

Curso Arduino

Conceptos básicos

*Claudio Albeto Ibañez Garduño
Karen Navarrete Mejía*

*Estudiantes Facultad de Ingeniería, UNAM.
Ingeniería Eléctrica y Electrónica.*

Cursos, talleres, asesorías, material:

55 6643-6200

55 6255-7612

Correo: claudio.p4.unam@gmail.com

Facebook: @Claudhino

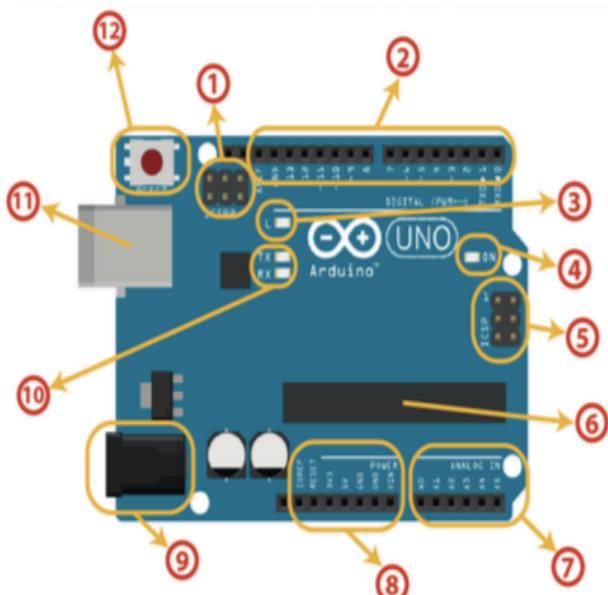
Conceptos básicos de micro controladores: Conociendo a Arduino

Por Antony García González

Arduino es una plataforma de electrónica abierta para la creación de prototipos basados en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos. Permite a través de la computadora y por medio de programación es posible que el usuario logre interactuar con circuitos electrónicos y controlarlos por software. De igual forma el Arduino es capaz de actuar de manera autónoma sin estar conectado a una computadora.

Existen múltiples modelos de Arduino con diferentes características. Cada modelo posee un nombre, formas, capacidades y funciones distintas. El modelo que más se utiliza para aprender lo básico de Arduino es el modelo UNO.

A continuación se muestra las partes de un Arduino UNO.



Nº	Descripción	Función
1, 5	Socket ICSP	Estos pines se utilizan con los programadores ICSP para introducirle instrucciones a Arduino sin el uso de la computadora.
2	Pines digitales	Son pines especiales para la entrada/salida de señales digitales. Tienen la capacidad de utilizar PWM. En el estado HIGH funcionando como Output pueden entregar 5 voltios y una corriente máxima de 40 mA. Funcionando como Input soportan un máximo de 5 voltios.
3	LED integrado	Este LED es el que indica cuando se está cargando el código en la placa. Permite verificar que la placa está funcionando bien si al presionar el botón "reset" parpadea dos veces. Cuando se está cargando código da información acerca del proceso de carga. Además está unido al pin 13, encendiéndose cuando hay un HIGH. Se utiliza para diferentes pruebas sin la necesidad de conectar un LED externo.
4	LED de estado	Indica si la placa está encendida o no
6	Socket del micro controlador	Aquí es donde se inserta el micro controlador. El modelo Arduino UNO permite retirar el circuito integrado para reemplazarlo por uno nuevo en caso de avería. El micro controlador, el ATMEGA328 para el modelo UNO, es el circuito integrado que da vida a Arduino.
7	Pines análogos	Estos pines se pueden utilizar para hacer lecturas de señales análogas en corriente directa no superiores a 5 voltios. También se pueden utilizar como pines digitales.
8	Entradas/Salidas de voltaje	Arduino posee sus propias entradas y salidas de voltaje. En Vin se puede aplicar un voltaje de una fuente externa.

Curso Arduino por Claudio & Karen

		para alimentar la placa; en 5 voltios y en 3.3 voltios se puede obtener voltajes para alimentar otros circuitos que no requieran demasiada corriente; el pin de Reset permite resetear el micro controlador sin tener que presionar el botón "reset". Los GND son los comunes de Arduino.
9	Power Jack	Una entrada que puede ser usada para alimentar Arduino utilizando un adaptador para corriente alterna.
10	LEDs de estado para comunicación serial	Los LEDs RX y TX indican cuando Arduino está transmitiendo (TX) o recibiendo (RX) información por medio de comunicación serial.
11	Puerto USB	Es donde se conecta el cable USB con el cual se da la comunicación con la computadora.
12	Botón "reset"	Reinicia el micro controlador.

