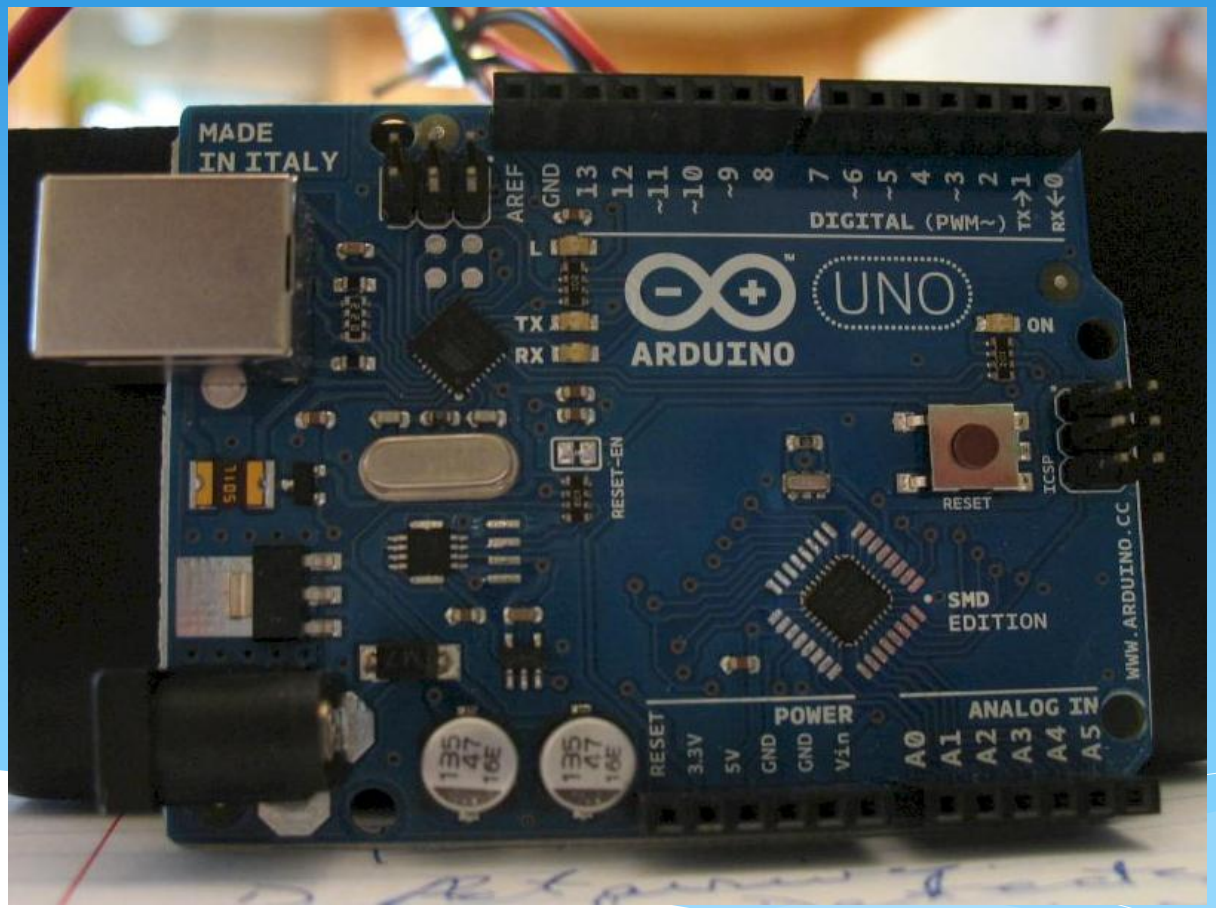
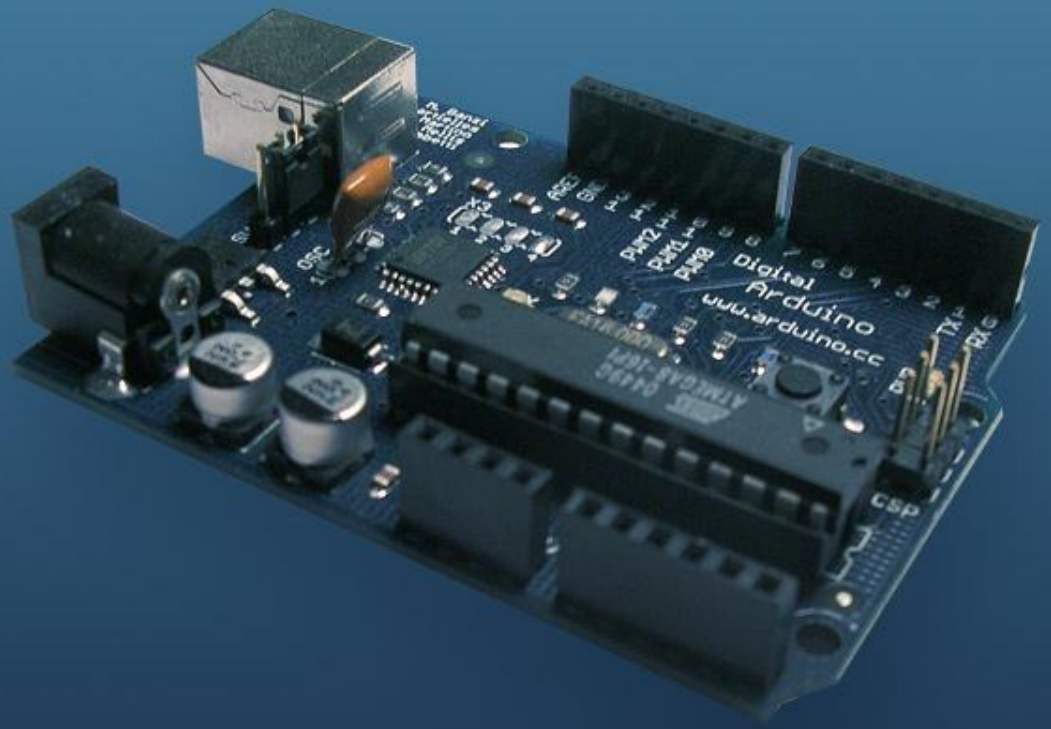
# Arduino arquitectura abierta




