

Arduinio

Test Placa

LunikSoft

Fco. Javier Andrade

<http://www.luniksoft.info>

El autor no se responsabiliza del contenido. No asume ninguna responsabilidad, por cualquier inexactitud en la documentación, en los programas, en el cableado y diagramas, ni de los daños derivados de la utilización del material proporcionado.

INTRODUCCIÓN :

Arduino

.Arduino es una plataforma hardware y software libre, con micro controlador y un entorno de desarrollo cuyo objetivo es simplificar el desarrollo de proyectos.

El hardware se basa en los micro controladores Atmel AVR (Atmega168, Atmega328...).

Ventajas con otras plataformas :

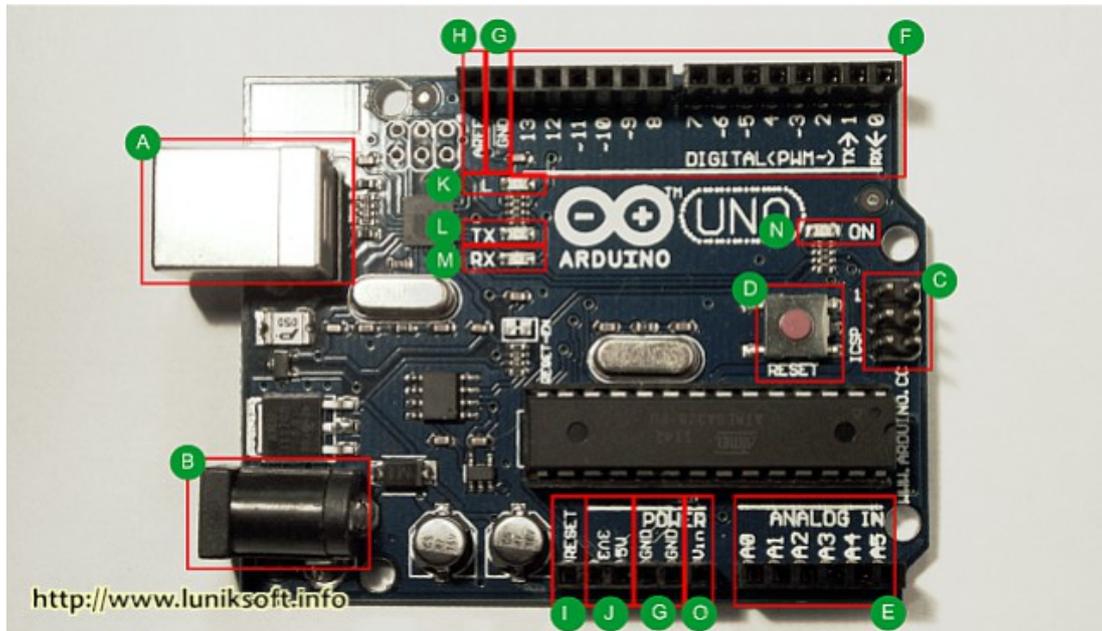
- Hardware ampliable y de Código abierto. Los planos están publicados bajo licencia Creative Commons. Cualquiera puede hacer mejoras o adaptaciones.
- Software ampliable a través de librerías C++ y también de Código abierto. Y si aun así se te queda pequeño podrías emplear AVR C , ya que el propio software Arduino está basado en el.
- Entorno de desarrollo simple. Ideal para principiantes y pensado para profesores.
- Multiplataforma : Windows, Macintosh OSX y Linux
- Asequible. Placas por menos de 30 €
- Multitud de módulos : Wifi, multitud de sensores, manejo de motores,... Muchos de estos módulos también open hardware.
- Se puede alimentar con una fuente externa (7 - 12v) o por un simple puerto USB (5v) (pues su consumo es ridículo).

Hay multitud de versiones de la placa, pero casi el 100% de los modelos tienen entradas analógicas, entradas-salidas digitales y digitales PWM. Lo que le permite ser una placa muy interesante para proyectos DIY (*Do It Yourself*, Hágalo Usted Mismo)



Algunos datos (Arduino UNO)

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz



- A. Conector USB. Este puerto contiene un protección contra cortocircuitos para proteger el puerto USB del ordenador (aunque estos ya tengan su protección no viene de mas una protección extra). Si se consumen mas de 500 mA el fusible saltaría. Bastaría con eliminar el cortocircuito y el fusible se autoarmaría automáticamente.
- B. Conector de Alimentación.
- C. Conector SPI:



- D. Botón de Reset
- E. Entradas Analógicas
 - Los pines A4 (SDA) y A5 (SCL). Soportan la comunicación TWI utilizando la librería Wire.
- F. Entradas/Salida Digitales.
 - De las cuales son PWM las señaladas con ~ (en este caso son 3, 5, 6, 9, 10 y 11).
 - Los Pines 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) proporcionan comunicación SPI.

