Arduino

De Wikipedia, la enciclopedia libre

Arduino (**Genuino** a nivel internacional hasta octubre 2016), es una compañía de hardware libre y una comunidad tecnológica que diseña y manufactura placas de desarrollo de hardware, compuestas por Microcontroladores, elementos pasivos y activos . Por otro lado las placas son programadas a través de unentorno de desarrollo (IDE), el cuál compila el código al modelo seleccionada de placa.

Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios.[1] [2] Toda la plataforma, incluyendo sus componentes de hardware (esquemáticos) y Software, son liberados con licencia de código abierto que permite libertad de acceso a ellos..[3]

El *hardware* consiste en una placa de circuito impreso con un microcontrolador, usualmente Atmel AVR, puertos digitales y analógicos de entrada/salida,[4] los cuales pueden conectarse a placas de expansión (shields), que amplían las características de funcionamiento de la placa Arduino. Asimismo, posee un puerto de conexión USB desde donde se puede alimentar la placa y establecer comunicación con el computador.

Por otro lado, el *software* consiste en un entorno de desarrollo (IDE) basado en el entorno de *processing* y lenguaje de programación basado en Wiring, así como en el cargador de arranque (*bootloader*) que es ejecutado en la placa.[4] El microcontrolador de la placa se programa mediante un computador, usando una comunicación serial mediante un convertidor de niveles RS-232 a TTL serial.

Arduino

Tipo	Placa computadora (microcontrolador de placa simple)
CPU	AVR, ARM Cortex, Intel Quark
Memoria	SRAM
Capacidad de almacenamiento	Memoria Flash, EEPROM
Página web	www.arduino.cc (http://www.arduino.cc) y www.arduino.org (http://www.arduino.org)

[editar datos en Wikidata]

La primera placa Arduino fue introducida en 2005, ofreciendo un bajo costo y facilidad de uso para novatos y profesionales. Buscaba desarrollar proyectos interactivos con su entorno mediante el uso de actuadores y sensores. A partir de octubre de 2012, se incorporaron nuevos modelos de placas de desarrollo que usan microcontroladores Cortex M3, ARM de 32 bits,[5] que coexisten con los originales modelos que integran microcontroladores AVR de 8 bits. ARM y AVR no son plataformas compatibles en cuanto a su arquitectura y por lo que tampoco lo es su set de instrucciones, pero se pueden programar y compilar bajo el IDE predeterminado de Arduino sin ningún cambio.

Las placas Arduino están disponibles de dos formas: ensambladas o en forma de kits "Hazlo tú mismo" (por sus siglas en inglés "DIY"). Los esquemas de diseño del Hardware están disponibles bajo licencia Libre, con lo que se permite que cualquier persona pueda crear su propia placa Arduino sin necesidad de comprar una prefabricada. Adafruit Industries estimó a mediados del año 2011 que, alrededor de 300 000 placas Arduino habían sido producidas comercialmente y en el año 2013 estimó que alrededor de 700 000 placas oficiales de la empresa Arduino estaban en manos de los usuarios.

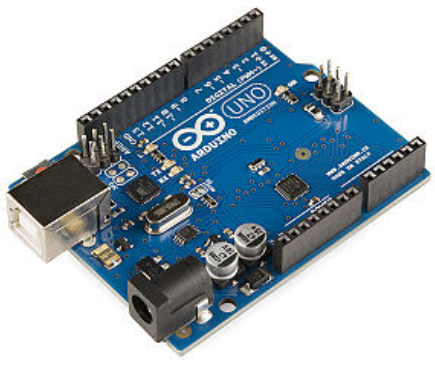
Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software tal como Adobe Flash, Processing, Max/MSP, Pure Data, etc. Una tendencia tecnológica es utilizar Arduino como tarjeta de adquisición de datos desarrollando interfaces en software como JAVA, Visual Basic y LabVIEW.[6] Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

El proyecto Arduino recibió una mención honorífica en la categoría de Comunidades Digitales en el Prix Ars Electrónica de 2006.[7] [8] [9]

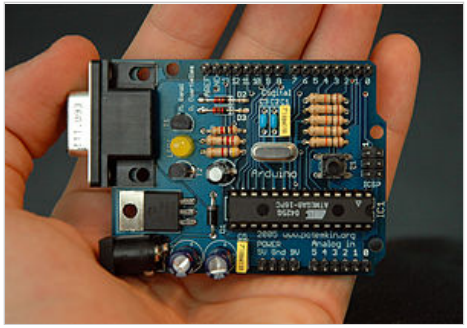
Arduino como herramienta educativa es muy útil y efectiva. Existen diferentes web con recursos, tutoriales, trucos, ejercicios… Existen tutoriales oficiales de Arduino (https://www.arduino.cc/en/Tutorial/HomePage).

Arduino tiene una gran comunidad a su alrededor donde puedes encontrar material de calidad y muy útil, desde tutoriales para iniciarse desde cero hasta aquellos destinados a usuarios más avanzados.

