

# Introducción a

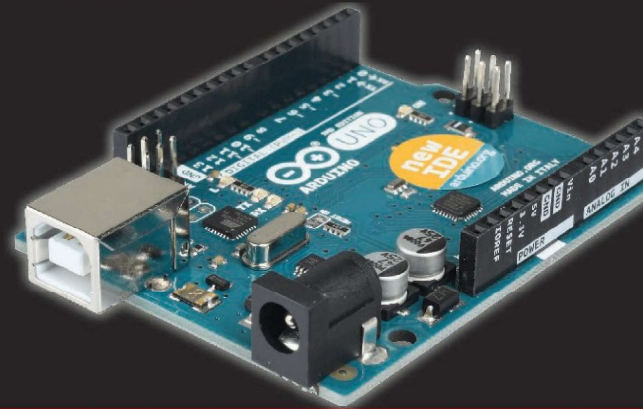
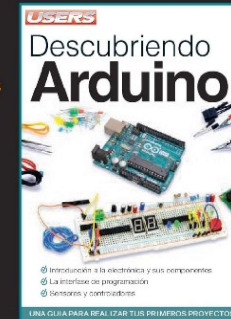


Claudio Peña / claudione@gmail.com

Arduino es una placa de desarrollo de hardware libre, que puede ser utilizada tanto por aficionados como por fabricantes para diseñar y construir dispositivos que interactúen con el mundo real, a través de una gran cantidad de sensores y otros elementos electrónicos que están disponibles en el mercado.

Aunque utilizamos el término Arduino para referirnos a un tipo específico de placa de desarrollo, también se usa para hablar de la empresa que fabrica estas placas o para describir a la comunidad en torno a las diferentes placas compatibles.

En nuestra [Guía Descubriendo Arduino](#) analizamos a fondo esta placa de desarrollo.



## Partes de una placa Arduino

**A nivel de hardware, Arduino se compone de varias partes e interfaces diferentes, todas reunidas en la placa de circuito. Aunque el diseño original ha cambiado a lo largo de los años y algunas placas derivadas incluyen elementos adicionales, en su forma básica encontraremos los siguientes elementos.**

### 1. Pines

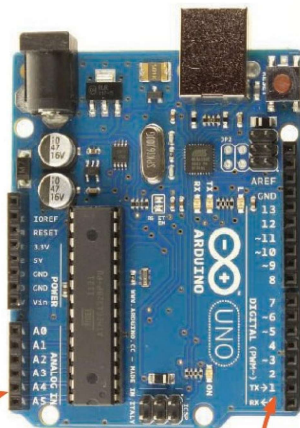
Podemos utilizarlos para conectar la placa con componentes externos. Pueden ser digitales o analógicos.

No importa si la placa Arduino es original o compatible (desarrollada por otra empresa pero basada en la arquitectura

de Arduino), todos los pines estarán dispuestos en un patrón específico. Por lo tanto, si compramos una placa adicional (shield o escudo), diseñada para encajar en ellos, deberíamos poder conectarla fácilmente en todas las placas Arduino.

#### Pines analógicos:

pueden leer un amplio rango de valores, por lo que son útiles para realizar un control más detallado de las lecturas obtenidas. En general, encontraremos seis pines analógicos. Mediante las entradas de este tipo, es posible obtener datos de sensores en forma de variaciones continuas de un voltaje.



#### Pines digitales:

son capaces de leer y escribir un solo estado, es decir, encendido o apagado. En la mayoría de las placas Arduino hay 14 pines de E/S (entrada/salida) digitales, que pueden configurarse como entrada o salida; a ellos podemos conectar cualquier dispositivo que sea capaz de transmitir o recibir señales digitales de 0 y 5V.

### 2. Conector de alimentación

Proporciona energía a la placa Arduino y también puede alimentar componentes conectados a ella, como LEDs y sensores. El conector de alimentación puede conectarse a un adaptador de CA o a una batería pequeña.

En la tabla adjunta podemos ver el consumo de energía de diferentes placas Arduino originales:

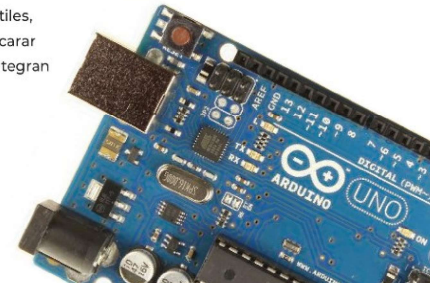
MODELO	CONSUMO EN MA	DURACION DE UNA BATERÍA DE 1200 MAH
Arduino UNO	46	26 horas
Arduino MEGA	93	Casi 13 horas
Arduino DUE	75	16 horas
Arduino Nano	15	80 horas

### 3. Microcontrolador

El microcontrolador es el chip principal que nos permite programar la placa para que ejecute comandos y pueda tomar decisiones basadas en las lecturas proporcionadas por varias entradas (pines). El chip puede variar según el tipo de Arduino con el que estemos trabajando, pero generalmente encontraremos controladores Atmel, como ATmega8, ATmega168, ATmega328, ATmega1280 o ATmega2560. Las diferencias entre ellos son sutiles, pero importantes a la hora de encarar ciertos proyectos; por ejemplo, integran distinta cantidad de memoria.

### 4. Conector USB

En la mayoría de las placas nuevas, existe un puerto USB estándar, que hace posible la comunicación con la placa desde la PC, así como la carga de nuevos programas para que Arduino los ejecute. En muchas ocasiones, también podemos alimentar la placa Arduino utilizando el puerto USB, lo que elimina la necesidad de realizar una conexión de alimentación en forma separada.