- En el pin 13 hay integrado un led conectado. Cuando se pone esta salida a high (se enciende) y en low (se apaga). Ver K.
- Los pines 0 y 1 ademas son usados para recibir (RX) transmitir (TX) datos a través de puerto serie TTL. Estos pines están conectados a los pines correspondientes del chip FTDI USB-to-TTL.
- Los pines 2 y 3 sirven también como interruptores externos. Ver función `attachInterrupt()`

G. Pines de toma de tierra

H. Voltaje de referencia para la entradas analógicas. Ver función `AnalogReference()`

I. Reset. Suministrar un valor LOW(0V) para reiniciar el microcontrolador.

J. Salida de 5 y 3.3 voltios generada en el chip FTDI integrado en la placa. La corriente máxima soportada 50mA.

K. Led integrado en el pin 13.

L. Led indicador de TX Transmisión Serie.

M. Led indicador de RX Recepción Serie.

N. Led indicador de encendido.

O. Conector para alimentación externa. 7v-12v

El Arduino uno se puede alimentar a través de una fuente de alimentación externa o por el puerto USB. Si optamos por la fuente externa este puede ser un adaptador de corriente o una simple batería. La alimentación se realizar por el jack de 2.1mm (B) o por Vin (O). Hay que tener en cuenta que aunque se puede alimentar con una tensión de entre 5 y 20 voltios es recomendable limitar ese rango entre los 7 y 12 voltios, pues una alimentación inferior a los 7 podría provocar inestabilidad a la placa y que en el pin de 5V no llegara a esa tensión. Y en caso de ser superior a los 12v el regulador voltaje se podría sobre calentar y dañar la placa.

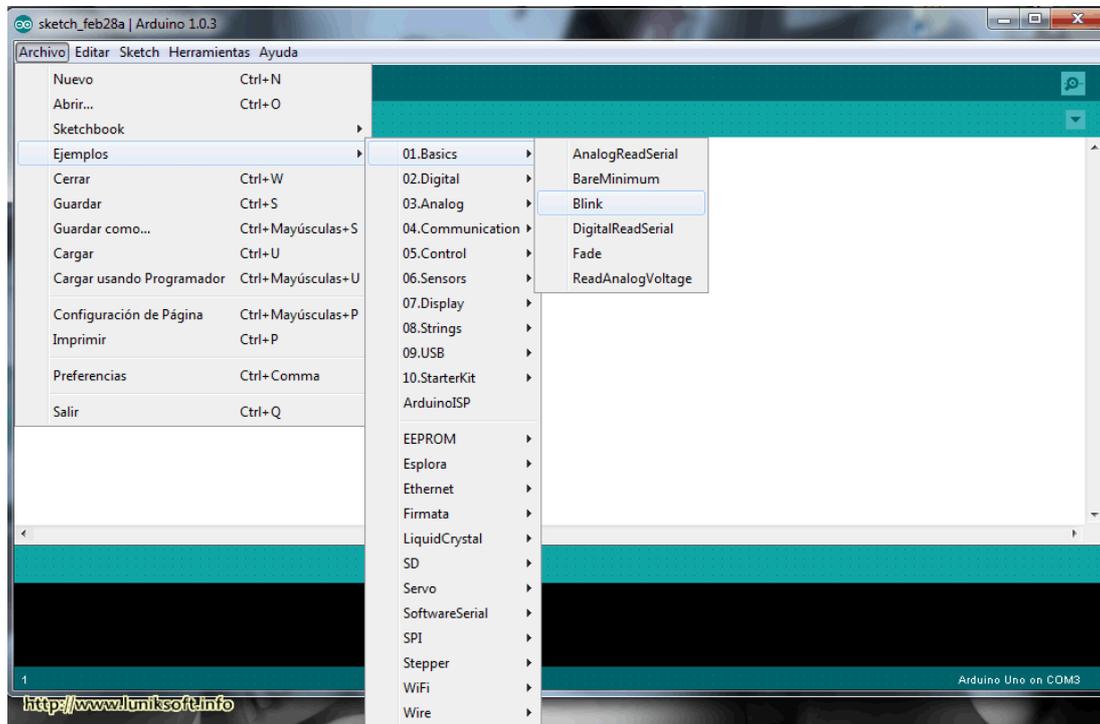
Test Placa

Una vez que nos hemos hecho con una placa Arduino lo primero que vamos a hacer es ver que funciona correctamente. Para ello vamos a hacer el que podría ser el "hola mundo" en un Arduino.