Índice



"Arduino Uno" Revisión 3



Placa Arduino RS232[10]

Historia

Arduino se inició en el año 2005 como un proyecto para estudiantes en el Instituto IVREA, en Ivrea (Italia). En ese tiempo, los estudiantes usaban el microcontrolador BASIC Stamp, cuyo coste era de 100 dólares estadounidenses, lo que se consideraba demasiado costoso para ellos. Por aquella época, uno de los fundadores de Arduino, Massimo Banzi, daba clases en Ivrea.[11]

El nombre del proyecto viene del nombre del *Bar di Re Arduino* (Bar del Rey Arduino) donde Massimo Banzi pasaba algunas horas. El rey Arduino fue rey de Italia entre los años 1002 y 1014. En la creación de este proyecto contribuyó el estudiante colombiano Hernando Barragán, quien desarrolló la tarjeta electrónica Wiring, el lenguaje de programación y la plataforma de desarrollo.[12] Una vez concluida dicha plataforma, los investigadores trabajaron para hacerlo más ligero, más económico y disponible para la comunidad de código abierto (*hardware* y código abierto). El instituto finalmente cerró sus puertas, así que los investigadores, entre ellos el español David Cuartielles, promovieron la idea.[11] Banzi afirmaría años más tarde, que el proyecto nunca surgió como una idea de negocio, sino como una necesidad de subsistir ante el inminente cierre del Instituto de diseño Interactivo IVREA. Es decir, que al crear un producto de *hardware* abierto, este no podría ser embargado.

Posteriormente, Google colaboró en el desarrollo del Kit Android ADK (Accesory Development Kit), una placa Arduino capaz de comunicarse directamente con teléfonos móviles inteligentes bajo el sistema operativo Android para que el teléfono controle luces, motores y sensores conectados de Arduino.[13] [14]

Para la producción en serie de la primera versión se tomó en cuenta que el coste no fuera mayor de 30 euros, que fuera ensamblado en una placa de color azul, debía ser Plug and Play y que trabajara con todas las plataformas informáticas tales como MacOSX, Windows y GNU/Linux. Las primeras 300 unidades se las dieron a los alumnos del Instituto IVREA, con el fin de que las probaran y empezaran a diseñar sus primeros prototipos.

En el año 2005, se incorporó al equipo el profesor Tom Igoe,[12] que había trabajado en computación física, después de que se enterara del mismo a través de Internet. Igoe ofreció su apoyo para desarrollar el proyecto a gran escala y hacer los contactos para distribuir las tarjetas en territorio estadounidense. En la feria Maker Fair de 2011 se presentó la primera placa Arduino 32 bit para realizar tareas más pesadas.[15]

Hardware

Los modelos de Arduino se categorizan en placas de desarrollo, placas de expansión (*shields*), kits, accesorios e impresoras 3d.

Placas
Arduino Galileo,[16] Arduino Uno, Arduino Leonardo, Arduino Due, Arduino Yún, Arduino Tre (En Desarrollo), Arduino Zero, Arduino Micro, Arduino Esplora, Arduino Mega ADK, Arduino Ethernet, Arduino Mega 2560, Arduino Robot, Arduino Mini, Arduino Nano, LilyPad Arduino Simple, LilyPad Arduino SimpleSnap, LilyPad Arduino, LilyPad Arduino USB, Arduino Pro Mini, Arduino Fio, Arduino Pro, Arduino MKR1000/Genuino MKR1000, Arduino MICRO/Genuino MICRO, Arduino 101/Genuino 101, Arduino Gemma.

Placas de expansión (*shields*)
Arduino GSM Shield, Arduino Ethernet Shield, Arduino WiFi Shield, Arduino Wireless SD Shield, Arduino USB Host Shield, Arduino Motor Shield, Arduino Wireless Proto Shield, Arduino Proto Shield.

Kits
The Arduino Starter Kit, Arduino Materia 101.

Accesorios
TFT LCD Screen, USB/Serial Light Adapter, Arduino ISP, Mini USB/Serial Adapter.

Impresoras 3d
Arduino Materia 101.

Equipo de desarrollo

El núcleo del equipo de desarrollo de Arduino está formado por Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David Mellis y Nicholas Zambetti.

Aplicaciones

La plataforma Arduino ha sido usada como base en diversas aplicaciones electrónicas:

- Xoscillo: Osciloscopio de código abierto[17]
- Equipo científico para investigaciones[18]
- Arduinome: Un dispositivo controlador MIDI[19]
- OBDuino: un económetro que usa una interfaz de diagnóstico a bordo que se halla en los automóviles modernos
- SCA-ino: Sistema de cómputo automotriz capaz de monitorear sensores como el TPS, el MAP y el 02S y controlar actuadores automotrices como la bobina de ignición, la válvula IAC y aceleradores electrónicos
- Humane Reader: dispositivo electrónico de bajo coste con salida de señal de TV que puede manejar una biblioteca de 5000 títulos en una tarjeta microSD[20]
- The Humane PC: equipo que usa un módulo Arduino para emular un computador personal, con un monitor de televisión y un teclado para computadora[21]
- Ardupilot: software y hardware de aeronaves no tripuladas
- ArduinoPhone: un teléfono móvil construido sobre un módulo Arduino[22] [23]
- Máquinas de control numérico por computadora (CNC)
- Open Theremín Uno: Versión digital de hardware libre del instrumento Theremín
- Impresoras 3D

Esquema de conexiones

Entradas y salidas

Poniendo de ejemplo al módulo Diecimila, este consta de 14 entradas digitales configurables como entradas y/o salidas que operan a 5 voltios. Cada contacto puede proporcionar o recibir como máximo 40 mA. Los contactos 3, 5, 6, 9, 10 y 11 pueden proporcionar una salida PWM (Pulse Width Modulation). Si se conecta cualquier cosa a los contactos 0 y 1, eso interferirá con la comunicación USB. Diecimila también tiene 6 entradas analógicas que proporcionan una resolución de 10 bits. Por defecto, aceptan de 0 hasta 5 voltios (aunque es posible cambiar el nivel más alto utilizando el contacto Aref y algún código de bajo nivel).