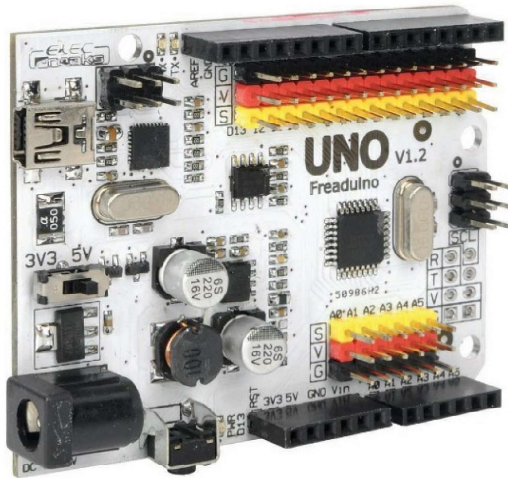


## ¿Qué necesitamos para comenzar?

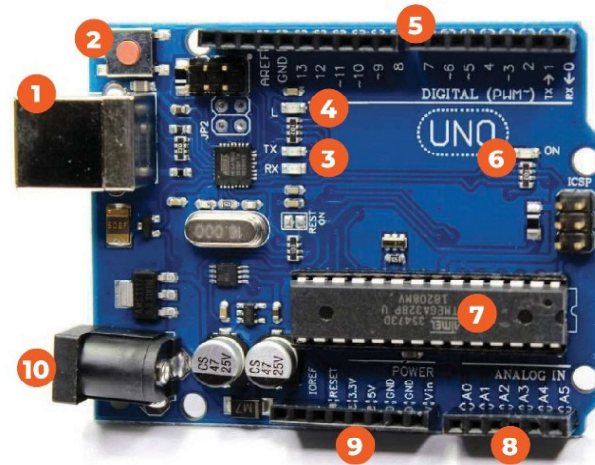
Para comenzar a trabajar con Arduino necesitamos, en primer lugar, la placa de desarrollo; pero también debemos contar con una serie de componentes electrónicos básicos, que dependerán de cada proyecto que decidamos realizar.

### Placa de desarrollo

Podemos utilizar cualquiera de las [placas oficiales de Arduino](#) o, también, una [placa compatible](#). En este caso, trabajaremos con Arduino Uno por ser una placa muy versátil y que ofrece todo lo necesario para proyectos tanto sencillos como avanzados. En la actualidad, su última versión es la R3.



La placa de desarrollo no puede trabajar sola: es necesario contar con una serie de componentes electrónicos adicionales, cada uno de los cuales nos servirá para una tarea específica y, en conjunto, para lograr proyectos completos.



- 1 Puerto USB:** se utiliza para entregar energía a la placa Arduino mientras estamos trabajando con la PC.
- 2 Botón de reinicio:** su función es permitirnos resetear el microcontrolador ATmega, de forma de eliminar lo que hayamos cargado.
- 3 LED TX y RX:** se utilizan para verificar que existe comunicación entre la placa y la computadora.
- 4 Clavija 13 LED:** es un activador que se presenta en forma predeterminada en Arduino UNO.
- 5 Clavijas digitales:** pueden usarse, por ejemplo, para `digitalRead()` o `analogWrite()`, entre otras opciones.
- 6 LED de corriente:** podemos utilizar este LED para verificar que la placa recibe energía en forma correcta.
- 7 Microcontrolador ATmega:** se considera el corazón de la placa Arduino UNO.
- 8 Entrada analógica:** conjunto de clavijas que funcionan como entradas analógicas; podemos utilizarlas con `analogRead()`.
- 9 Clavijas GND y 5V:** son adecuadas para otorgar corriente de +5V a los circuitos en los que trabajemos, y también una toma de tierra.
- 10 Conector de corriente:** puede trabajar con voltajes que van desde 7V hasta 12V.



### Protoboard

Una placa sin soldadura, protoboard o breadboard es un elemento necesario e ideal para trabajar con circuitos temporales y prototipos. Este tipo de placas no requiere soldadura para lograr la creación de un circuito, por lo que sirven para desarrollar diseños temporales o para probar ideas en forma rápida, con la posibilidad de utilizarlas una y otra vez.



### Cables de puente

Estos cables se utilizan para realizar las conexiones de los diversos componentes en el protoboard y con la placa Arduino. Tienen diferentes longitudes para efectuar conexiones a distancias variadas, y vienen en distintos colores para que su manejo e identificación una vez conectados pueda realizarse rápidamente.

### Condensador

También llamado capacitor, es un componente electrónico pasivo, como las resistencias. Se utiliza generalmente para almacenar carga eléctrica, en forma de campo eléctrico. Estos elementos desempeñan un papel importante en muchos circuitos eléctricos y electrónicos, por ejemplo, en los flashes de las cámaras de fotos: el condensador se carga desde la batería para después soltar toda su energía y conseguir tensiones muy altas durante un corto período.



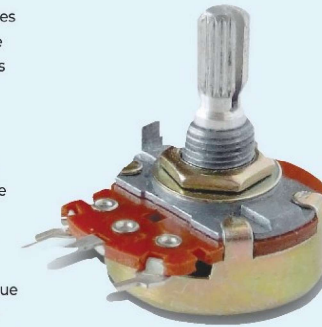
### Diodo

Es un componente electrónico que permite el flujo de la electricidad en un solo sentido. Su funcionamiento es similar al de un interruptor que abre o cierra los circuitos.

Estos dispositivos están conformados por dos clases de materiales diferentes, que se traducen a dos terminales, un ánodo (+) y un cátodo (-). Pueden ser de varios tipos, aunque los más conocidos cuando nos iniciamos en Arduino son los diodos LED, aquellos que emiten luz cuando están polarizados directamente.

### Potenciómetro

Es una resistencia variable que posee tres terminales o conectores. Dos de ellos se conectan a una resistencia fija, mientras que el tercero se puede mover para conseguir valores diferentes. Gracias a este elemento podemos elegir el valor por tomar; de esta forma, controlaremos la intensidad de corriente que fluye por el circuito o la diferencia de potencial, según esté conectado en paralelo o en serie. La variación de la resistencia va desde un valor mínimo, que generalmente es de 0 ohmios, hasta un valor máximo  $R_{max}$  (5k, 10k o 20k ohmios).



### Broche de presión de pila

Aunque es un componente sencillo, lo necesitamos para conectar una pila de 9V a la clavija de corriente, que a su vez conectaremos al protoboard o a la placa Arduino. Es una estructura que posee dos broches adecuados para las baterías de 9V, unidos a dos filamentos que serán los que tendremos que conectar al protoboard o a la placa.