Jaime Cid y Fernando Reyes

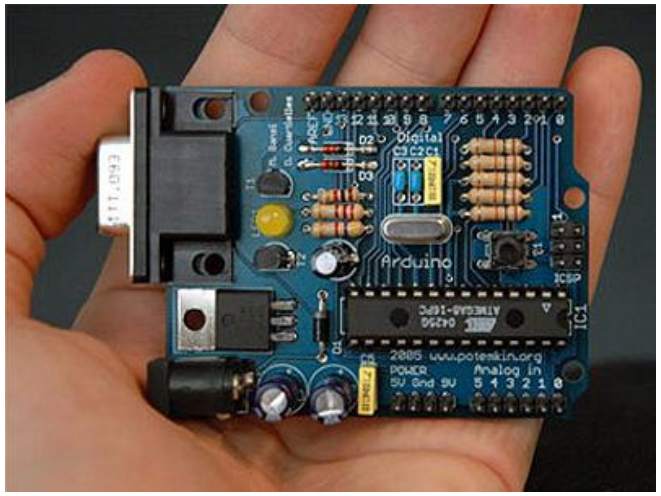
# Arduino

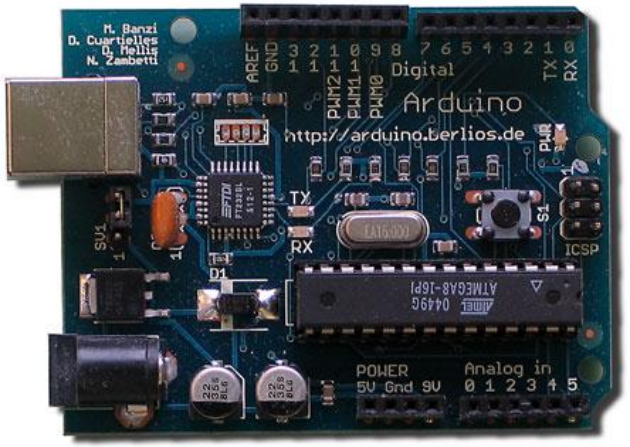
Physical Computing I/O board



# Arduino es una plataforma

Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos.



- Barato: Las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras. La versión menos cara del modulo Arduino puede ser ensamblada a mano, e incluso los módulos de Arduino preensamblados cuestan menos de 50$.

- Multiplataforma: El software de Arduino se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux. La mayoría de los sistemas microcontroladores están limitados a Windows.

- Entorno de programación simple y claro: El entorno de programación de Arduino es fácil de usar para principiantes, pero suficientemente flexible para que usuarios avanzados puedan aprovecharlo también. Para profesores, está convenientemente basado en el entorno de programación Processing, de manera que estudiantes aprendiendo a programar en ese entorno estarán familiarizados con el aspecto y la imagen de Arduino.

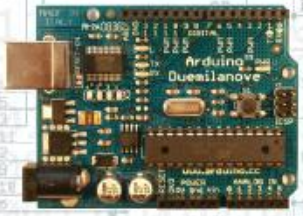
- Código abierto y software extensible: El software Arduino está publicado como herramientas de código abierto, disponible para extensión por programadores experimentados. El lenguaje puede ser expandido mediante librerias C++, y la gente que quiera entender los detalles técnicos pueden hacer el salto desde Arduino a la programación en lenguaje AVR C en el cual está basado. De forma similar, puedes añadir código AVR-C directamente en tus programas Arduino si quieres.

- Código abierto y hardware extensible: El Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, extendiéndolo y mejorándolo. Incluso usuarios relativamente inexpertos pueden construir la versión de la placa del módulo para entender como funciona y ahorrar dinero.

# Otros modelos oficiales:

SmartProjects (Italia)

Nano

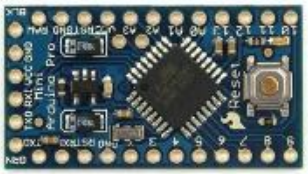


Duemilanove

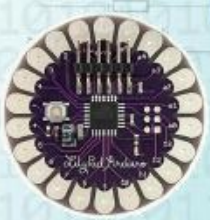
Diecimila

Sparkfun (EEUU)