Especificaciones

Las especificaciones de los distintos modelos de placas Arduino se resumen en la siguiente tabla:

Modelo	Microcontrolador	Voltaje de entrada	Voltaje del sistema	Frecuencia de reloj	Entradas/salidas digitales	Entradas analógicas	PWM	UART	Memoria flash	Cargador	Interfaz de programación

Los modelos Arduino Diecimila, Arduino Duemilanove y Arduino Mega están basados en los microcontroladores ATmega168, ATmega328 y ATmega1280

	ATmega168	ATmega328	ATmega1280
Voltaje operativo	5 V	5 V	5 V
Voltaje de entrada recomendado	7-12 V	7-12 V	7-12 V
Voltaje de entrada límite	6-20 V	6-20 V	6-20 V
Entradas y salidas digitales	14 (6 proporcionan PWM)	14 (6 proporcionan PWM)	54 (14 proporcionan PWM)
Entradas analógicas	6	6	16
Intensidad de corriente	40 mA	40 mA	40 mA
Memoria Flash	16KB (2KB reservados para el bootloader)	32KB (2KB reservados para el bootloader)	128KB (4KB reservados para el bootloader)
SRAM	1 KB	2 KB	8 KB
EEPROM	512 bytes	1 KB	4 KB
Frecuencia de reloj	16 MHz	16 MHz	16 MHz

Lenguaje de programación Arduino

La plataforma Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel Processing que es similar a C++.

Funciones básicas y operadores

Arduino está basado en C y soporta todas las funciones del estándar C y algunas de C++.[24] A continuación se muestra un resumen con la estructura y sintaxis del lenguaje Arduino:

Sintaxis básica

- Delimitadores:;, {}
- Comentarios: //, /* */
- Cabeceras: #define, #include
- Operadores aritméticos: +, -, *, /, %
- Asignación: =
- Operadores de comparación: ==, !=, <, >, <=, >=
- Operadores Booleanos: &&, ||, !
- Operadores de acceso a punteros: *, &
- Operadores de bits: &, |, ^, ~, <<, >>
- Operadores compuestos:
  - Incremento y decremento de variables: ++, --
  - Asignación y operación: +=, -=, *=, /=, &=, |=

Estructuras de control

- Condicionales: if, if...else, switch case
- Bucles: for, while, do. while
- Bifurcaciones y saltos: break, continue, return, goto

Variables

En cuanto al tratamiento de las variables también comparte un gran parecido con el lenguaje C.

Constantes

- HIGH/LOW: representan los niveles alto y bajo de las señales de entrada y salida. Los niveles altos son aquellos de 3 voltios o más.
- INPUT/OUTPUT: entrada o salida.
- false (falso): Señal que representa al cero lógico. A diferencia de las señales HIGH/LOW, su nombre se escribe en letra minúscula.
- true (verdadero): Señal cuya definición es más amplia que la de *false*. Cualquier número entero diferente de cero es "verdadero", según el álgebra de Boole, como en el caso de -200, -1 o 1. Si es cero, es "falso".

Tipos de datos

- void, boolean, char, unsigned char, byte, int, unsigned int, word, long, unsigned long, float, double, string, array.

Conversión entre tipos

Estas funciones reciben como argumento una variable de cualquier tipo y devuelven una variable convertida en el tipo deseado.

- char(), byte(), int(), word(), long(), float()

Cualificadores y ámbito de las variables

- static, volatile, const.

Utilidades

- sizeof()

Funciones básicas

E/S digital

- pinMode(pin, modo).
- digitalWrite(pin, valor).
- int digitalRead(pin).

E/S analógica

- analogReference(tipo)
- int analogRead(pin)
- analogWrite(pin, valor)

E/S avanzada

- shiftOut(dataPin, clockPin, bitOrder, valor)
- unsigned long pulseIn(pin, valor)

Tiempo

- unsigned long millis()
- unsigned long micros()
- delay(ms)
- delayMicroseconds(microsegundos)

Matemáticas

- min(x, y), max(x, y), abs(x), constrain(x, a, b), map(valor, fromLow, fromHigh, toLow, toHigh), pow(base, exponente), sqrt(x)

Trigonometría

- sin(rad), cos(rad), tan(rad)

Números aleatorios

- randomSeed(semilla), long random(máx), long random(mín, máx)

Bits y Bytes

- lowByte(), highByte(), bitRead(), bitWrite(), bitSet(), bitClear(), bit()

Interrupciones externas

- attachInterrupt(interrupción, función, modo)
- detachInterrupt(interrupción)

Interrupciones

- interrupts(), noInterrupts()

Comunicación por puerto serie

Las funciones de manejo del puerto serie deben ir precedidas de la palabra "Serial" aunque no necesitan ninguna declaración en la cabecera del programa. Por esto se consideran funciones base del lenguaje.[25] Estas son las funciones para transmisión serial:

- begin(), available(), read(), flush(), print(), println(), write()

Manipulación de puertos

Los registros de puertos permiten la manipulación a más bajo nivel y de forma más rápida de los contactos de entrada/salida del microcontrolador de las placas Arduino.[26] Los contactos eléctricos de las placas Arduino están repartidos entre los registros B(0-7), C (analógicos) y D(8-13). Mediante estas variables se observa y se modificada su estado:

- DDR[B/C/D]: Data Direction Register (o dirección del registro de datos) del puerto B, C ó D. Es una variable de Lectura/Escritura que sirve para especificar cuales contactos serán usados como entrada y salida.
- PORT[B/C/D]: Data Register (o registro de datos) del puerto B, C ó D. Es una variable de Lectura/Escritura.
- PIN[B/C/D]: Input Pins Register (o registro de pines de entrada) del puerto B, C ó D. Variable de sólo lectura.