## Capítulo 01



# Instalación y configuración

Antes de trabajar con Arduino IDE es necesario realizar la instalación y configuración básica de este entorno integrado de desarrollo. En este primer capítulo, aprenderás a realizar estas tareas.

### ¿Qué es Arduino IDE? / 9

- Características principales
- IDEs alternativos

### Instalar Arduino IDE / 15

- Instalar en Windows
- Instalar en GNU/Linux
- Instalar en otros sistemas
- Construir desde el código fuente

### Configurar el IDE / 22

### Entorno de trabajo / 28

### Actividades / 33

- Test de autoevaluación
- Ejercicios prácticos

## ¿Qué es Arduino IDE?

RU

Arduino es una placa de desarrollo basada en un microcontrolador Atmel. Es importante precisar que los microcontroladores son circuitos integrados en los que es posible grabar instrucciones, las que deben escribirse con un lenguaje de programación y utilizando un entorno de desarrollo compatible. En el caso de Arduino, este entorno se llama Arduino IDE. El **entorno de desarrollo integrado** o **IDE de Arduino** es una aplicación **multiplataforma** que puedes utilizar para escribir y cargar programas en placas Arduino y también en aquellas que sean compatibles. Pero no solo eso, ya que gracias a núcleos generados por terceros, también se puede utilizar para cargar programas en placas de desarrollo de otros proveedores.

```
Blink | Arduino 1.8.5
Blink
This example code is in the public domain.
http://www.arduino.cc/en/Tutorial/Blink
*/
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
32 Arduino/Genuino Uno on COM1
```

Figura 1.1.

## 1. Instalación y configuración

Por lo tanto, antes de comenzar a programar la placa Arduino, es necesario que cuentes con su IDE, pues de esa forma dispondrás del conjunto de herramientas de software adecuadas para desarrollar y grabar todo el código que hará que tu placa funcione como lo requieres.

El IDE de Arduino te permitirá no solo escribir tu código, sino también **depurar**, editar y grabar los programas o **sketches** en la placa de desarrollo, de una manera rápida y sencilla.

Teniendo esto en cuenta, lo primero que debes hacer es instalar Arduino IDE en tu computadora, aunque también puedes elegir utilizar la versión del IDE online, la que conocerás en el último capítulo de este libro.

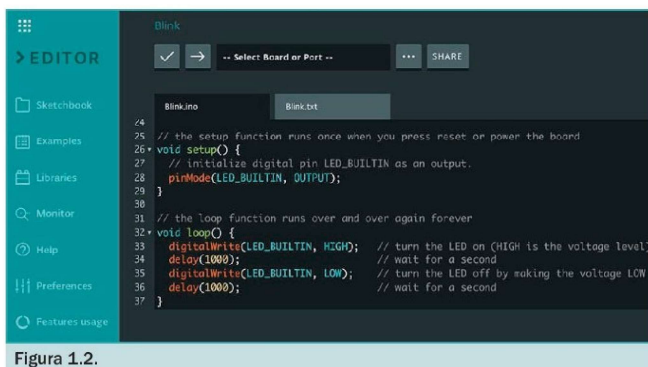


Figura 1.2.

## Características principales

Arduino IDE se distribuye en forma gratuita, por lo que solo necesitas acceder al sitio web oficial de la aplicación para descargar una copia instalable. También se trata de un software que se distribuye con una licencia libre, de modo que es posible acceder al **código fuente** del IDE y construir el instalador desde él o realizar las modificaciones que consideres

necesarias. Para la mayoría de los usuarios, bastará con descargar el instalador adecuado para el sistema operativo y proceder con la instalación, tal como aprenderás más adelante en este mismo capítulo. Por otra parte, el código fuente de Arduino IDE se encuentra disponible en la dirección <https://github.com/arduino/Arduino>.

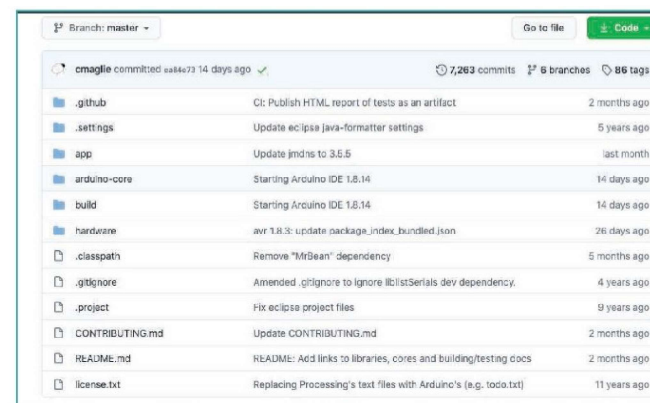


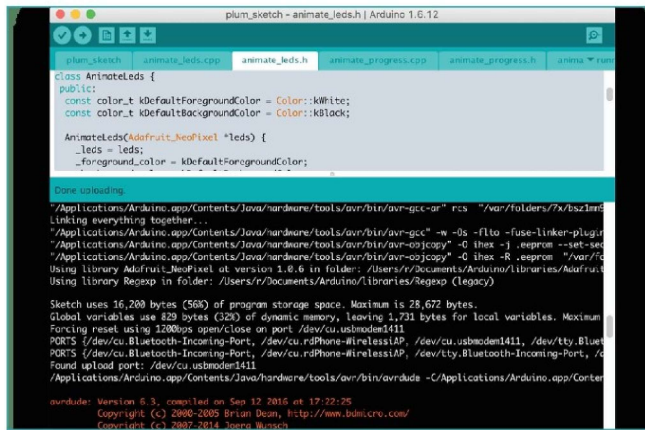
Figura 1.3. En el sitio <https://github.com/arduino/Arduino> puedes descargar el código fuente de Arduino IDE.

Otra de las ventajas de este IDE es que se trata de una aplicación multiplataforma, es decir que puede ser instalada y utilizada en diferentes sistemas operativos, por ejemplo, Microsoft Windows, GNU/Linux o Mac OSX, entre otros. Para obtener el instalador adecuado, debes visitar el sitio web oficial del IDE y, allí, elegir la opción que necesites.