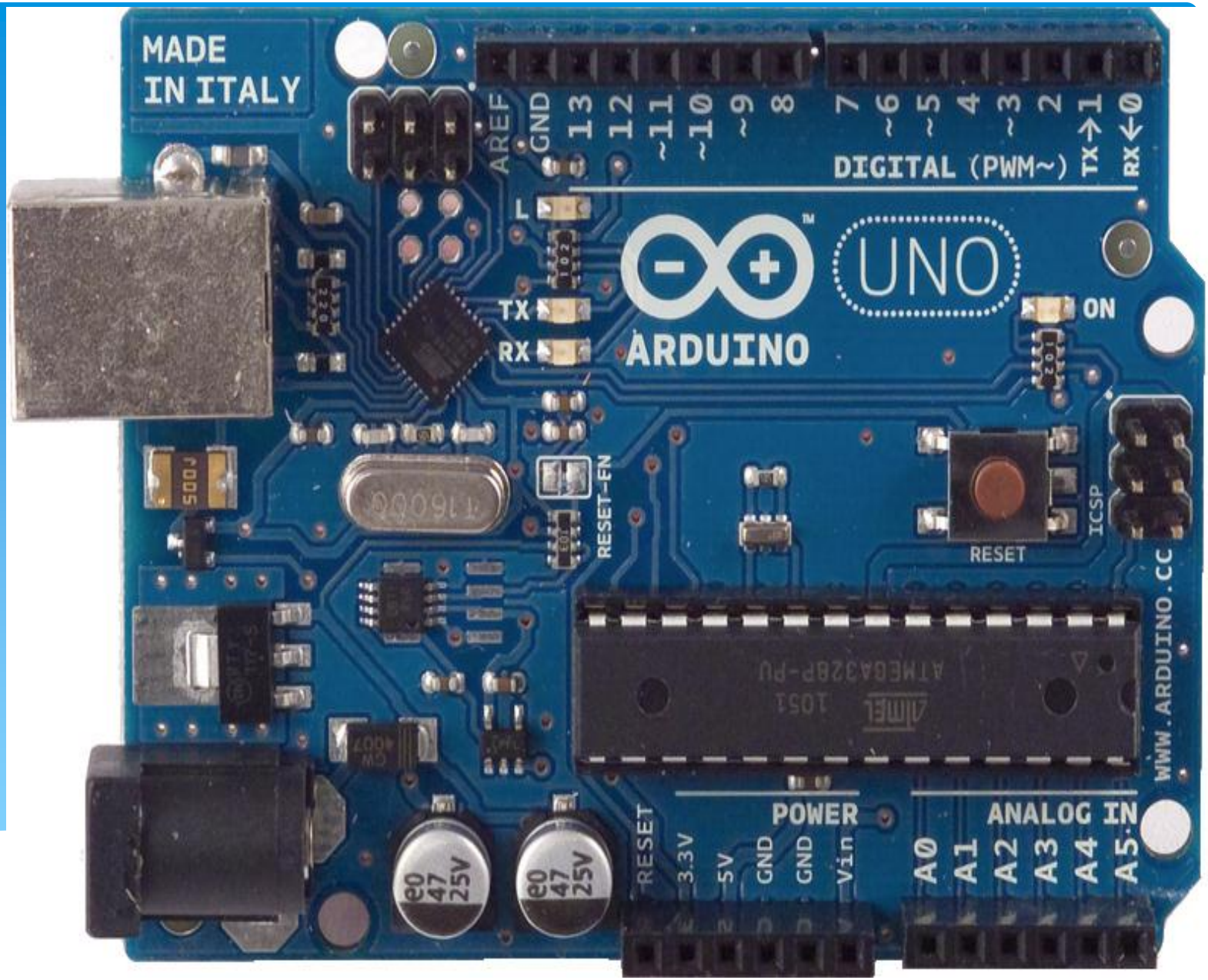
PRo



Pro Mini



Lilypad

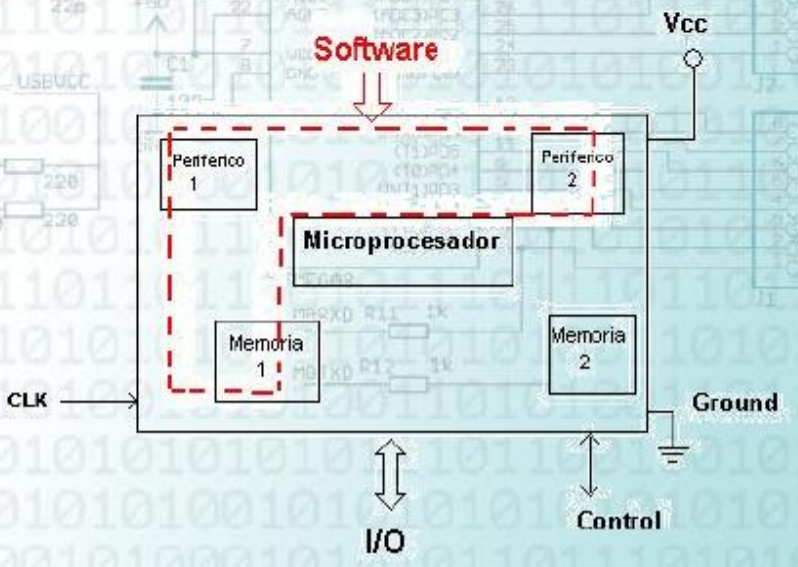

MADE
IN ITALY
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
RESET-EN
RESET
ICSP
WWW.ARDUINO.CC
POWER
ANALOG IN
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

# Características

| | |
|---|---|
| Microcontrolador | ATmega328 |
| Voltaje de operación | 5V |
| Voltaje de entrada (recomendada) | 7-12V |
| Voltaje de entrada (límites) | 6-20V |
| Pines Digitales E/S | 14 (de los cuales 6 proveen salidas PWM ) |
| Pines de entradas Analógicas | 6 |
| Corriente DC por pin E/S | 40 mA |
| Corriente DC para pin 3.3V | 50 mA |
| Memoria Flash | 32 KB de los cuales 0.5 KB son usados para bootloader |
| SRAM | 2 KB (ATmega328) |
| EEPROM | 1 KB (ATmega328) |
| Frecuencia del reloj | 16 MHz |

# ¿Qué es un microcontrolador (µC)?

- Circuito integrado con las 3 unidades funcionales de una computadora:
  - CPU (Unidad central de procesamiento)
  - Memoria
  - Periféricos de I/O



## Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips $I^2C$ compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.75 µA (Including 32 kHz RTC)

