cadenas de texto o datos mientras se está ejecutando se puede utilizar una memoria Flash en vez de la SDRAM, al hacerlo se utiliza el teclado PROGMEM.

Referencias

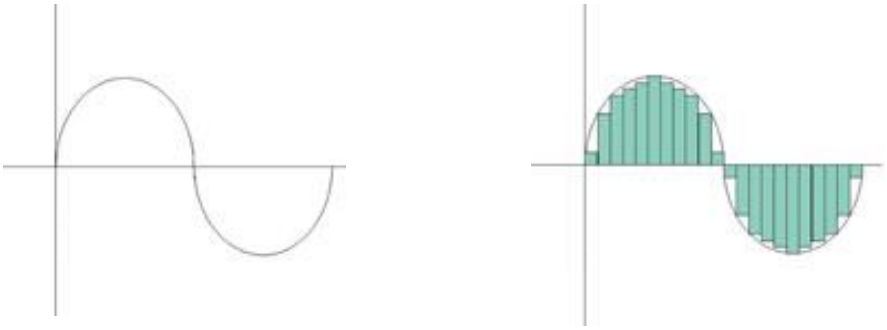
1. Nuñez Rodríguez J. A. (2014). *Diseño e integración de un sistema de adquisición de datos mediante el uso de Arduino y Raspberry Pi*.
2. Arduino, recuperado 20 de septiembre de 2015 de www.arduino.cc
3. *Creative Commons,* recuperado 15 de septiembre de 2015 de http://creativecommons.org/licenses/by-sa/3.0/
4. Arduino, recuperado 20 de septiembre de 2015 de https://www.arduino.cc/en/Main/ArduinoBoardUno
5. Señales PWM, recuperado 22 de septiembre de 2015 de http://digital.ni.com/public.nsf/allkb/AA1BDEA4AA224E3E86257CE400707527
6. Señales PWM, recuperado 22 de septiembre de 2015 de http://rduinostar.com/documentacion/general/salidas-digitales-pwm-arduino/
7. Números binarios, recuperado 23 de septiembre de 2015 de http://www.disfrutalasmatematicas.com/numeros/binarios-digitos.html

Señales digitales y análogas

La señal analógica implica que para llegar al dato requerido, se recorre la curva de la señal, pasando por los valores intermedios, además de que el trayecto es continuo y puede tomar valores infinitos.

Por su parte, la señal digital sufre saltos y pasa de un valor a otro sin mostrar los rangos intermedios, es discontinua y solo puede registrar valores o estados en 0 y 1, que pueden ser impulsos eléctricos de baja y alta tensión, interruptores, etcétera.

Aunque la señal digital está asociada a representaciones binarias (0 y 1), las primeras computadoras no usaban dos dígitos de manera básica, por ejemplo la ENIAC empleaba diez. Por ello, lo digital no siempre implica el uso de información binaria. La conversión de una onda analógica a una representación digital conlleva asociar lo más cercano posible una serie de ceros y unos a cierto valor de la onda.



A partir de lo señalado previamente, se deduce que para almacenar o transmitir una onda analógica en formato digital se requieren más números, en un mecanismo de aproximaciones sucesivas. Para identificar esta situación analicemos el caso del sonido o imágenes: si se usan más bits por segundo para digitalizar una canción o un video, el resultado será cercano a la señal original. ¿Cuántos bits se emplean en la digitalización de música o películas? Aunque cualquier usuario podría definir la cantidad de números para aproximarse a la onda analógica de su elección, esto sería poco práctico para compartir la información digital resultante con otros usuarios.

Por ello, existen estándares internacionales para discos compactos, de video, televisión directa al hogar o radio satelital, por mencionar algunos. Así es como se garantiza que el público en general pueda adquirir aparatos o programas para conversión analógica a digital o viceversa, producidos por diversos fabricantes, lo que permite reducir el costo y ampliar el uso de estas tecnologías.

Referencias

1. http://recursostic.educacion.es/secundaria/edad/4esotecnologia/quincena5/4q2_contenidos_1a.htm
2. http://www.enterate.unam.mx/Articulos/2004/septiembre/analdigi.htm