Arduino se conecta a la computadora por medio de un cable USB. La programación del micro controlador se hace a través del Arduino IDE.

Este software se puede descargar de manera gratuita desde la página oficial de Arduino:

<http://arduino.googlecode.com/files/arduino-1.0.5-windows.zip>



Curso Arduino por Claudio & Karen

∞ INSTALANDO ARDUINO EN LA COMPUTADORA

Dentro del archivo descargado vienen algunas carpetas con ficheros dentro. Una de estas carpetas, bajo el nombre "drivers" contiene los controladores necesarios para instalar Arduino en cualquier computadora que utilice Windows.

Si el sistema operativo utilizado por el usuario es Windows Vista/7, basta con ir a inicio buscar Equipo, dar clic derecho/propiedades. Se abrirá una ventana en cuyo margen izquierdo tendrá la opción "Administrador de Dispositivos". Si Arduino está conectado por USB a la computadora aparecerá un dispositivo desconocido entre todos los dispositivos utilizados por la computadora. Se le da clic derecho al dispositivo desconocido y se busca la opción "Actualizar Software Controlador". Luego se elige "Buscar software controlador en el equipo" y se selecciona la carpeta que se descargó desde la página de Arduino. La computadora debería buscar el driver e instalarlo automáticamente. Cualquier notificación que aparezca debe ser aceptada.

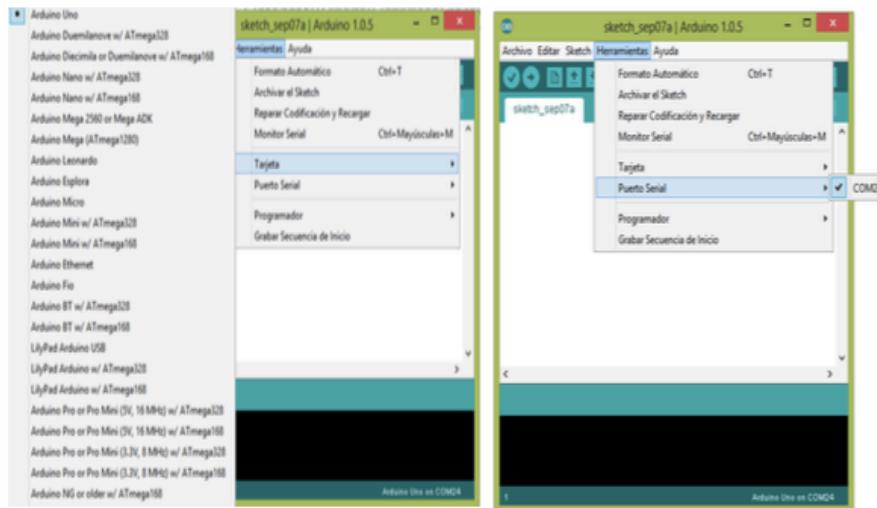
∞ CONOCIENDO ARDUINO IDE

Arduino IDE presenta una interfaz sencilla y amigable al usuario. Lo primero que se hace es seleccionar el puerto serie y el modelo de Arduino que se va a utilizar.

