

Introducción a Arduino

Fernando Curiel Aguirre

Taller de Electrónica Libre

6 de Abril 2016

1 Introducción

2 Hardware

3 Software

- Introducción
- Variables
- Control de flujo
- Manejo de pines

Esquema de la presentación

1 Introducción

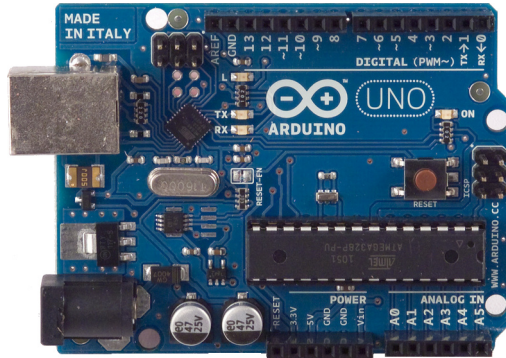
2 Hardware

3 Software

- Introducción
- Variables
- Control de flujo
- Manejo de pines

¿Qué es Arduino?

Arduino Uno



Basada en el **software libre**. Permite:

- Estudiar el hardware para entender cómo funciona
- Hacer modificaciones al hardware
- Poder compartir esas modificaciones con la comunidad

Basada en el **software libre**. Permite:

- Estudiar el hardware para entender cómo funciona
- Hacer modificaciones al hardware
- Poder compartir esas modificaciones con la comunidad

Basada en el **software libre**. Permite:

- Estudiar el hardware para entender cómo funciona
- Hacer modificaciones al hardware
- Poder compartir esas modificaciones con la comunidad

Licencias de Arduino:

- Diseños CAD de las placas bajo licencia Creative Commons Attribution Share-Alike
- Software del entorno de desarrollo bajo licencia GPL
- Librerías de C/C++ para el microcontrolador liberadas bajo licencia LGPL

[Main Site](#) [Blog](#) [Playground](#) [Forum](#) [Labs](#) [Store](#) [Help](#) | [Sign in](#) or [Register](#)



[Buy](#) [Download](#) [Getting Started](#) [Learning](#) [Reference](#) [Hardware](#) [FAQ](#)



Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer (e.g. Flash,

Consecuencias:

- Comunidad activa de usuarios y desarrolladores (foros, ejemplos, tutoriales, etc)
- Gran cantidad de software y hardware disponible para autoconstruir
- Precios bajos

Esquema de la presentación

1 Introducción

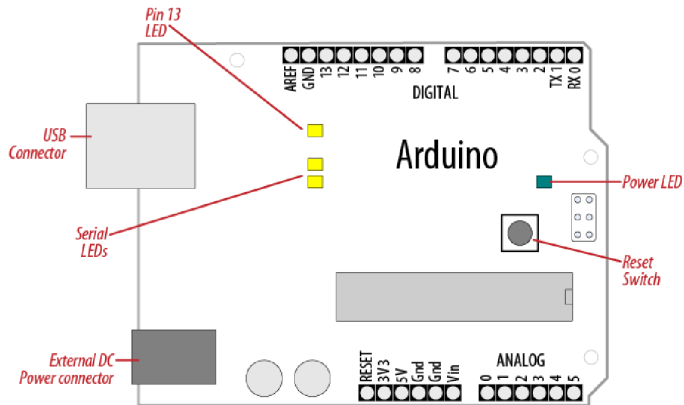
2 Hardware

3 Software

- Introducción
- Variables
- Control de flujo
- Manejo de pines

Características

- Microcontrolador: ATmega328 (8 bits)
- Alimentación via USB (5 V) o independiente (7-12 V)
- 14 pines de entrada/salida (I/O) digitales (6 con Pulse Width Modulation: PWM)
- Corriente máxima por pin I/O: 40 mA
- 6 pines de entrada analógica
- Permite comunicación serial
- Memoria flash: 32 KB
- Frecuencia del reloj: 16 MHz