ATMEL

# 8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

## ATmega48PA
## ATmega88PA
## ATmega168PA
## ATmega328P

**Figure 2-1.** Block Diagram

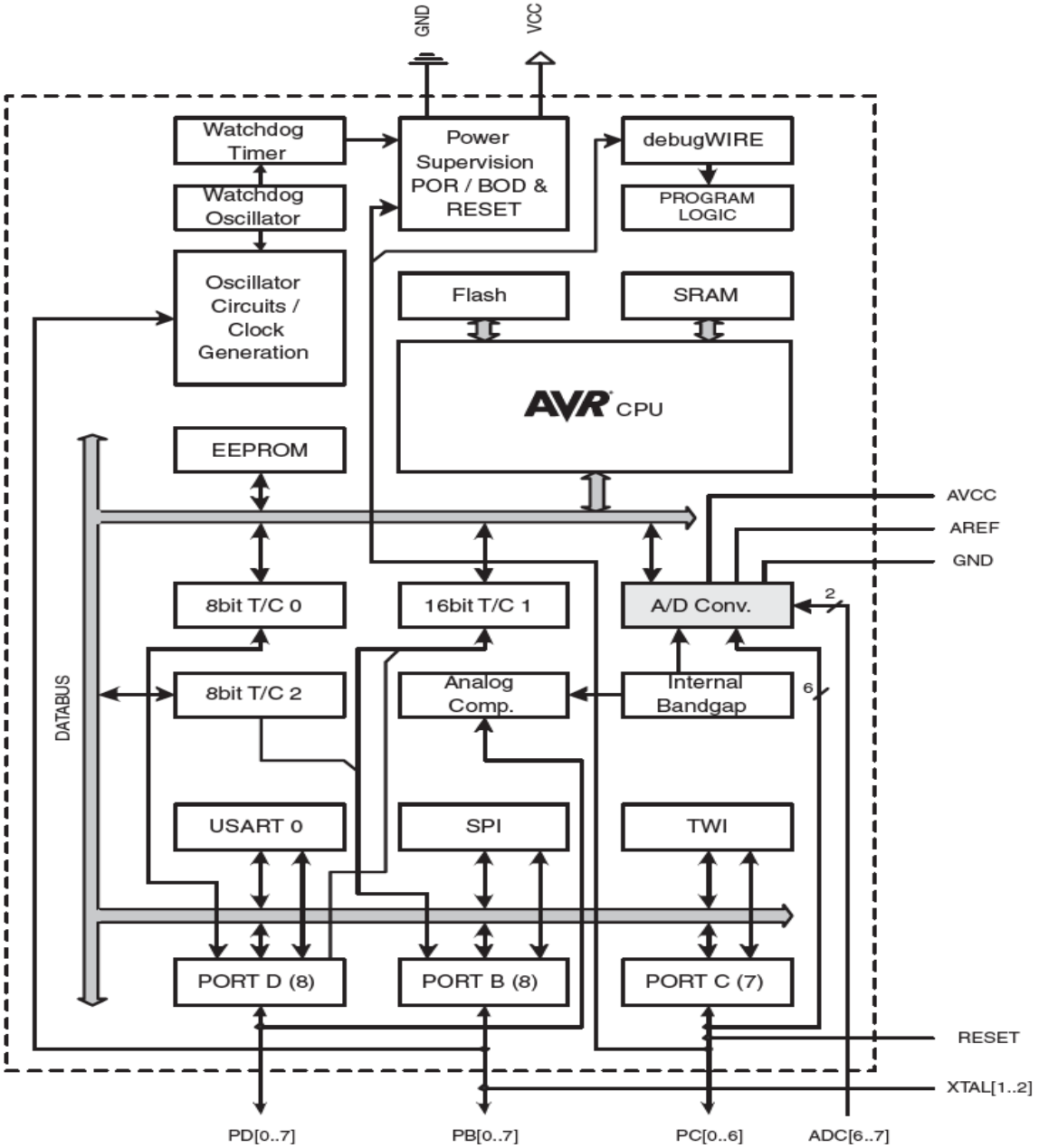


**Figure 6-1.** Block Diagram of the AVR Architecture

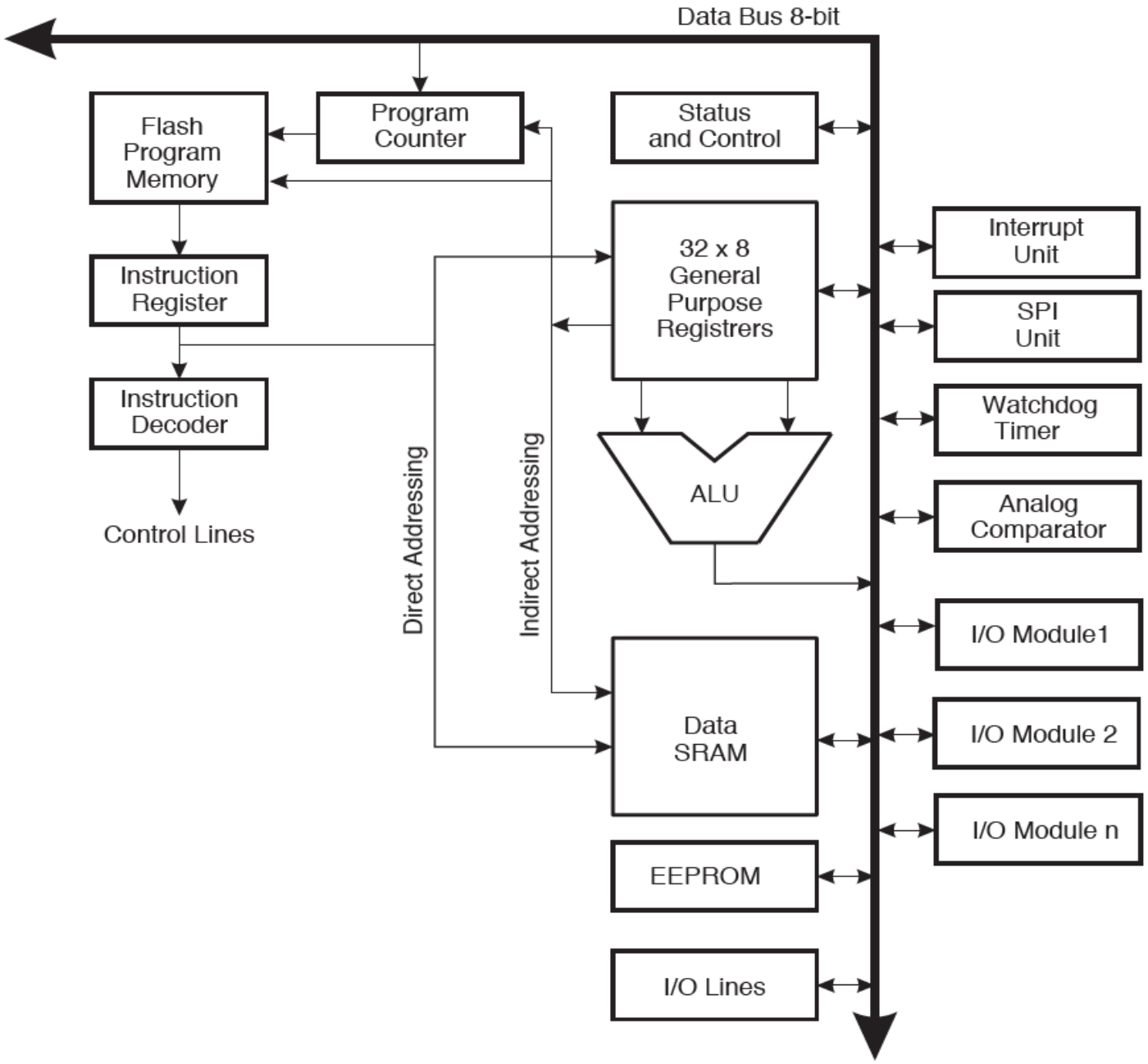
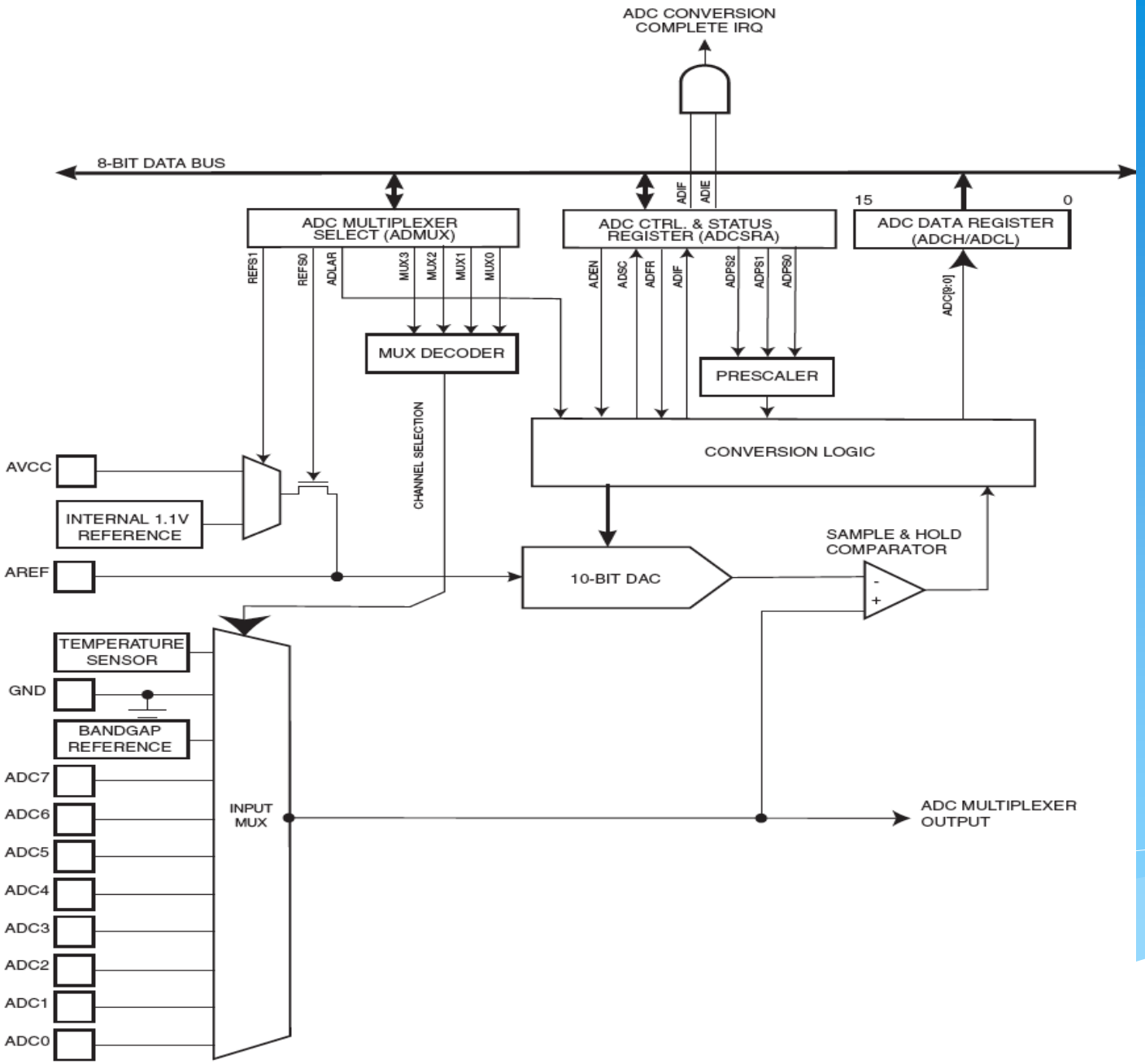