Por otra parte, la interfaz de usuario de Arduino IDE es muy sencilla. Aunque esto parezca una debilidad debido a las escasas opciones que se presentan al acceder por primera vez, a medida que la conozcas más a fondo, verás que proporciona todo lo necesario y más aún. La disposición de las áreas

## 1. Instalación y configuración

de trabajo y los menús de opciones, junto a una consola de errores, se reúnen en la pantalla principal del IDE; en resumen, cada opción está justo donde la precisas, sin que la ventana principal ocupe toda la pantalla.



```

plum_sketch - animate_leds.h | Arduino 1.6.12
animate_leds.h
class AnimateLeds {
public:
const color_t kDefaultForegroundColor = Color::kWhite;
const color_t kDefaultBackgroundColor = Color::kBlack;

AnimateLeds(Arduino_NeoPixel *leds) {
  _leds = leds;
  _foreground_color = kDefaultForegroundColor;
}
};

Sketch uses 16,208 bytes (58%) of program storage space. Maximum is 28,672 bytes.
Global variables use 328 bytes (32%) of dynamic memory, leaving 1,731 bytes for local variables. Maximum
Forcing reset using 1200bps open/close on port /dev/cu.usbmodem1411
PORTS: /dev/cu.Bluetooth-Incoming-Port, /dev/cu.rdPhone-WirelessIAP, /dev/cu.usbmodem1411, /dev/tty.Bluet
PORTS: /dev/cu.Bluetooth-Incoming-Port, /dev/cu.rdPhone-WirelessIAP, /dev/tty.Bluetooth-Incoming-Port, /c
Found upload ports: /dev/cu.usbmodem1411
/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/avrddude -C/Applications/Arduino.app/Conten
ardude: Version 6.3, compiled on Sep 12 2015 at 17:12:25
Copyright (C) 2006-2005 Brian Dean, http://www.bdmicro.com/
Copyright (C) 2007-2014 Texas Instruments

```

Figura 1.4.

Otro aspecto importante es que, gracias a las herramientas de Arduino IDE, podrás cargar los programas que escribas directamente en la **memoria flash** de Arduino, en unos pocos pasos, sin necesidad de ejecutar complejos procedimientos. De esta forma, tendrás tu placa ejecutando un programa en un tiempo muy reducido. Como verás más adelante, en este mismo capítulo, la configuración inicial de Arduino IDE se realiza en pocos pasos y te llevará un tiempo mínimo. Así tendrás las herramientas de programación listas para usar sin complicaciones. Más adelante se explica en detalle el proceso de configuración del IDE. La última versión de este IDE propone algunas características novedosas, como detección automática de la placa conectada,

información sobre memoria flash y SRAM ocupada por un sketch o proyecto, autoguardado al compilar y cargar un sketch, y carga de sketches vía red para algunas placas.

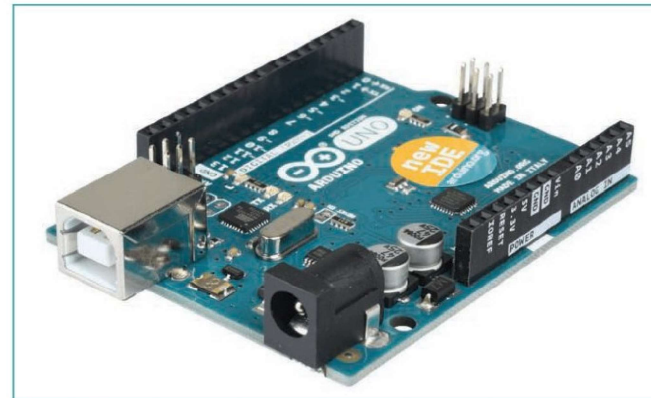


Figura 1.5.

## IDEs alternativos

Algunos IDEs alternativos instalables y opciones que permiten programar para Arduino son los siguientes:

- **Stino** (<https://github.com/Robot-Will/Stino>) es un plugin para el editor Sublime Text. Permite desarrollar para Arduino de manera fácil, ya que ofrece acceso muy rápido a toda la estructura del proyecto, posee autocompletado y otras características importantes (Figura 1.6.)
- **Atmel Studio** (<https://microchipdeveloper.com/atstudio:studio7intro>) se basa en Visual Studio, y es adecuado para trabajar con Arduino y con los distintos microprocesadores compatibles de Atmel. Solo es compatible con Microsoft Windows (Figura 1.7.)



## 1. Instalación y configuración

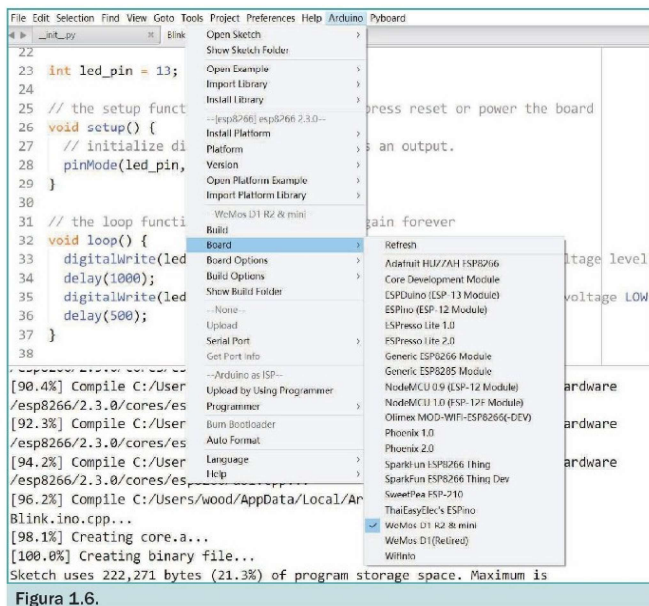


Figura 1.6.