M. Margolis, *Arduino Cookbook*, O'Reilly Media 2011.

Esquema de la presentación

1 Introducción

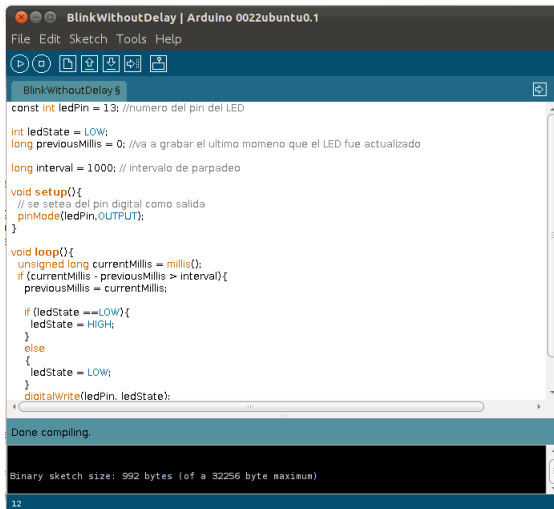
2 Hardware

3 Software

- Introducción
- Variables
- Control de flujo
- Manejo de pines

- Basado en Processing y similar a C++
- Sketches = Código fuente
- Lenguaje compilado

Entorno de desarrollo: Arduino IDE



```
const int ledPin = 13; //numero del pin del LED

int ledState = LOW;
long previousMillis = 0; //va a grabar el ultimo momento que el LED fue actualizado

long interval = 1000; // intervalo de parpadeo

void setup(){
  // se setea del pin digital como salida
  pinMode(ledPin,OUTPUT);
}

void loop(){
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis > interval){
    previousMillis = currentMillis;

    if (ledState == LOW){
      ledState = HIGH;
    }
    else
    {
      ledState = LOW;
    }
    digitalWrite(ledPin, ledState);
  }
}
```

Done compiling.

Binary sketch size: 992 bytes (of a 32768 byte maximum)

12

Proceso

Código fuente

Proceso

Código fuente -> Compilación ("verify")



The screenshot shows the Arduino IDE interface. The title bar reads "BlinkWithoutDelay | Arduino 0022ubuntu0.1". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for running, stopping, saving, opening, and other sketch-related actions. The sketch name "BlinkWithoutDelay" is displayed in the top right of the editor area. The code in the editor is as follows:

```
const int ledPin = 13; //numero del pin del LED

int ledState = LOW;
long previousMillis = 0; //va a grabar el ultimo momeno que el LED fue actua
long interval = 1000; // intervalo de parpadeo

void setup(){
  // se setea del pin digital como salida
  pinMode(ledPin,OUTPUT);
}

void loop(){
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis > interval){
    previousMillis = currentMillis;

    if (ledState == LOW){
      ledState = HIGH;
    }
    else{
      ledState = LOW;
    }
  }
}
```

At the bottom of the IDE, a status bar indicates "Compiling..." with a progress bar.

Proceso

Código fuente -> Compilación ("verify") -> Programarlo en la placa ("upload")



The screenshot shows the Arduino IDE interface. The title bar reads "BlinkWithoutDelay | Arduino 0022ubuntu0.1". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for running, stopping, saving, and other functions. The sketch name "BlinkWithoutDelay" is displayed in the top right of the editor area. The code in the editor is as follows:

```
const int ledPin = 13; //numero del pin del LED

int ledState = LOW;
long previousMillis = 0; //va a grabar el ultimo momeno que el LED fue actua

long interval = 1000; // intervalo de parpadeo

void setup(){
  // se setea del pin digital como salida
  pinMode(ledPin,OUTPUT);
}

void loop(){
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis > interval){
    previousMillis = currentMillis;

    if (ledState == LOW){
      ledState = HIGH;
    }
    else{
      ledState = LOW;
    }
  }
}
```

At the bottom of the IDE window, a status bar shows "Uploading to I/O Board..." and a progress bar. The line number "1" is visible at the bottom left of the code editor.