Por ejemplo, para especificar los contactos 9 a 13 como salidas y el 8 como entrada (puesto que el puerto D usa los pines de la placa Arduino 8 al 13 digitales) bastaría utilizar la siguiente asignación:

```
DDRD = B11111110;
```

Como se ha podido comprobar, el conocimiento del lenguaje C, permite la programación en Arduino debido a la similitud entre este y el lenguaje nativo del proyecto, lo que implica el aprendizaje de algunas funciones específicas de que dispone el lenguaje del proyecto para manejar los diferentes parámetros. Se pueden construir aplicaciones de cierta complejidad sin necesidad de muchos conceptos previos.

AVR Libc

Los programas compilados con Arduino (salvo en las placas con CorteX M3) se enlazan contra AVR Libc[24] por lo que tienen acceso a algunas de sus funciones. AVR Libc es un proyecto de software libre con el objetivo de proporcionar una biblioteca C de alta calidad para utilizarse con el compilador GCC sobre microcontroladores Atmel AVR. Se compone de 3 partes:

- avr-binutils
- avr-gcc
- avr-libc

La mayoría del lenguaje de programación Arduino está escrita con constantes y funciones de AVR y ciertas funcionalidades sólo se pueden obtener haciendo uso de AVR.[27]

Interrupciones

Las señales de interrupción son las siguientes:

- cli(): desactiva las interrupciones globales
- sei(): activa las interrupciones

Esto afectará al temporizador y a la comunicación serial. La función delayMicroseconds() desactiva las interrupciones cuando se ejecuta.

Temporizadores

La función delayMicroseconds() crea el menor retardo posible del lenguaje Arduino que ronda los 2µs. Para retardos más pequeños se debe utilizar la llamada de ensamblador 'nop' (no operación). Cada sentencia 'nop' se ejecutará en un ciclo de máquina (16 MHz) de aproximadamente 62,5ns.

Manipulación de puertos

La manipulación de puertos con código AVR es más rápida que utilizar la función digitalWrite() de Arduino.

Establecer Bits en variables

cbi y sbi son mecanismos estándar (AVR) para establecer o limpiar bits en PORT y otras variables.

Diferencias con Processing

La sintaxis del lenguaje de programación Arduino es una versión simplificada de C/C++ y tiene algunas diferencias respecto de Processing.[28] [29] Debido a que Arduino está basado en C/C++ mientras que Processing se basa en Java, existen varias diferencias en cuanto a la sintaxis de ambos lenguajes y el modo en que se programa:

Arreglos

Arduino	Processing
`int bar[8];` `bar[0] = 1;`	`int[] bar = new int[8];` `bar[0] = 1;`

`int foo[] = { 0, 1, 2 };`	`int foo[] = { 0, 1, 2 };`
	o bien
	`int[] foo = { 0, 1, 2 };`

Impresión de cadenas

Arduino	Processing
`Serial.println("hello world");`	`println("hello world");`
`int i = 5;` `Serial.println(i);`	`int i = 5;` `println(i);`

`int i = 5;`
`Serial.print("i = ");`
`Serial.print(i);`
`Serial.println();`
	`int i="5";`
	`println("i =" + i);`

Ejemplo sencillo de programación en Arduino

El primer paso antes de comprobar que la instalación es correcta y empezar a trabajar con Arduino, es usar ejemplos prácticos que vienen disponibles con el dispositivo. Se recomienda abrir el ejemplo “led_blink” el cual crea una intermitencia por segundo en un led conectado en el pin 13.

El código necesario es el siguiente:

```
# define LED_PIN 13
void setup () {
 // Activado del contacto 13 para salida digital
 pinMode (LED_PIN, OUTPUT);
}
// Bucle infinito
void loop () {
 // Encendido del diodo LED enviando una señal alta
 digitalWrite (LED_PIN, HIGH);
 // Tiempo de espera de 1 segundo (1000 ms)
 delay (1000);
 // Apagado del diodo LED enviando una señal baja.
 digitalWrite (LED_PIN, LOW);
 // Tiempo de espera de 1 segundo
 delay (1000);
}
```

Bibliotecas en Arduino

Las bibliotecas estándar que ofrece Arduino son las siguientes:[30]

Serial

Lectura y escritura por el puerto serie.

EEPROM

Lectura y escritura en el almacenamiento permanente.[31]

- read(), write()

Ethernet

Conexión a Internet mediante “Arduino Ethernet Shield“. Puede funcionar como servidor que acepta peticiones remotas o como cliente. Se permiten hasta cuatro conexiones simultáneas.[32] Los comandos usados son los siguientes:

- Servidor: Server(), begin(), available(), write(), print(), println()
- Cliente: Client(), connected(), connect(), write(), print(), println(), available(), read(), flush(), stop()

Firmata

Es una biblioteca de comunicación con aplicaciones informáticas utilizando el protocolo estándar del puerto serie.[33]

LiquidCrystal

Control de LCDs con chipset Hitachi HD44780 o compatibles.[34] La biblioteca soporta los modos de 4 y 8 bits.

Servo

Biblioteca para el control de servo motores.[35] A partir de la versión 0017 de Arduino la biblioteca soporta hasta 12 motores en la mayoría de las placas Arduino y 48 en la Arduino Mega. Estos son los comandos usados:

- attach(), write(), writeMicroseconds(), read(), attached(), detach()

SoftwareSerial

Comunicación serie en contactos digitales.[36] Por defecto Arduino incluye comunicación sólo en los contactos 0 y 1 pero gracias a esta biblioteca puede realizarse esta comunicación con los restantes.