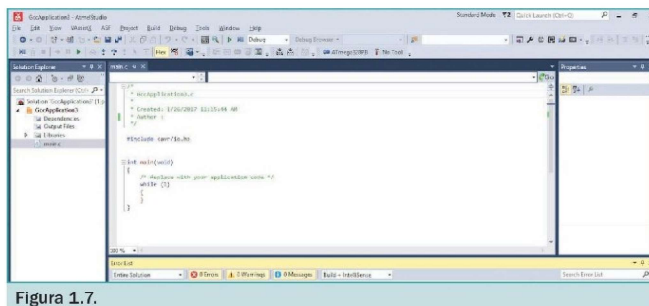


Figura 1.7.



- MariaMole (<http://dalpix.com/mariamole>) es un editor de Arduino bastante ligero, por lo que puede ejecutarse en computadoras con recursos limitados. Permite trabajar con opciones de ordenación por proyectos, ofrece variadas mejoras visuales y se ejecuta en diferentes sistemas operativos.

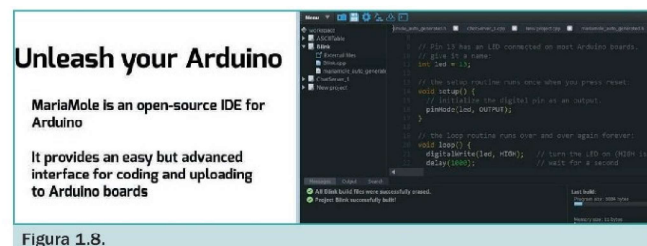


Figura 1.8.

## Instalar Arduino IDE

La última versión del IDE de Arduino es la 1.8.xx, pero los grandes cambios no se produjeron en los últimos lanzamientos sino en la actualización de la versión 0.22 a la 1.0, y más tarde, en la actualización de la versión 1.0.6 a la 1.6.0, las que entregaron importantes mejoras en el IDE. Entre estos cambios es destacable, desde la versión 1.6.2, la incorporación de la gestión de librerías y la gestión de placas mejoradas respecto a la versión anterior, y también los avisos de actualización tanto de librerías como de cores. Todos los cambios realizados entre cada versión pueden verse en la dirección [www.arduino.cc/en/Main/ReleaseNotes](http://www.arduino.cc/en/Main/ReleaseNotes). De cualquier modo, siempre es aconsejable instalar la última versión estable al momento de realizar la descarga desde el sitio oficial (Figura 1.9.)

## 1. Instalación y configuración

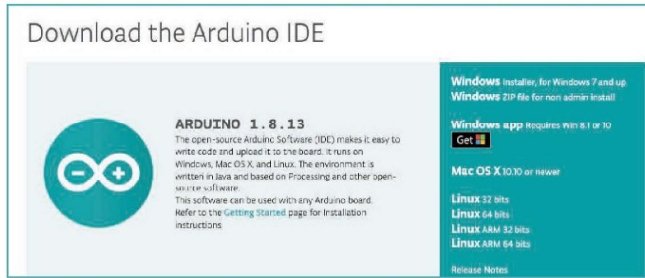


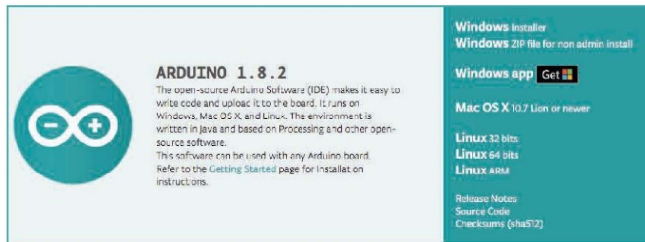
Figura 1.9.

También está disponible para los usuarios la versión 1.9.0 en fase beta, con el fin de probar las novedades que incorporará el IDE, aunque no es recomendable utilizarla para entornos de producción. A continuación, se realizará la instalación del IDE en diferentes sistema operativos.

## Instalar en Windows

### PASO 1

Ingresa en la dirección [www.arduino.cc](http://www.arduino.cc) y haz clic en el enlace **Software**, que se encuentra en la barra superior de opciones. En la ventana que se abre, baja hasta la sección Arduino IDE y presiona sobre **Windows Installer**.



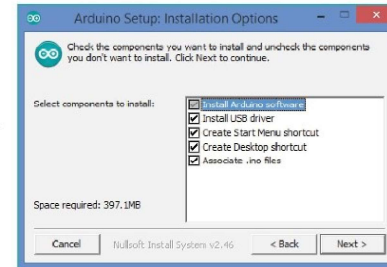
### PASO 2

Una vez que la descarga haya finalizado, haz doble clic sobre el archivo adecuado y espera mientras se inicia el asistente de instalación. En la primera pantalla será necesario aceptar el acuerdo de licencia, para lo cual presiona **I Agree**.



### PASO 3

Selecciona los componentes que serán instalados; es recomendable marcar, al menos, las opciones **Install Arduino Software**, **Install USB driver** y **Associate .ino files**; luego presiona el botón **Next**.



### PASO 4

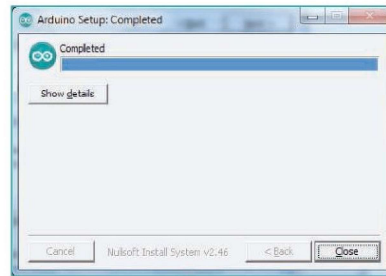
Es posible que aparezca una ventana que indica que se instalarán algunos drivers, acepta que así sea. Luego el proceso continuará.



## 1. Instalación y configuración

### PASO 5

El proceso de instalación solo tardará un momento y podrás ver su avance gracias a la barra de la ventana. Cuando se complete, presiona sobre el botón **Close**.



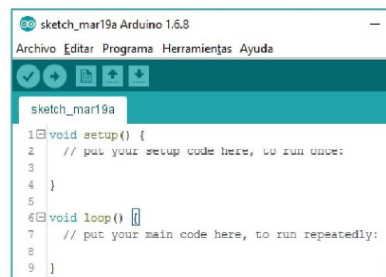
### PASO 6

En este momento, el IDE de Arduino ya se encontrará instalado en la computadora. Es necesario saber que, en las nuevas versiones, no se necesita instalar los drivers, pues vienen integrados. Inicia el software y verás su pantalla de carga.



### PASO 7

Una vez cargado, la apariencia del IDE será similar a la que se observa en la imagen.