Recordar que para bajar el software para programar el Arduino basta con ir a la página oficial del proyecto (<http://www.arduino.cc/>)

Lo descomprimos (en mi caso la versión Windows en E:\arduino-1.0.3)

Para hacer este primer test nos bastará con cargar uno de los ejemplos que trae el IDE del Arduino:



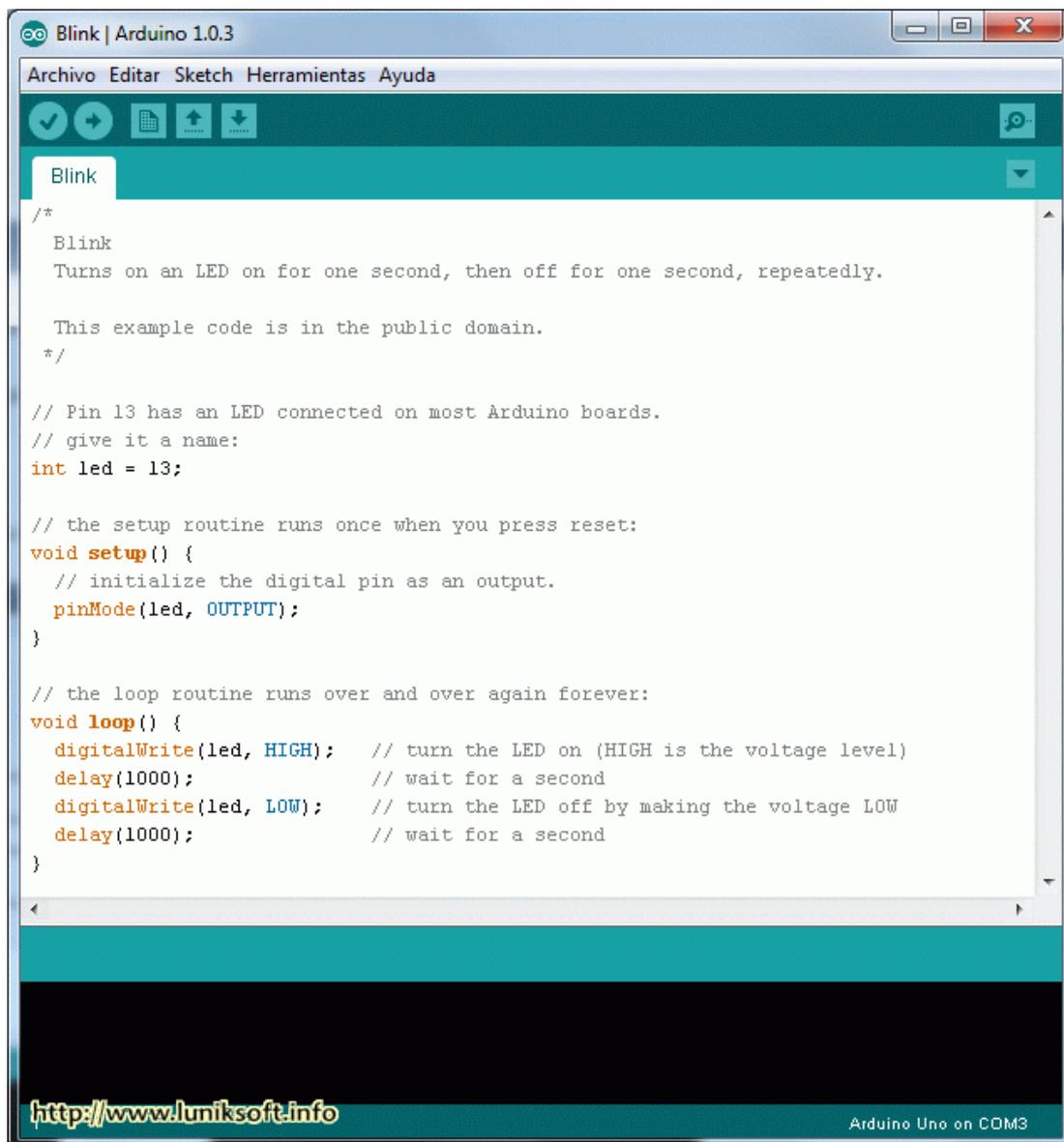
COMPONENTES

- 1 x Arduino Uno (o compatible)
- 1 x En algún caso una resistencia y un led.