Stepper

Control de motores paso a paso unipolares o bipolares.[37]

- Stepper(steps, pin1, pin2), Stepper(steps, pin1, pin2, pin3, pin4), setSpeed(rpm), step(steps)

Wire

Envío y recepción de datos sobre una red de dispositivos o sensores mediante Two Wire Interface (TWI/I2C).[38]

Las bibliotecas *Matrix* y *Sprite* de Wiring son totalmente compatibles con Arduino y sirven para manejo de matrices de diodos LED. También se ofrece información sobre diversas bibliotecas desarrolladas por diversos colaboradores que permiten realizar muchas tareas.

Creación de bibliotecas

Los usuarios de Arduino tienen la posibilidad de escribir sus propias bibliotecas.[39] Ello permite disponer de código que puede reutilizarse en otros proyectos, mantener el código fuente principal separado de las bibliotecas y la organización de los programas construidos es más clara.

Ejemplo de biblioteca

El siguiente ejemplo permite el envío de caracteres mediante el código Morse:

Se crea el archivo Morse.h que incluye la definición de la clase Morse que tiene 3 funciones: un constructor (Morse()), una función para enviar 1 punto (dot()) y una función para enviar una raya (dash()). La variable _pin permite indicar el contacto a usar.

```
/*
  Morse.h - Biblioteca para el envío de Código Morse.
  Creado por David A. Mellis, el 2 de noviembre de 2007.
  Liberado al dominio público.
*/

# ifndef Morse_h
# define Morse_h

# include "WProgram.h"

class Morse
{
  public:
    Morse(int pin);
    void dot();
    void dash();
  private:
    int _pin;
};

# endif
```

Debe ser creado el archivo Morse.cpp con el código, es decir con la implementación de los métodos declarados:

```
/*
  Morse.cpp - Biblioteca para el envío de Código Morse.
  Creado por David A. Mellis, el 2 de noviembre de 2007.
  Liberado al dominio público.
*/

# include "WProgram.h"
```

```
# include "Morse.h"

Morse::Morse(int pin)
{
  pinMode(pin, OUTPUT);
  _pin = pin;
}

void Morse::dot()
{
  digitalWrite(_pin, HIGH);
  delay(250);
  digitalWrite(_pin, LOW);
  delay(250);
}

void Morse::dash()
{
  digitalWrite(_pin, HIGH);
  delay(1000);
  digitalWrite(_pin, LOW);
  delay(250);
}
```

La biblioteca creada así puede ser usada mediante el comando #include. Si se desea enviar una petición de auxilio SOS por el contacto 13 bastaría con llamar a Morse(13) y ejecutar la siguiente secuencia:

```
morse.dot(); morse.dot(); morse.dot();
morse.dash(); morse.dash(); morse.dash();
morse.dot(); morse.dot(); morse.dot();
```

Ejemplos de Código

La página de Arduino cuenta con una serie de ejemplos para comenzar a entender su funcionamiento, con componentes base tales como Pantallas, LED's, Potenciometros, etc.

Ejemplo de parpadeo de LED

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Ejemplo de lectura de Potenciometro

```
/*
  ReadAnalogVoltage
  Reads an analog input on pin 0, converts it to voltage, and prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

ejemplo hacer contar un display de 7 segmentos de 1 a 3 cada 1 segundo, llamando a una función

```
void setup(){   // configuramos los pines de salida donde conectaremos los pines con una resistencia en serie al display de 7 segmentos
  pinMode(34, OUTPUT);  // a
  pinMode(36, OUTPUT);  // b
  pinMode(38, OUTPUT);  // c
  pinMode(40, OUTPUT);  // d
  pinMode(42, OUTPUT);  //e
  pinMode(44, OUTPUT);  //f
  pinMode(46, OUTPUT);  //g
 }
void display (int a, int b, int c, int d, int e, int f, int g)// Función del display
{
  digitalWrite (34,a);
  digitalWrite (36,b);
  digitalWrite (38,c);
  digitalWrite (40,d);
  digitalWrite (42,e);
  digitalWrite (44,f);
  digitalWrite (46,g);
}
 void loop(){
   display (0,1,1,0,0,0,0); //  mostrará 1 en el display
  delay(1000);
display (1,1,0,1,1,0,1); //  mostrará 2 en el display
  delay(1000);
display (1,1,1,1,0,0,1); //  mostrará 3 en el display
  delay(1000);  // por Laiolo Santiago