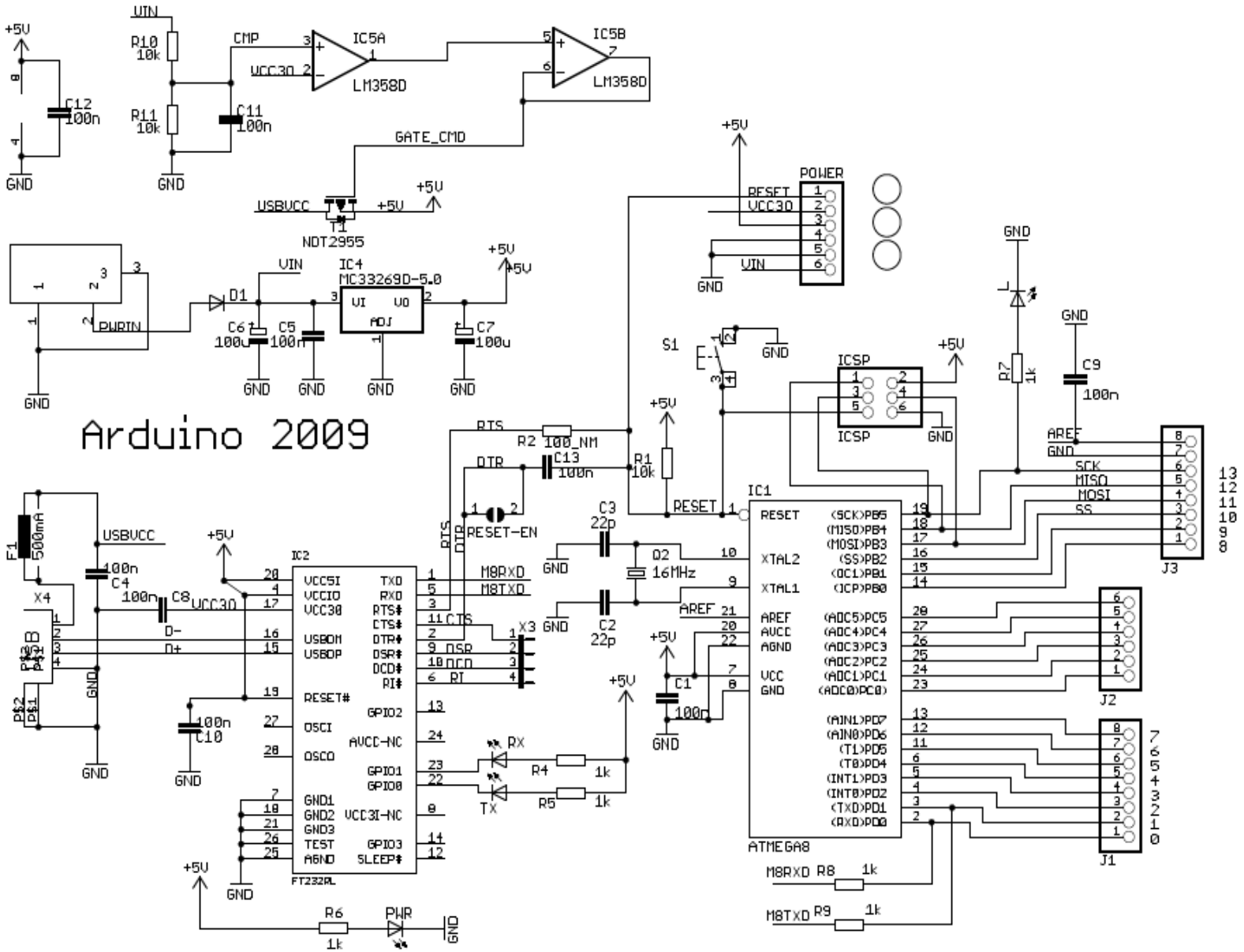
**Figure 23-1.** Analog to Digital Converter Block Schematic Operation,