CODIGO FUENTE (SKETCH)

```
/*  
  Hacer parpadear un led(autonomamente)  
*/  
  
int led = 13; // Pin donde conectar el LED. Emplear 13 si se quiere  
emplear el led incorporado en algunas placas arduino  
  
//Rutina setup que se ejecuta cuando se presiona reset  
void setup() {  
  // Inicializar el pin como de salida  
  pinMode(led, OUTPUT);  
}
```

```
//Rutina principal que se ejecuta indefinidamente
void loop() {
digitalWrite(led, HIGH);    // Enciende el led. Haciendo el voltaje
HIGH
delay(1000);                // espera un segundo
digitalWrite(led, LOW);    // Apaga el led. Haciendo el voltaje LOW
delay(1000);                // espera otro segundo
}
```

The image shows a screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 1.0.3". The menu bar includes "Archivo", "Editar", "Sketch", "Herramientas", and "Ayuda". The toolbar contains icons for file operations and execution. The main editor area displays the following code:

```
/*
  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
}
```

At the bottom of the IDE, there is a status bar with the URL <http://www.lunisoft.info> on the left and "Arduino Uno on COM3" on the right.

Descripción del código :

```
int led = 13;
```

La primera parte del código simplemente crea una variable led de tipo entero y que le asigna el valor 13. Esta variable es la que indica sobre que pin esta conectado el led. En este caso el pin 13 ya lleva incorporado una resistencia y un led. Si quisiéramos emplear otro pin habría que cambiar el valor 13 por otro y en el Arduino conectar un led con su correspondiente resistencia (entre 220 y 500 ohmios, valor que dependerá del consumo del diodo).

Función **void setup()**. Esta función se llama cuando se pulsa reset (o cuando se arranca el arduino), es la función de configuración donde deberemos establecer los valores y estados iniciales.

Como se puede ver en el caso que nos ocupa simplemente indica que el pin led (13) sera de salida.

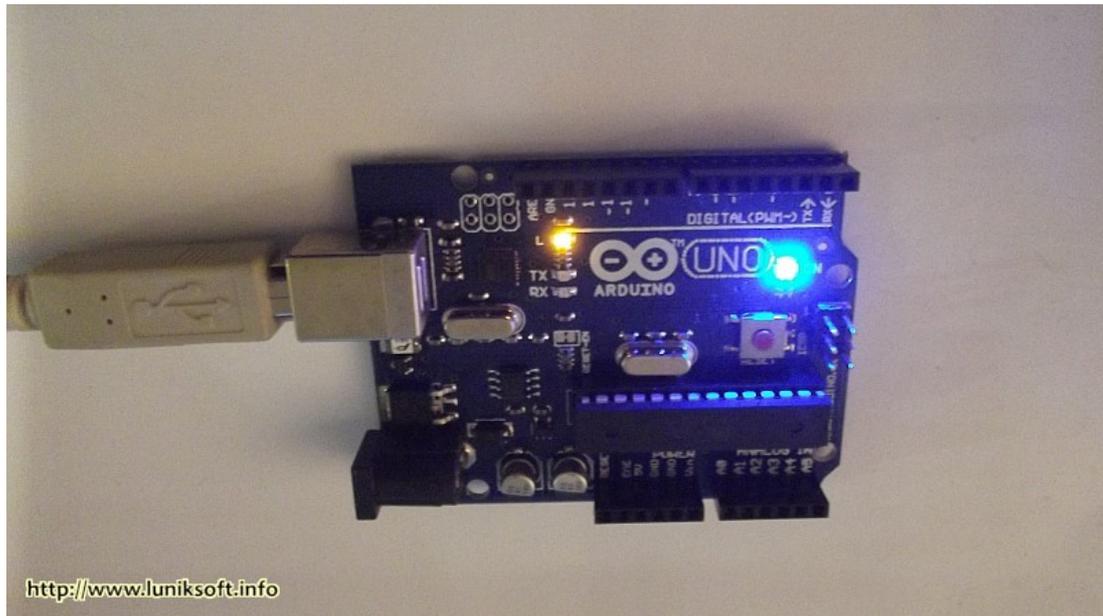
Función **void loop()**. Esta es la función que es ejecutada por el Arduino y se llama indefinidamente (en forma de bucle infinito). El código que nos encontramos también es simple.