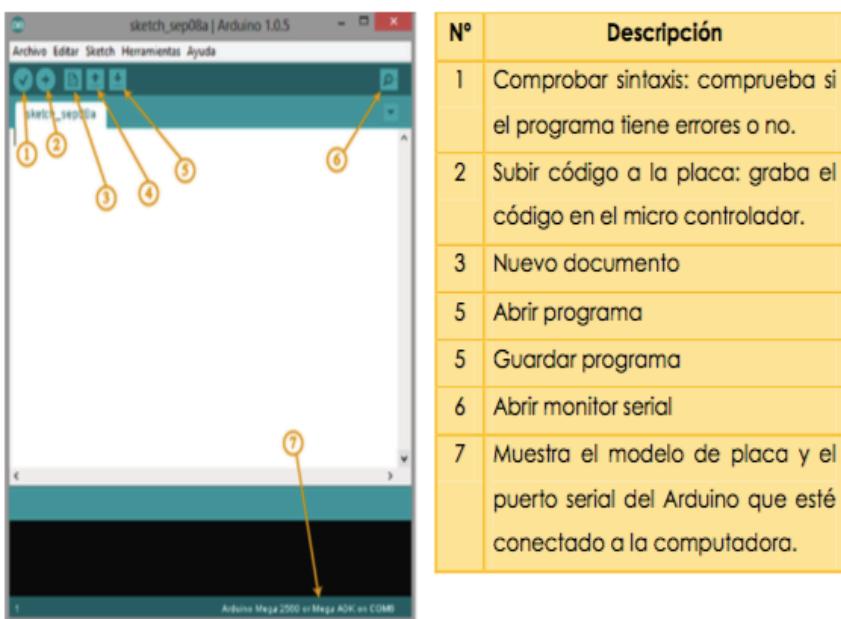
Cuando se instala Arduino en una computadora, automáticamente se le asigna un puerto serie, denotado por COM y un número cualquiera (ejemplo: COM2, COM3, COM15, etc.). Arduino se comunica con la computadora a través del puerto serie, a pesar de estar conectado por medio de USB. La nomenclatura de los dispositivos conectados al puerto serie utiliza el prefijo COM.

En el menú herramientas estará disponible los menús desplegables "Tarjeta" y "Puerto Serial".

Curso Arduino por Claudio & Karen



La barra de herramientas de Arduino IDE posee algunos accesos directos muy útiles para cuando se trabaja con Arduino.



5

Curso Arduino por Claudio & Karen

∞ CONOCIENDO EL LENGUAJE DE PROGRAMACIÓN ARDUINO

→ Funciones principales

Todo programa escrito en Arduino consta de dos funciones principales. Ambas son de tipo void (no devuelven un valor) y se denominan de la siguiente manera:

```
void setup(){  
}  
void loop(){  
}
```

La función **setup** se ejecuta una sola vez mientras que el **loop** se ejecuta un número infinito de veces.

En el "setup" normalmente se establece las funciones que llevarán a cabo los pines de Arduino, se inicia la comunicación serial, se establecen configuraciones y métodos que solamente necesitan ejecutarse una única vez.

En la función **loop** se establece el comportamiento que tendrá Arduino en tiempo de ejecución. El loop se ejecutará una tras otra vez, de principio a fin.

Todo programa debe contar con estas dos funciones, de lo contrario no será posible ejecutar el mismo.

A continuación se muestra una lista de los comandos más utilizados para programar e Arduino.

Comando	Descripción
pinMode(pin, modo);	Con esta función se establece el comportamiento de los pines digitales de Arduino. En el parámetro "pin" se escribe el número del pin a configurar; en "modo" se determina si el pin será de entrada (INPUT) o salida (OUTPUT). Ejemplo: pinMode(13, OUTPUT); el pin 13 se ha configurado como salida de voltaje.

Curso Arduino por Claudio & Karen

<code>digitalWrite(pin, estado);</code>	Se establece el estado que adoptará un pin determinado. En "pin" se coloca el número del pin y en "estado" si se requiere un HIGH o un LOW. Para poder usar esta función hay que establecer previamente el pin que se usará utilizando la función <code>pinMode()</code> . Ejemplo: <code>digitalWrite(13, HIGH);</code> el pin 13 entrará en estado "HIGH", es decir, entregará 5 voltios.
<code>Serial.begin(baudRate);</code>	Inicia la comunicación serial. Hace posible la comunicación entre el usuario y el micro controlador por medio de la computadora. En " <u>baudRate</u> " se establece la velocidad de transmisión de datos, siendo 9600 el valor más utilizado comúnmente. Ejemplo: <code>Serial.begin(9600);</code>
<code>Serial.println(String);</code>	Con esta función se puede enviar (imprimir) mensajes por medio del puerto serial. En " <u>String</u> " se coloca el mensaje que se desea enviar. Lo que sea que se envíe podrá ser visto por el usuario utilizando un monitor serial. Ejemplo: <code>Serial.println("hola a todos");</code> en el puerto serial aparecerá el mensaje "hola a todos".
<code>Serial.available();</code>	Devuelve un valor entero que será mayor que cero si es que hay un carácter disponible para ser leído en el puerto serie.
<code>Serial.read();</code>	Permite a Arduino leer un mensaje que se le haya enviado por medio del puerto serie. Es la forma como el usuario le da órdenes a Arduino.
<code>delay(tiempo);</code>	Provoca un retraso en la ejecución del código. El tiempo se da en milisegundos. Ejemplo: <code>delay(1000);</code> provoca un retraso de mil milisegundos, es decir, un segundo.