Estructura de un archivo

```
declaracion de variables;
```

```
void setup()  
{  
  seteo de pines;  
  inicializacion de la comunicacion serial;  
}
```

```
void loop()  
{  
  lo que voy a hacer todo el tiempo;  
}
```

- byte: Enteros (1 byte): -128 a 127
- int: Enteros (2 bytes): -32.768 a 32767
- long: Enteros (4 bytes)
- float, double: Números en punto flotante (4 bytes)
- boolean: Verdadero (**TRUE**) o Falso (**FALSE**)
- char: Un solo caracter
- String: Listas de caracteres

Variables

Definición

```
int nombreVariable1 = 0
```

```
float nombreVariable2 = 1.24
```

```
int nombreArray[] = {valor0, valor1, valor2, ...}
```

```
int nombreArray2[5]
```

¡No olvidar “;” al final de cada sentencia!

Variables

Definición

```
int nombreVariable1 = 0;
```

```
float nombreVariable2 = 1.24;
```

```
int nombreArray[] = {valor0, valor1, valor2, ...};
```

```
int nombreArray2[5];
```

```
/* Este es un  
bloque de  
comentarios  
*/
```

```
// En cambio, este es un comentario de linea
```

- $A == B$: A igual a B
- $A != B$: A distinto de B
- $A < B$: A menor que B
- $A <= B$: A menor o igual a B
- $A > B$: A mayor que B
- $A >= B$: A mayor o igual a B

- `A && B` : **A AND B**
- `A || B` : **A OR B**
- `!A` : **NOT A**

Control de flujo

if

```
if (condicion) {  
    hacer algo;  
}
```

```
if (condicion) {  
    hacer algo;  
}
```

```
if (condicion) {  
    hacer algo;  
} else {  
    hacer otra cosa;  
}
```

Control de flujo

if

```
if (condicion) {  
    hacer algo;  
}
```

```
if (condicion) {  
    hacer algo;  
} else {  
    hacer otra cosa;  
}
```

```
// Ejemplo:  
if (buttonPushCounter % 4 == 0){  
    digitalWrite(ledPin, HIGH);  
}  
else {  
    digitalWrite(ledPin, LOW);  
}
```

Control de flujo

while

```
while (condicion) {  
    hacer algo;  
}
```


Control de flujo

while

```
while (condicion) {  
    hacer algo;  
}
```

```
// Ejemplo:  
int var = 0;  
while(var < 200){  
    // hacer algo repetitivamente 200 veces  
    var++;  
}
```

Control de flujo

for

```
for (inicializacion; condicion; incremento) {  
    hacer algo;  
}
```

Control de flujo

for

```
for (inicializacion; condicion; incremento) {  
    hacer algo;  
}
```

// Ejemplo: atenuar un LED usando un pin con PWM

```
int PWMpin = 10; // hay que poner una resistencia de 470 Ohm  
    en serie
```

```
void setup(){  
    // no es necesario configurar nada  
}
```

```
void loop(){  
    for (int i=0; i <= 255; i++){  
        analogWrite(PWMpin, i);  
        delay(10);  
    }  
}
```

Manejo de pines digitales

Los pines digitales (pin 0 a 13) pueden tomar sólo 2 valores: 0 V (LOW) o +5 V (HIGH)¹.

```
int pinEntrada = 10;
int pinSalida = 13;

void setup(){
  pinMode(pinEntrada, INPUT);
  pinMode(pinSalida, OUTPUT);
}

void loop(){
  digitalWrite(pinSalida, HIGH); //pone el pin 13 en +5V
  valor = digitalRead(pinEntrada); //lee el valor del
    pin 10
}
```

Manejo de pines analógicos

Los pines analógicos (A0 a A5) sólo sirven para **leer** señales, no para escribir.

La Arduino tiene un convertidor Analógico/Digital (A/D), de 10 bits, que retorna enteros entre 0 y 1023.

Para leer el valor en el pin A2, se debe usar la función `analogRead()`:

```
int valor; // variable que va a almacenar lo que lee un pin

void setup(){
    // no es necesario inicializar nada
}

void loop(){
    valor = analogRead(A2); // valor es un int entre 0 y 1023
}
```