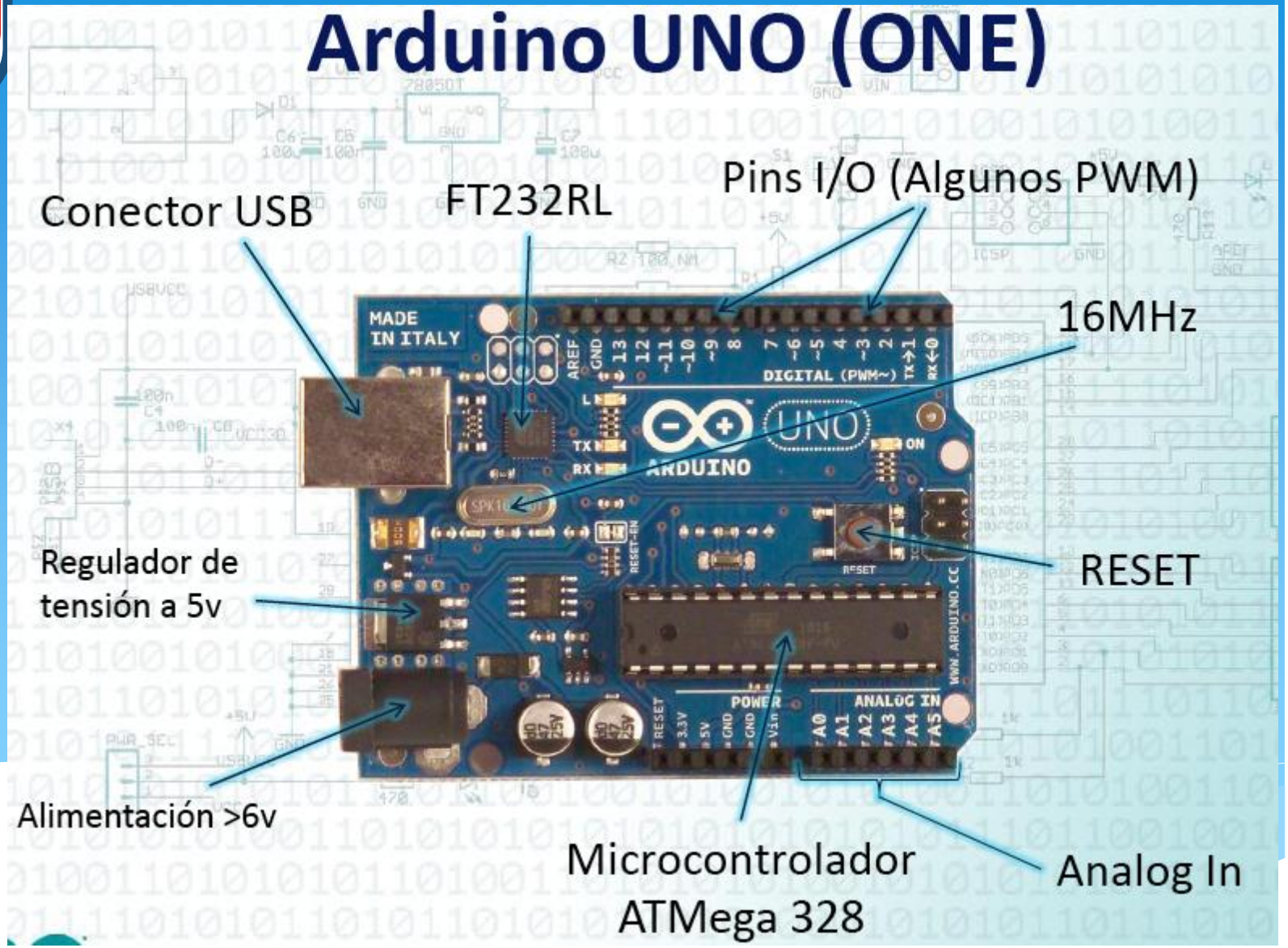
# Proyecto Arduino

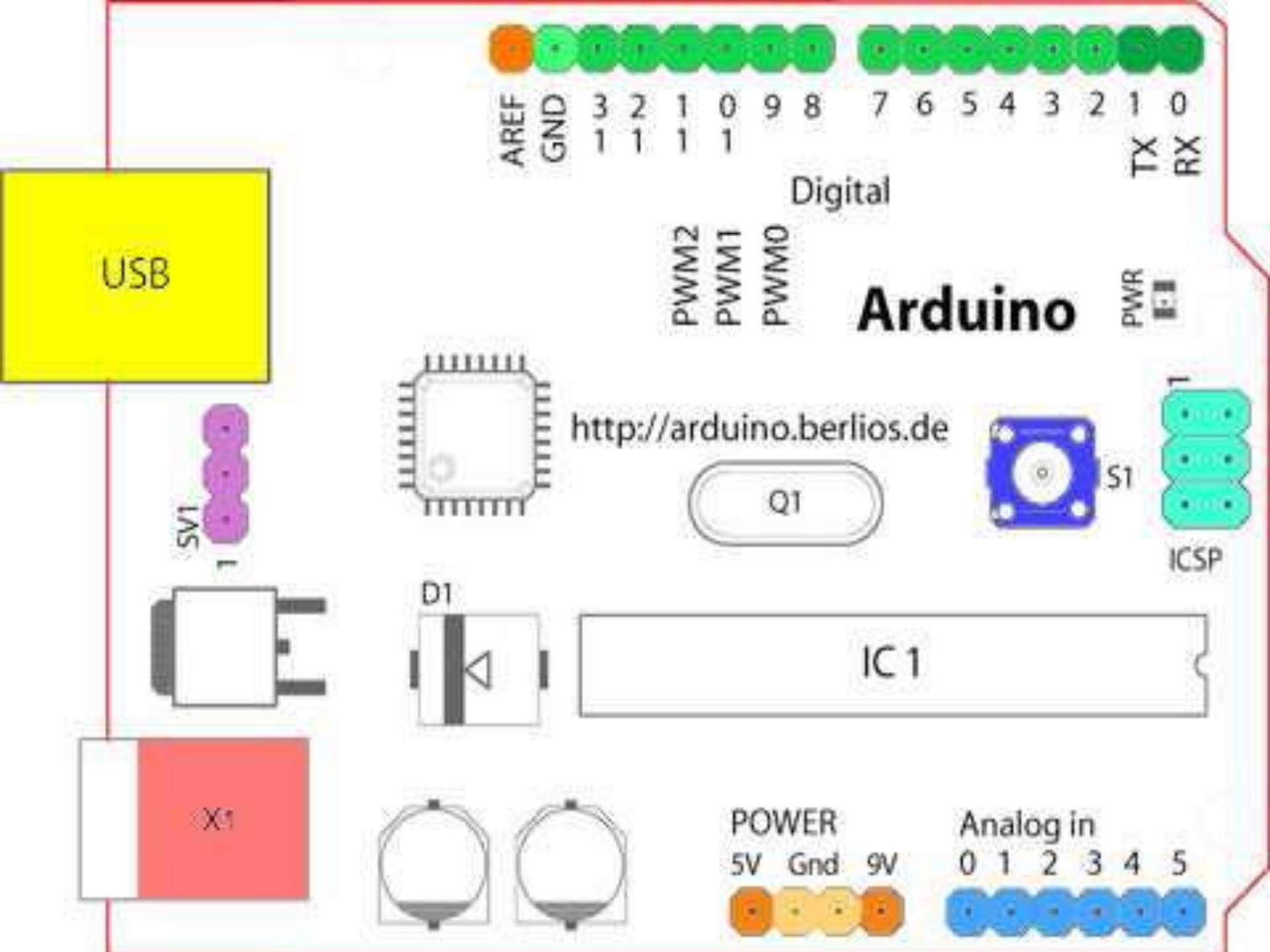
- Plataforma de  hardware y software libre (CC).
- Formado por:
  - Microcontrolador Atmel (AVR)
  - Circuito que facilita el uso:
    - USB
    - Pins externos.
    - Cristal cuarzo 16MHz
- http://www.arduino.cc/es/