Curso Arduino por Claudio & Karen

<code>millis();</code>	Devuelve la cantidad de milisegundos que han pasado desde que se inició la ejecución del programa.
<code>micros();</code>	Devuelve la cantidad de microsegundos que han transcurrido desde que se inició la ejecución del programa.
<code>digitalRead(pin);</code>	Lee el estado de un pin digital, sea HIGH o LOW. Para poder usar esta función hay que declarar el "pin" previamente como INPUT. Ejemplo: <code>digitalRead(10);</code> devuelve el estado del pin 10: si hay voltaje aplicado a dicho pin el estado es HIGH, de lo contrario será LOW.
<code>analogWrite(pin, nivel);</code>	Esta función permite usar PWM (modulación por ancho de pulsos). Establece un nivel análogo de voltaje. En "pin" se coloca el pin que se usará (debe ser capaz de manejar PWM). En "nivel" se coloca un número entre 0 y 255, siendo 0 el 0% y 255 el 100%. Ejemplo: <code>analogWrite(10, 127);</code> el pin 10 estará entregando voltaje al 50% del ciclo de trabajo.
<code>analogRead(pin);</code>	Se utiliza con los pines análogos, entre A0 y A7. Lee el nivel análogo de voltaje en el pin seleccionado, el cual no debe ser mayor a 5 voltios. Es el "voltímetro" integrado que posee Arduino. Ejemplo: <code>analogRead(A0);</code> lee el voltaje aplicado al pin A0 (no debe ser mayor a 5 voltios).
<code>pulseIn(pin, estado);</code>	Devuelve el tiempo en milisegundos que dura un pulso, sea HIGH o LOW. En "pin" se establece el pin que se utilizará (debe haber sido configurado como INPUT usando <code>pinMode</code>) y en "estado" si el pulso a leer es HIGH o LOW. Ejemplo: <code>pulseIn(10, HIGH);</code> devuelve el tiempo que dura el pulso de voltaje aplicado al pin 10.

Curso Arduino por Claudio & Karen

→ TIPOS DATOS

Dato	Descripción
boolean	Dato booleano; puede ser TRUE O FALSE
char	Dato tipo carácter
byte	Almacena un número sin signo de 8-bit, desde 0 hasta 255.
int	Son el principal tipo de datos para almacenar números, y guardan valores de 2 bytes . Esto produce un rango entre -32,768 hasta 32,767
unsigned int	Son los mismos enteros de modo que almacenan un valor de dos bytes. En lugar de almacenar números negativos, sólo almacenan valores positivos, generando un rango útil desde 0 a 65,535
long	Son variables de tamaño extendido para almacenamiento de números, y 32 bits (4 bytes), desde -2,147,483,648 hasta 2,147,483,647.
unsigned long	Son variables extendidas para almacenar números, y almacenar 32 bits (4 bytes). Por el contrario que las variables long estándar, las unsigned long no almacenan números negativos, haciendo que su rango sea de 0 a 4,294,967,295
float	El tipo variable para los números en coma flotante (número decimal). Estos números son usados, habitualmente, para aproximar valores analógicos y continuos, debido a que ofrecen una mayor resolución que los enteros. Las variables tipo float tienen el valor máximo 3.4028235E+38, y como mínimo pueden alcanzar el -3.4028235E+38.
double	Número en coma flotante de doble precisión. Ocupa 4 bytes.
String	Los strings se representan como arrays de caracteres (tipo char) que terminan con el carácter NULL.

Curso Arduino por Claudio & Karen

array	Una matriz o "array" es una colección de variables que son accedidas mediante un número de índice. Los "arrays" en el lenguaje de programación C, en el cual está basado Arduino, pueden ser complicadas, pero usar "arrays" simples es relativamente sencillo.
void	La palabra reservada void se usa sólo en la declaración de funciones. Indica que se espera que no devuelva información a la función donde fue llamada.