RU

## Instalar en GNU/Linux

Para usar Arduino IDE en un sistema GNU/Linux solamente deberás acceder a la **terminal** de comandos y escribir lo siguiente:

```
sudo apt-get install arduino
```

Una vez que la tarea termine, verás que has logrado la instalación de un IDE básico que no posee un gestor de librerías actualizado, por lo que la configuración se deberá realizar en forma manual. Para extender el uso de Arduino será necesario acceder a la página de descargas en [www.arduino.cc](http://www.arduino.cc) y buscar el módulo para Linux. Una vez descargado, debes extraer el contenido del archivo, acceder a la carpeta adecuada e instalar mediante el comando:

```
sudo ./install.sh
```

Luego de hacerlo, sí tendrás acceso a todas las opciones necesarias para escribir tus programas en cualquier placa Arduino, con una gran variedad de librerías para ejecutar tus códigos.

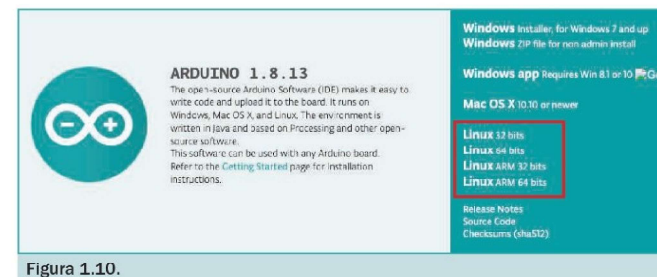


Figura 1.10.



## 1. Instalación y configuración

### Instalar en otros sistemas

Si deseas utilizar el IDE de Arduino en otro sistema operativo, debes acceder a [www.arduino.cc](http://www.arduino.cc), hacer clic sobre **Software** y elegir la descarga que desees; por ejemplo, el instalador para Mac OSX, cuyo peso comprimido es de aproximadamente 158 MB.

Si utilizas el navegador web Safari, el archivo será automáticamente descomprimido; como haces con todas las aplicaciones en Mac OSX, tienes que copiar el programa en la carpeta de **Aplicaciones**.

Con el IDE de Arduino instalado, solo resta iniciarlo. Una vez que el programa se cargue, verás la ventana principal y podrás comenzar a trabajar.

### Construir desde el código fuente

Quizá los usuarios más avanzados requieran construir el IDE desde su código fuente. Si se realiza en un sistema Windows será necesario contar con **Cygwin**, **Java JDK** y **Ant**.

Cygwin se encuentra en [www.cygwin.com](http://www.cygwin.com) o mediante el enlace directo [www.cygwin.com/setup-x86.exe](http://www.cygwin.com/setup-x86.exe) para 32 bits o [www.cygwin.com/setup-x86\\_64.exe](http://www.cygwin.com/setup-x86_64.exe) para Windows de 64 bits, aunque también está disponible en [www.redhat.com/services/custom/cygwin](http://www.redhat.com/services/custom/cygwin).

Mientras se realiza la instalación de Cygwin, debes seleccionar los paquetes:

- **git** (para control de versiones)
- **make, mingw64-x86\_64-gcc-g++** (para compilar arduino.exe)
- **perl**
- **zip** (para descomprimir archivos)

Además, aunque se encuentran incluidos en los valores seleccionados en forma predeterminada, deberás asegurarte de que estén marcados **coreutils** (o **textutils**), **gzip** y **tar**.

RU

Finalmente, aunque no resultan esenciales, también pueden ser útiles:

- **openssh** (cliente ssh de línea de comandos)
- **nano** (editor de texto)

A continuación, es necesario descargar e instalar **Ant** (<https://ant.apache.org>). Debes agregar el **apache-ant-xxx\bindirectorio** a su ruta y asegurarte de que la ruta al **apache-ant-xxxdirectorio** no contenga comillas.

Para continuar, descarga e instala Java JDK. Apunta la **variable** de entorno **JAVA\_HOME** al directorio raíz JDK. Si aparece un mensaje de error diciendo que “**No se puede ubicar tools.jar. Se espera encontrarlo en C:\Archivos de programa\Java\jre6\lib\tools.jar**”, significa que es necesario configurar **JAVA\_HOME** en la instalación JDK (no JRE).

Por otra parte, si te encuentras en un sistema Linux, necesitarás el **SDK de Java**, además de **avr-gcc**, **avr-g++**, **avr-libc**, **make**, **ant**, **git** y **unzip**.

En Ubuntu o Debian debes ejecutar:

```
sudo apt-get install git make gcc ant openjdk-8-jdk unzip
```

También puede ser necesaria la biblioteca **openjfx** para compilar la versión 1.8.8; para esto ejecuta el comando:

```
sudo apt-get install openjfx
```

En Arch Linux ejecuta lo siguiente:

```
sudo pacman -S jdk8-openjdk jre8-openjdk apache-ant git base-devel
```

## 1. Instalación y configuración

En CentOS Linux:

```
rpm -qa --qf "%{name}\n" git make gcc xz-lzma-compat ant  
\*openjdk\* | sort
```

```
ant  
gcc  
git  
java-1.8.0-openjdk  
java-1.8.0-openjdk-devel  
java-1.8.0-openjdk-headless  
xz-lzma-compat  
make
```

Para instalar los paquetes necesarios, luego:

```
sudo yum -y install make gcc ant xz-lzma-compat java-  
1.8.0-openjdk
```

# Configurar el IDE

Para continuar la puesta a punto del IDE, es preciso configurar el entorno de trabajo. Aunque no se trata de una tarea imprescindible, es una buena idea completar cada uno de los pasos, pues facilitarán la edición de los programas que vas a desarrollar.

RU

## PASO 1

Inicia el IDE de Arduino que instalaste siguiendo las instrucciones de la sección anterior. Cuando aparezca la interfaz principal del IDE, estarás listo para comenzar el proceso de configuración inicial.