1. Primero manda una señal alta para encender el led situado en el pin 13.
2. Luego hace una pausa de 1 segundo
3. Sitúa la señal del pin 13 en baja, para apagar el led.
4. Vuelve a hacer una pausa de 1 segundo.

Como se indico esta función es cíclica (se llama continuamente) por lo que veremos que el led se enciende y apaga repetidamente.

CIRCUITO

Aprovecho ya la misma conexión USB que me sirvió para programar para alimentarlo.



En la imagen parpadea el diodo smd naranja que es el que esta conectado internamente al pin 13. El led azul nos indica que el arduino esta encendido.

Resistencia en otro pin:

Para hacer el cálculo del valor de la resistencia hay que ver las características del fabricante del diodo led. Por ejemplo el led empleado es de 2v y 20mA. Hacemos el cálculo teniendo en cuenta los 5V que proporciona el Arduino.

$$5V \text{ (Arduino)} - 2v \text{ (caída tensión led)} = 3V$$

$$3V / 0.020 \text{ mA} = 150 \text{ ohmios (esta seria el valor de la resistencia a poner en serie junto al diodo)}$$

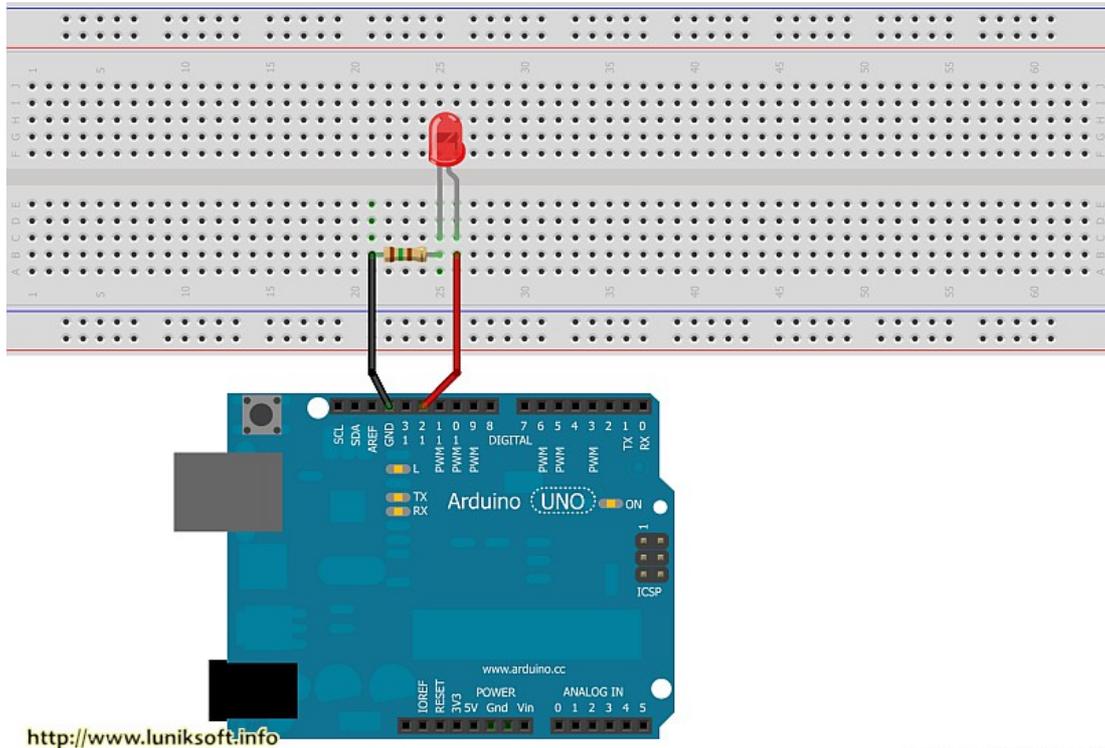
Cuidado con polaridad del diodo (la patilla mas larga suele ser el + y una muesca en el diodo indica -)

Lo conectamos a la placa a la salida 12 (el positivo) y a GND el negativo. Seguimos el resto de pasos, como se indica mas arriba teniendo en cuenta en cambiar el código antes de subir al Arduino. Las lineas a cambiar son

```
int led = 13;
```

```
por
```

```
int led = 12;
```



Made with  Fritzing.org