}
```

Instalación en diferentes entornos

Windows

Los pasos a seguir son los siguientes:

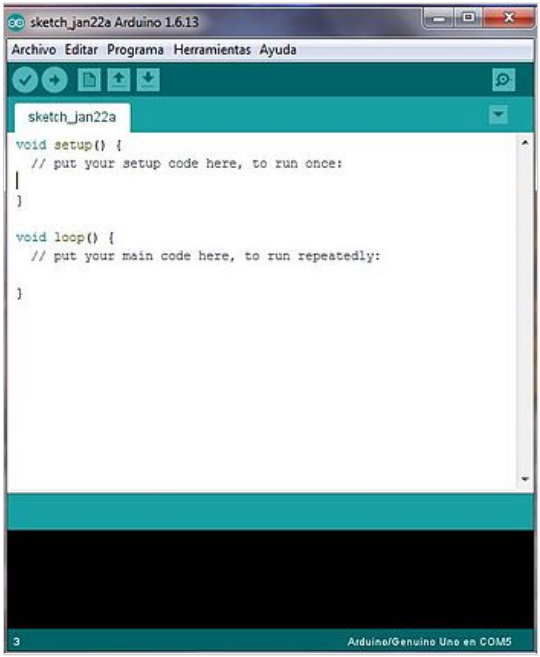
- Descargar las versiones más reciente de Java Runtime Enviroment (J2RE) y del IDE Arduino.
- Instalar los controladores FTDI USB, con la placa Arduino conectada.
- Ejecutar el IDE Arduino para abrir la interfaz y configurar el puerto USB donde está conectada la placa.

GNU/Linux

Para instalar Arduino en un sistema GNU/Linux necesitamos los siguientes programas para resolver las dependencias:

- Sun java runtime, jre.
- avr-gcc, compilador para la familia de microcontroladores avr de atmel.
- avr-libc, libc del compilador avr-gcc.

En algunas distribuciones conviene desinstalar, si no es necesario, el programa "brltty" que permite el acceso al terminal a personas invidentes. Para concluir, se descarga el framework de Arduino, se descomprime y ejecuta.



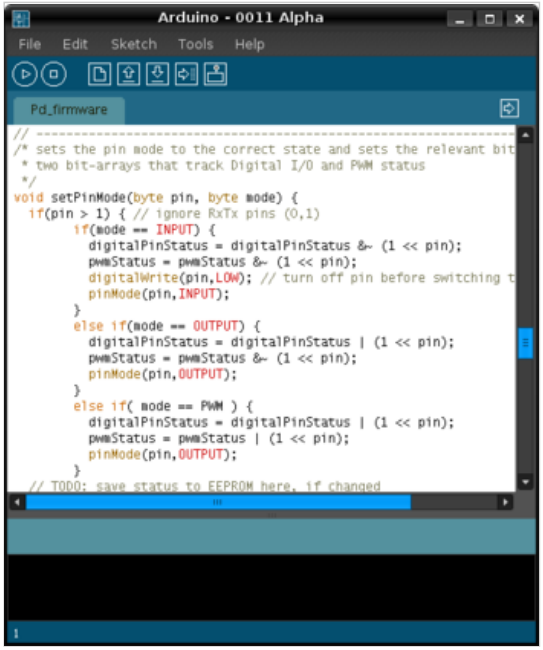
Interfaz del entorno de desarrollo Arduino en el Sistema Operativo Windows.

Otras interfaces de programación

Es posible comunicar una aplicación que corra sobre Arduino con otros dispositivos que corran otros lenguajes de programación y aplicaciones populares,[40] debido a que Arduino usa la transmisión serial de datos, la cuál es soportada por la mayoría de los lenguajes que se mencionan a continuación. Y para los que no soportan el formato serie de forma nativa, es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida. Algunos ejemplos de lenguajes son:

- 3DVIA Virtools: aplicaciones interactivas y de tiempo real.
- Adobe Director
- BlitzMax (con acceso restringido)
- C
- C++ (mediante libSerial o en Windows)
- C#
- Cocoa/Objective-C (para Mac OS X)
- Flash (mediante ActionScript)
- Gambas
- Isadora (Interactividad audiovisual en tiempo real)
- Instant Reality (X3D)
- Java
- Liberlab (software de medición y experimentación)
- Mathematica
- Matlab
- MaxMSP: Entorno gráfico de programación para aplicaciones musicales, de audio y multimedia
- Minibloq: Entorno gráfico de programación, corre también en las computadoras OLPC
- Perl

- Php
- Physical Etoys: Entorno gráfico de programación usado para proyectos de robótica educativa
- Processing
- Pure Data
- Python
- Ruby
- Scratch for Arduino (S4A): Entorno gráfico de programación, modificación del entorno para niños Scratch, del MIT
- Squeak: Implementación libre de Smalltalk
- SuperCollider: Síntesis de audio en tiempo real
- VBScript
- Visual Basic .NET
- VVVV: Síntesis de vídeo en tiempo real



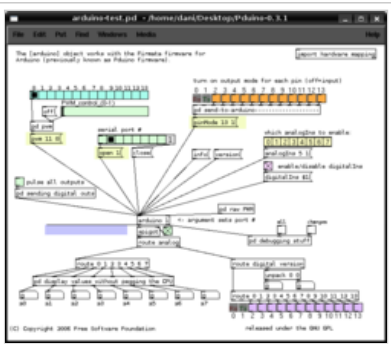
Interfaz del entorno de desarrollo Arduino S.O. GNU/Linux.

Pduino

Pduino nace de la fusión de los proyectos Pure Data y Arduino. Ambos proyectos de fuente abierta permiten trabajar con interfaz gráfica. Cargando el firmware de Pure Data (PD) a la placa Arduino se puede acceder a ella mediante el lenguaje de programación gráfico.

Minibloq

Minibloq es un entorno gráfico de programación que puede generar código nativo de Arduino y escribirlo directamente en la memoria flash de la placa. Tiene un modo que permite visualizar el código generado, el cual también puede ser copiado y pegado en el Arduino-IDE, para los usuarios que intentan hacer el pasaje de una herramienta gráfica a la programación en sintaxis C/C++. Minibloq es de uso libre y sus fuentes también están disponibles gratuitamente. Una característica importante, es que puede correr también en la computadora portátil OLPC, mediante el software Wine.


Patch Pduino.

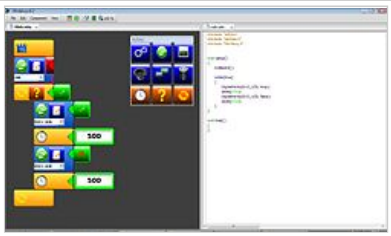
Physical Etoys

Physical Etoys es una extensión libre y gratuita que permite que diversos dispositivos electrónicos como Lego NXT, las placas Arduino, Sphero, Kinect, Joystick Wiimote, entre otros, puedan ser programados fácilmente y que interactúen entre sí gracias a su sistema de bloques.

En el caso de Arduino, Physical Etoys ofrece dos modos de programación, el modo "directo" y el modo "compilado".

Modo directo

El modo "directo", en el cual los programas se ejecutan en la computadora del usuario y las órdenes se transmiten inmediatamente a través del puerto serie. El modo "directo" permite modificar los programas y ver los cambios producidos de manera inmediata en el comportamiento del robot, lo cual facilita la programación, sobre todo al usuario inexperto. Asimismo, permite ver constantemente los valores de los sensores y utilizar el robot, por ejemplo, como para adquirir datos.


Pantalla de Minibloq.

Modo compilado