MADE
IN ITALY
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
RESET-EN
RESET
ICSP
WWW.ARDUINO.CC
POWER
ANALOG IN
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5


Arduino 2009
IC5A
IC5B
LM358D
GATE_CMD
NDT2955
IC4
MC33269D-5.0
POWER
ICSP
RESET-EN
IC1
ATMEGA8
IC2
FT232RL
J1
J2
J3
16MHz
500mA

# Arduino UNO (ONE)




AREF
GND
3
2
1
0
9
8
7
6
5
4
3
2
1
0
TX
RX
Digital
PWM2
PWM1
PWM0
PWR
Arduino
USB
http://arduino.berlios.de
Q1
S1
ICSP
SV1
D1
IC 1
X1
POWER
5V
Gnd
9V
Analog in
0
1
2
3
4
5

# Software de Arduino

- Escrito en Java y basado en Processing, avr-gcc y otros programas de código abierto.
- Librerías de clases:
- IDE:
  - **Multiplataforma**
  - Se programa en **C/C++**
- Descarga: http://arduino.cc/es/Main/Software

# Antes de comenzar

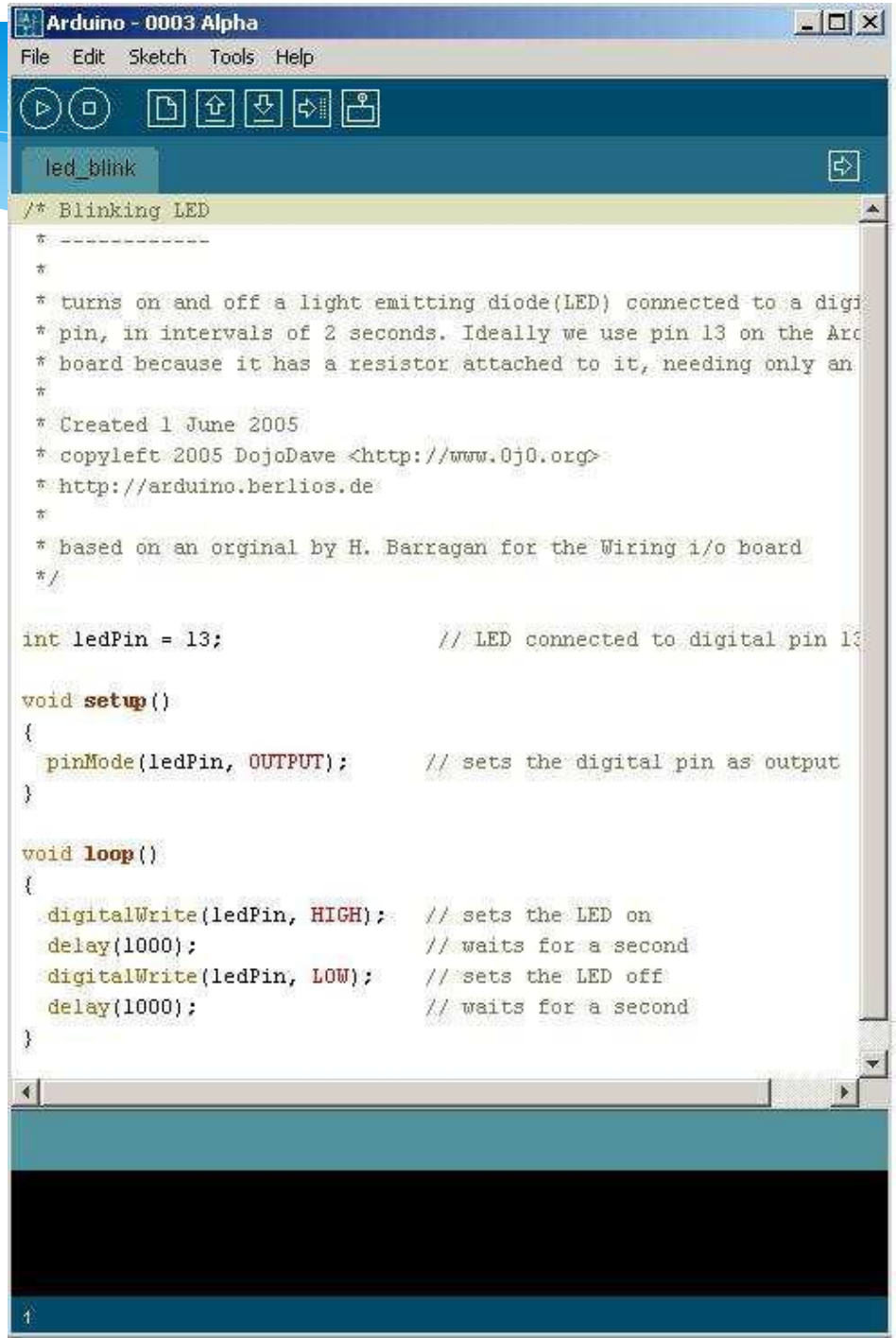
* Descarga el programa de www.arduino.cc
* Descomprime la carpeta
* Conecta el Arduino
* Instala los drivers FTDI
* Anota el puerto COM del Arduino