## PASO 2

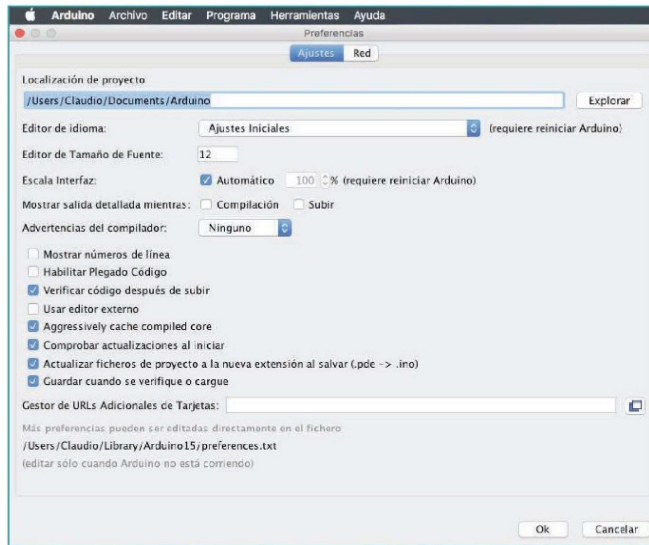
Si has iniciado el IDE en un sistema Windows, tendrás que hacer clic en el menú **Archivo** y, luego, elegir la opción **Preferencias**. Por otra parte, si estás en un sistema Mac OSX, presiona en el menú **Arduino** y, posteriormente, sobre **Preferencias**.



## 1. Instalación y configuración

### PASO 3

En este punto, verás la ventana de preferencias del IDE, la cual se divide en dos pestañas: **Ajustes** y **Red**. Para este proceso de configuración, utilizarás algunas opciones que se encuentran en la primera.



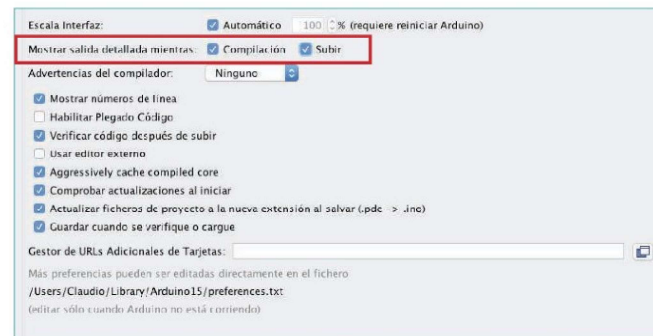
### PASO 4

Busca la casilla que corresponde a la opción **Mostrar números de línea**. Esta permite ver los números de cada línea de los programas que realices. Su importancia radica en que será más sencillo ubicar una línea específica, sobre todo, cuando trabajes con códigos extensos.



### PASO 5

Localiza la opción **Mostrar salida detallada mientras**. Junto a ella se presentan dos alternativas: **Compilación** y **Subir**; marca ambas. De esta manera, tendrás acceso a información detallada tanto cuando realices la compilación del programa como cuando lo subas a Arduino. Esta salida puede incluir datos importantes para ubicar algún problema o dificultad que pudiera presentarse en el código.

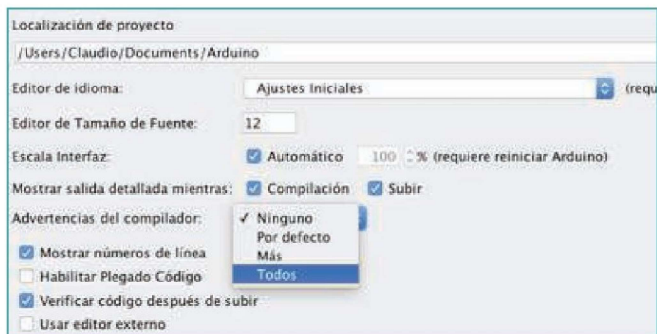




## 1. Instalación y configuración

### PASO 6

Ahora ve a la opción **Advertencias** del **compilador**, despliega el menú que se encuentra a su lado y elige **Todos**. La opción predeterminada es **Ninguno**, que puede ser útil cuando tengas experiencia en el manejo del IDE, pero si recién comienzas con la programación en Arduino, bien vale la pena tener a tu disposición toda la información que pueda ser de ayuda.



### PASO 7

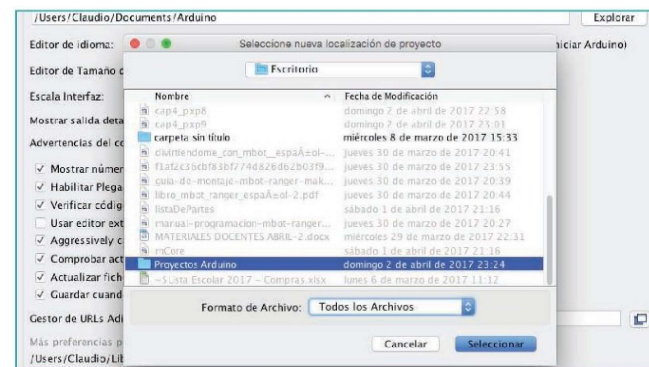
La siguiente opción que marcarás será el plegado de código. Esto facilitará tu trabajo cuando manejes códigos extensos, pues permitirá agrupar y plegar ciertos segmentos para tener el programa más organizado, hacer que sea más entendible y no tener que navegar mucho tiempo para llegar a la línea de código que necesitas.



RU

### PASO 8

Para finalizar la configuración inicial del IDE, elige la ruta que utilizarás para almacenar los proyectos. En forma predeterminada, encontrarás una ubicación como la siguiente: **Users/Claudio/Documents/Arduino**, pero puedes elegir la ruta que más te guste, por ejemplo, **Escritorio/Proyectos Arduino**.



### PASO 9

Una vez que hayas aplicado todas las opciones mencionadas, será necesario hacer clic sobre **Aceptar** para que se almacenen y estén disponibles en todos los proyectos que inicies de aquí en adelante.

Aunque la configuración inicial del IDE debería de bastar para cualquier usuario principiante, ten en cuenta que existen opciones de configuración avanzadas que están reservadas para usuarios con mayor experiencia. Para acceder a ellas, debes abrir el archivo **preferences.txt**, ubicado en la carpeta **AppData/Local/Arduino15**. Hay que tener mucho cuidado al editar las opciones de este archivo y, tal como se advierte en el IDE, realizar las modificaciones solo cuando el programa no se encuentre en ejecución.