El modo "compilado", en el cual los programas se traducen a C++ y se bajan a la placa, para luego ejecutarse de manera independiente de la computadora. El modo "compilado", por su parte, elimina el retardo que introduce la comunicación con la computadora, lo cual lo hace preferible para el desarrollo de tareas autónomas, en las cuales la velocidad de respuesta del robot debe ser óptima.

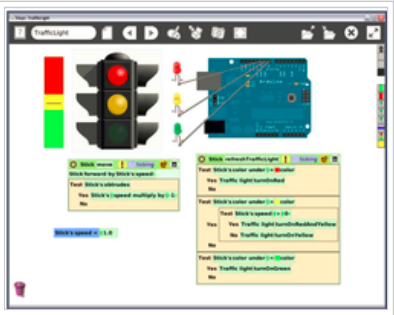

Combinación de una computadora de bajo costo OLPC, el software Minibloq y una placa Arduino.

Véase también

- BASIC Stamp
- Impresión 3D
- Gumstix
- Minibloq
- MiniPC
- OOPIC
- PICAXE
- Raspberry Pi
- Physical Etoys
- Robot
- X10
- Sanguino

Referencias

1. «Interview with Casey Reas and Ben Fry» (http://rhizome.org/editorial/2009/sep/23/interview-with-casey-reas-and-ben-fry/).
2. [1] (http://wiring.org.co/)
3. [2] (https://www.arduino.cc/en/Guide/Introduction#)
4. «Project homepage» (http://arduino.cc).
5. «Arduino Due is finally here» (http://arduino.cc/blog/2012/10/22/arduino-due-is-finally-here/).
6. https://www.youtube.com/watch?v=n3AwL-_UCS4
7. http://web.archive.org/web/http://www.aec.at/en/p
8. «Ars Electrónica Archiv» (http://90.146.8.18/de/archives/prix_archive/prix_year_cat.asp?iProjectID=13638&iCategoryID=12420) (en alemán). Consultado el 18 de febrero de 2009.
9. «Ars Electronica Archiv / ANERKENNUNG» (http://web.archive.org/web/http://90.146.8.18/de/archives/prix_archive/prix_projekt.asp?iProjectID=13789#) (en alemán). Archivado desde el original (http://90.146.8.18/de/archives/prix_archive/prix_projekt.asp?iProjectID=13789#) el 26 de noviembre de 2015. Consultado el 18 de febrero de 2009.
10. Placa Arduino Serial (http://arduino.cc/en/Main/ArduinoBoardSerial)
11. David Kushner (26 de octubre de 2011). «The Making of Arduino» (http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino). *IEEE Spectrum*.
12. «Algo de Historia para comenzar» (http://platea.pntic.mec.es/~lmarti2/arduino/introardu/index.htm). Consultado el 22 de diciembre de 2013.
13. «Google presenta ADK, interfaz basada en Arduino para Android» (http://blog.bricogeek.com/noticias/arduino/google-presenta-adk-interfaz-basada-en-arduino-para-android/). BricoGeek.com. 12 de mayo de 2011. Consultado el 22 de diciembre de 2013.
14. «Accessory Development Kit 2012 Guide» (http://developer.android.com/tools/adk/adk2.html) (en inglés). Consultado el 22 de diciembre de 2013.
15. Terrence O'Brien (19 de septiembre de 2011). «Arduino brings the (new) goods to Maker Faire New York, welcomes ARM into the fold» (http://www.engadget.com/2011/09/19/arduino-brings-the-new-goods-to-maker-faire-new-york-welcomes/) (en inglés). Consultado el 22 de diciembre de 2013.
16. https://www.arduino.cc/en/ArduinoCertified/IntelC
17. «Xoscillo: A software oscilloscope that acquires data using an Arduino or a parallax.» (http://code.google.com/p/xoscillo/) (en inglés). Consultado el 22 de diciembre de 2013.
18. Joshua M. Pearce (14 de septiembre de 2012). «Building Research Equipment with Free, Open-Source Hardware» (http://web.archive.org/web/http://211.144.68.84:9998/91keshi/Public/File/41/337-6100/pdf/1303.full.pdf) (en inglés). Washington, EE.UU.: American Association for the Advancement of Scienc. p. 3. doi:10.1126/science.1228183 (http://dx.doi.org/10.1126%2Fscience.1228183). Archivado desde el original (http://211.144.68.84:9998/91keshi/Public/File/41/337-6100/pdf/1303.full.pdf) el 26 de noviembre de 2015. Consultado el 22 de diciembre de 2013.
19. Peter Kirn (20 de agosto de 2008). «Aug 20 2008 Arduinome: An Arduino-Based Monome Clone, Behind the Scenes» (http://createdigitalmusic.com/2008/08/arduinome-an-arduino-based-monome-clone-behind-the-scenes/) (en inglés). Consultado el 22 de diciembre de 2013.
20. «Humane Reader» (http://humaneinfo.com/reader.html) (en inglés). Consultado el 22 de diciembre de 2013.
21. «The Humane PC» (http://humaneinfo.com/pc.html) (en inglés). Consultado el 22 de diciembre de 2013.
22. «ArduinoPhone» (http://www.instructables.com/id/ArduinoPhone/ArduinoPhone) (en inglés). Consultado el 22 de diciembre de 2013.
23. Esteban Zamorano (28 de noviembre de 2013). «Construye tu propio celular por USD$200 gracias a Arduino» (http://www.fayerwayer.com/2013/11/construye-tu-propio-celular-por-usd200-gracias-a-arduino/). Consultado el 22 de diciembre de 2013.
24. «Language Reference» (http://arduino.cc/en/Reference/Extended). *http://arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
25. «Serial» (http://arduino.cc/en/Reference/Serial). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
26. «Port Registers» (http://arduino.cc/en/Reference/PortManipulation). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
27. «AVR Code» (http://www.arduino.cc/playground/Main/AVR). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