→ Operadores Aritméticos

Operador	Función
+	Suma
-	Resta
*	Multiplicación
/	División (parte entera)
[^]	Potencia
=	Asignación
%	División (parte decimal)

→ Operadores Lógicos

Operador	Función
&&	Y
	O
!	Negación

→ Comparadores

Signo	Función
==	Igual a
!=	Diferente de
<	Menor que
>	Mayor que
<=	Menor o igual a
>=	Mayor o igual a

→ Estructuras de control

Estructura	Función
If (condición) { instrucciones }	Estructura comparativa simple. Si se cumple la "condición" se ejecutarán las instrucciones que se coloquen entre las llaves que abren y cierran la estructura.
If (condición) { Instrucciones A } else { Instrucciones B }	Estructura comparativa doble. Si se cumple un la primera condición entonces se ejecutará el bloque de <u>instrucciones A</u> . Si la condición inicial no se cumple entonces se procede a ejecutar el bloque de <u>instrucciones B</u> . Existe la posibilidad de agregar más condiciones y más bloques de instrucciones que permiten contemplar un número determinado de posibilidades.
For (int i=numero 1; i<= numero 2; i++) {	Se utiliza una variable contadora, en este caso i. Se repetirá un mismo proceso un número finito de veces, desde <u>que i es igual</u>

}	a un número inicial hasta que i alcanza un valor final. En cada ciclo se aumenta el valor de i en una unidad.
Switch (variable){ case condición 1: Instrucciones A case condición 2: Instrucciones B }	Estructura selectiva que permite escoger entre un número determinado de opciones dependiendo del valor que adopte la variable. Si el valor de la variable cumple con la condición 1, entonces se ejecuta el bloque de instrucciones A; si la variable cumple con la condición 2 entonces se ejecuta el bloque de instrucciones B. Las condiciones pueden ser múltiples.
While (condición) { Instrucciones }	Estructura condicione. Mientras se esté cumpliendo una condición se ejecutarán las instrucciones entre las llaves de la estructura un número infinito de veces.
Break;	Rompe un bloque de código o estructura.
Continue;	Continúa con la ejecución de un bloque de código.
return variable;	Devuelve el valor de una variable. Se utiliza con funciones.

En todo programa en Arduino se debe incluir punto y coma al final de cada línea. Si se colocan dos slash (//) frente a una línea automáticamente se considerará dicha sentencia como un comentario y no formará parte de la ejecución del software.

Se pueden implementar librerías utilizando la llamada include. Ejemplo:

#include <librería.h>

Arduino posee una colección de ejemplos sencillos y fáciles de utilizar, los cuales están disponibles en el menú Archivo del Arduino IDE. Están clasificados por categoría e incluyen fragmentos de código que le permite a los novatos acostumbrarse al entorno de desarrollo de Arduino.

Los programas normalmente no ocupan mucho espacio, todo depende del tipo de proyecto. A continuación un sencillo ejemplo de un programa en Arduino.

Curso Arduino por Claudio & Karen

El próximo código permitirá que se conecte un LED al pin 13, el cual empezará a parpadear permaneciendo un segundo (1000 milisegundos) encendido y luego 1 segundo apagado.

```
//primero se declara el pin donde se colocará el LED
int led = 13;

void setup() {
    // se le indica a Arduino que el pin 13 será Output, es decir, entregará voltaje
    pinMode(led, OUTPUT);
}

// En el loop se coloca el código que se ejecutará una tras otra vez infinitamente
void loop() {
    digitalWrite(led, HIGH); //Se enciende el LED
    delay(1000);           // se espera un segundo
    digitalWrite(led, LOW); // Se apaga el LED
    delay(1000);           // Se espera un segundo
}
```

Curso Arduino por Claudio & Karen

REFERENCIAS

Para más información...

- ✓ Sobre Arduino
 - <http://www.arduino.cc/es/>
 - <http://es.wikipedia.org/wiki/Arduino>
- ✓ Sobre el lenguaje de programación Arduino
 - <http://arduino.cc/es/Reference/HomePage>