## 1. Instalación y configuración

El archivo **preferences.txt** se puede encontrar en las siguientes ubicaciones, dependiendo del sistema operativo:

- C:\Documents and Settings\\Application Data\arduino\preferences.txt (Windows)
- /Users/<Username>/Library/Arduino/preferences.txt (Mac)
- ~/.arduino/preferences.txt (Linux)

# Entorno de trabajo

El área de trabajo de Arduino IDE se divide en cinco grandes secciones:



Figura 1.11.

RU

La mayoría del tiempo trabajarás en la sección del editor de código, donde puedes desarrollar tus proyectos.

En la barra de botones encontrarás algunos elementos importantes:

- **Verificar:** realiza dos funciones: comprueba que no haya error en el código y, si no hay problemas, lo compila.
- **Subir:** se utiliza luego de verificar. Su función es cargar en la memoria del **microcontrolador** el programa que has escrito.
- **Nuevo:** crea un nuevo sketch vacío.
- **Abrir:** despliega un menú con todos los sketches disponibles para abrir. Puedes abrir tus propios sketches, como la gran cantidad que vienen listos como ejemplos para probar, clasificados en categorías dentro del menú.
- **Guardar:** almacena el código de tu sketch en un fichero, el cual tendrá la extensión ".ino". Puedes guardar estos ficheros donde quieras, pero el IDE Arduino ofrece una carpeta específica para hacerlo: **Arduino**, dentro de **Documentos**. Ahí se creará una carpeta con el nombre de tu sketch, así el IDE evita que se mezclen los archivos de los distintos sketches.
- **Monitor Serial:** abre el monitor serial, que permite ver información transmitida desde tu Arduino por el puerto de comunicación serial; luego verás un poco más de él.
- **Menú contextual:** esta pestaña se ubica bajo el botón de monitor serial y permite abrir nuevas pestañas. Esto es de suma utilidad cuando tienes códigos tan largos que necesitas dividirlo en partes para trabajar con más comodidad. Es así porque todas las nuevas pestañas abiertas forman parte del mismo proyecto que la primera pestaña original. Lo más habitual es utilizar pestañas separadas para la definición de funciones, constantes o variables globales.

Por otra parte, en la barra de menú se encuentran las opciones más tradicionales, que de igual manera son bastante útiles: **Archivo**, **Editar**, **Programa**, **Herramientas** y **Ayuda**.

## 1. Instalación y configuración

- **Archivo:** ofrece acciones estándar, como crear un nuevo documento sketch, abrir uno existente, guardarlo, cerrarlo, cerrar el IDE, etc. También añade otras funciones interesantes, como **Ejemplos**, donde puedes acceder a los sketches de ejemplo que vienen de serie con el IDE, y **Proyectos**, que permite acceder a tus propios sketches guardados en las diferentes subcarpetas que hay dentro de la carpeta Arduino.

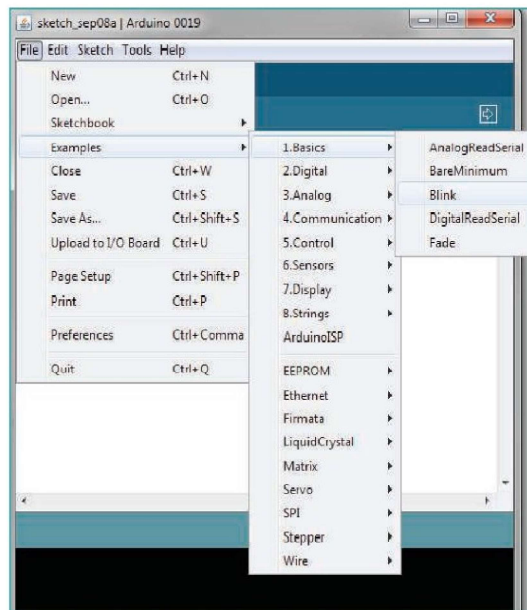


Figura 1.12.

- **Editar:** además de ofrecer acciones estándar, como deshacer y rehacer, cortar, copiar y pegar texto, seleccionar todo el texto,

o buscar y reemplazar texto, puedes ver otras acciones interesantes. Por ejemplo, gracias a **Copiar al foro**, puedes copiar el código de tu sketch al Portapapeles del sistema operativo en una forma que resulta especialmente adecuada para pegarlo directamente en el foro oficial de Arduino. También puedes **Copiar como HTML**, que copia de una manera especial para pegar en páginas web genéricas.

- **Programa:** en este menú se ofrece la acción de verificar/compilar el sketch, abrir la capeta donde está guardado el fichero .ino que se está editando en ese momento, añadir en una nueva pestaña un nuevo fichero de código a tu sketch e importar librerías.
- **Herramientas:** brinda diferentes herramientas, como la posibilidad de autoformatear el código para hacerlo más legible, guardar una copia de todos los sketches del proyecto actual en formato .zip, abrir el monitor serie, etc. Otras herramientas mucho más avanzadas son, por ejemplo, la entrada **Programador**, que puedes usar para seleccionar un programador externo y grabar el sketch en memoria a través de dicho programador; o **Quemar bootloader**, útil cuando quieras grabar un nuevo bootloader en el microcontrolador de la placa (Figura 1.13.).
- **Ayuda:** desde este menú puedes acceder a varias secciones de la página web oficial de Arduino que contienen diferentes artículos, tutoriales y ejemplos de ayuda. No se necesita Internet para consultar estas secciones, ya que la documentación se descarga junto con el propio IDE, por lo que su acceso se realiza en forma local.
- **Monitor Serial:** es una ventana del IDE que permite, desde tu computadora, enviar y recibir datos textuales a la placa Arduino usando el cable USB (el cual utiliza la conexión serie). Para enviar datos, simplemente hay que escribir el texto deseado en la caja de texto que aparece en su parte superior y presionar el botón **Enviar**. Por otro lado, los datos recibidos provenientes de la placa se mostrarán en la sección central del **Monitor serial** (Figura 1.14.).