28. «Arduino/Processing Language Comparison» (http://arduino.cc/en/Reference/Comparison?from=Main.ComparisonProcessing). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
29. «Language Reference(API)/Processing 2+» (http://processing.org/reference/). *http://processing.org/* (en inglés). Consultado el 22 de diciembre de 2013."
30. «Arduino - Libraries» (http://arduino.cc/en/Reference/Libraries). *http://arduino.cc/* (en inglés). Consultado el 22 de diciembre de 2013.
31. «EEPROM Library» (http://arduino.cc/en/Reference/EEPROM). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
32. «Ethernet Library» (http://arduino.cc/en/Reference/EEPROM). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
33. «Firmata Library» (http://arduino.cc/en/Reference/Firmata). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
34. «LiquidCrystal Library» (http://arduino.cc/en/Reference/LiquidCrystal). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
35. «Servo Library» (http://arduino.cc/en/Reference/Servo). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
36. «SoftwareSerial Library» (http://arduino.cc/en/Reference/SoftwareSerial). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
37. «Stepper Library» (http://arduino.cc/en/Reference/Stepper). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
38. «Wire Library» (http://arduino.cc/en/Reference/Wire). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
39. «Writing a Library for Arduino» (http://arduino.cc/en/Hacking/LibraryTutorial). *http://www.arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.
40. «Interfacing with Other Software» (http://www.arduino.cc/playground/Main/InterfacingWithSoftware). *http://arduino.cc* (en inglés). Consultado el 22 de diciembre de 2013.



Proyecto de un semáforo realizado con Arduino y Physical Etoys.

Bibliografía

- Oxer, Jonathan; Blemings, Hugh (28 de diciembre de 2009). *Practical Arduino: Cool Projects for Open Source Hardware* (http://www.apress.com/book/view/9781430224778) (1ª edición). Apress. p. 500. ISBN 1430224770.
- Noble, Joshua (15 de julio de 2009). *Programming Interactivity: A Designer's Guide to Processing, Arduino, and openFramework* (http://or

eilly.com/catalog/9780596800581/) (1ª edición). O'Reilly Media. p. 768. ISBN 0596154143.
- Banzi, Massimo (24 de marzo de 2009). *Getting Started with Arduino* (https://books.google.com.ar/books?id=Xd3SBQAAQBAJ&printsec=frontcover&hl=es&source=gbs_atb#v=onepage&q&f=false) (1ª edición). Make Books. p. 128. ISBN 9781449363291.
- Martínez de Carvajal Hedrich, Ernesto (20 de diciembre de 2015). *100 Proyectos de Robótica con Bitbloq y Arduino* (2ª edición). Martínez de Carvajal Hedrich. p. 386. ISBN 978-8460843177.

Enlaces de externos

- Wikimedia Commons alberga contenido multimedia sobre **Arduino**.
- Proyecto Arduino (Sitio oficial de la empresa radicada en Italia) (http://www.arduino.org/)
- Proyecto Arduino (Sitio oficial con foros y referencia de programación) (http://www.arduino.cc/)
- Arduino Stack Exchange (http://arduino.stackexchange.com/) sitio de preguntas y respuestas
- Comparativa de los distintos modelos disponibles (https://www.sparkfun.com/arduino_guide)
- Proyectos con arduino paso a paso (http://www.youngeek.com/proyectos-arduino/) (ejemplos con videotutoriales)
- Biicode (https://www.biicode.com/): Aplicación para gestionar librerías y proyectos de Arduino
- Entorno gráfico de programación para Arduino (http://minibloq.org)
- Winkhel: cómo un proyecto Arduino puede convertirse en un producto real (http://www.winkhel.com) (en español)
- Scada para Arduino (http://www.acimut.com/monitoriza/monitorizaforarduino.html)
- S-Remote Control: Aplicación Android para controlar Arduino por UDP o TCP (https://play.google.com/store/apps/details?id=com.appopulus.remotecontrol&feature=search_result#?t=W251bGwsMSwxLDEsImNvbS5hcHBvcHVsdXMucmVtb3RlY29udHJvbCJd)
- B4X B4R -Desarrolle aplicaciones en Basic gratuitamente que generan código Arduino (https://www.b4x.com/b4r.html)
- Ejercicios de diferente nivel para iniciarse en la electrónica con Arduino (http://www.drouiz.com/tutoriales-arduino-2/)

Enlaces externos de Arduino educativo para las escuelas:

14 robots y kits para niños para enseñarles robótica y programación (https://descubrearduino.com/10-kits-roboticos-para-ensenar-a-los-ninos-robotica-y-programacion/)

Prácticas Arduino (http://www.thinkbit.org/practicas-arduino/)

Categorías: Wikipedia:Trasladar a Wikilibros | Hardware libre | Microcontroladores | Robótica | Internet de las cosas | Computadoras monoplaca | Electrónica digital | Arduino

---