## SOFTWARE

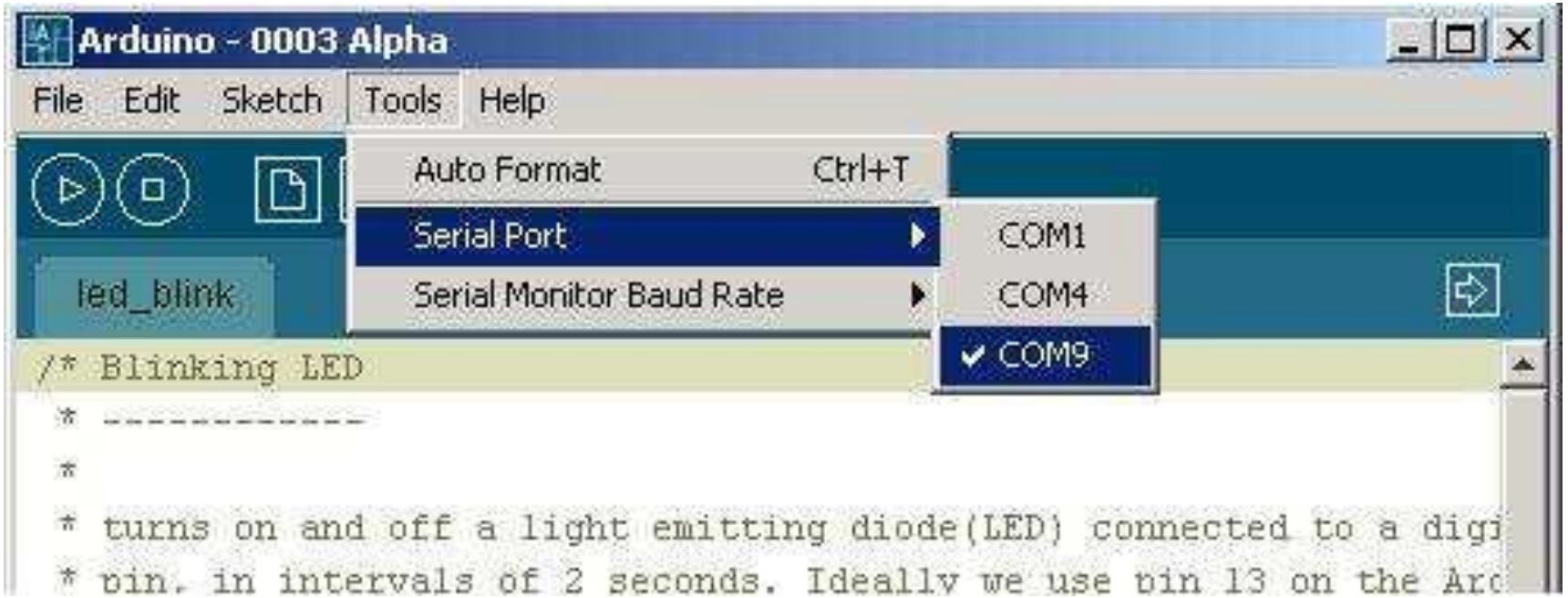
ENTORNO DE DESARROLLO
Para programar la placa es necesario descargarse de la página web de Arduino

(http://www.arduino.cc/en/Main/Software) el entorno de desarrollo (IDE).

En caso disponer de una placa USB es necesario instalar los drivers.

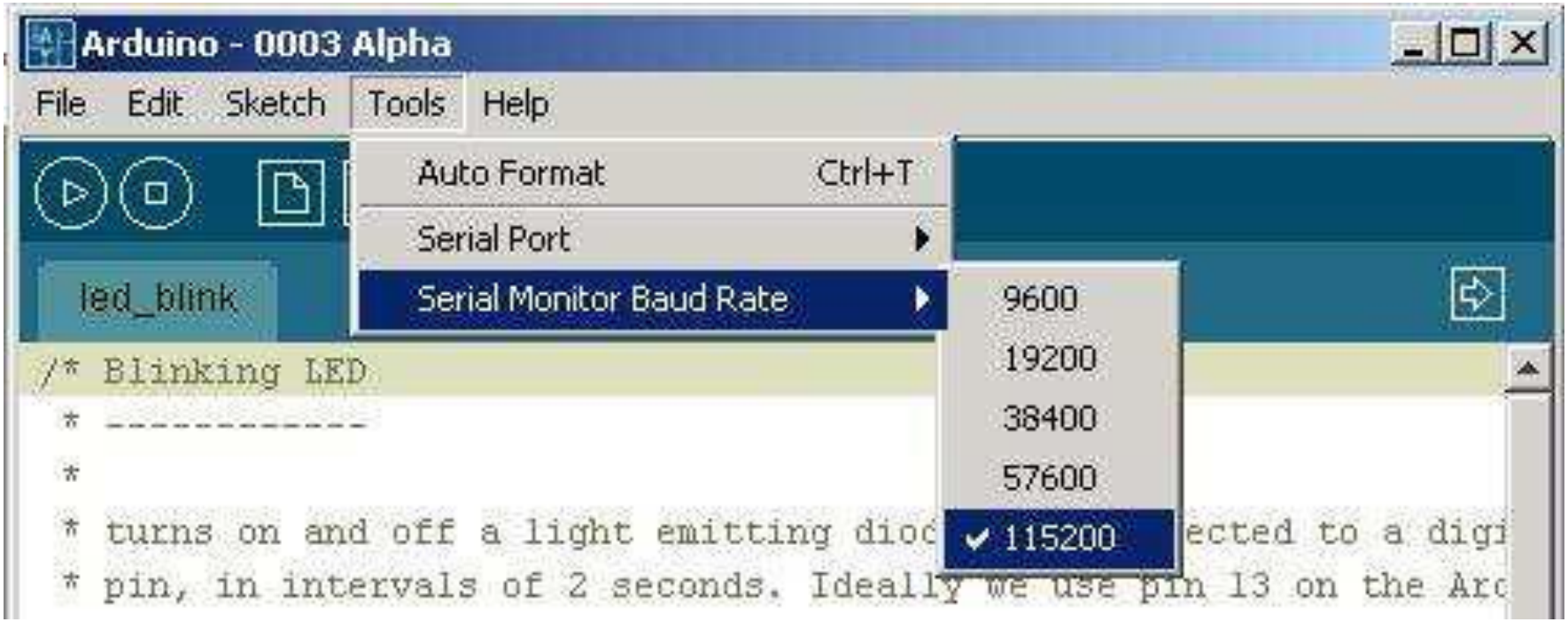


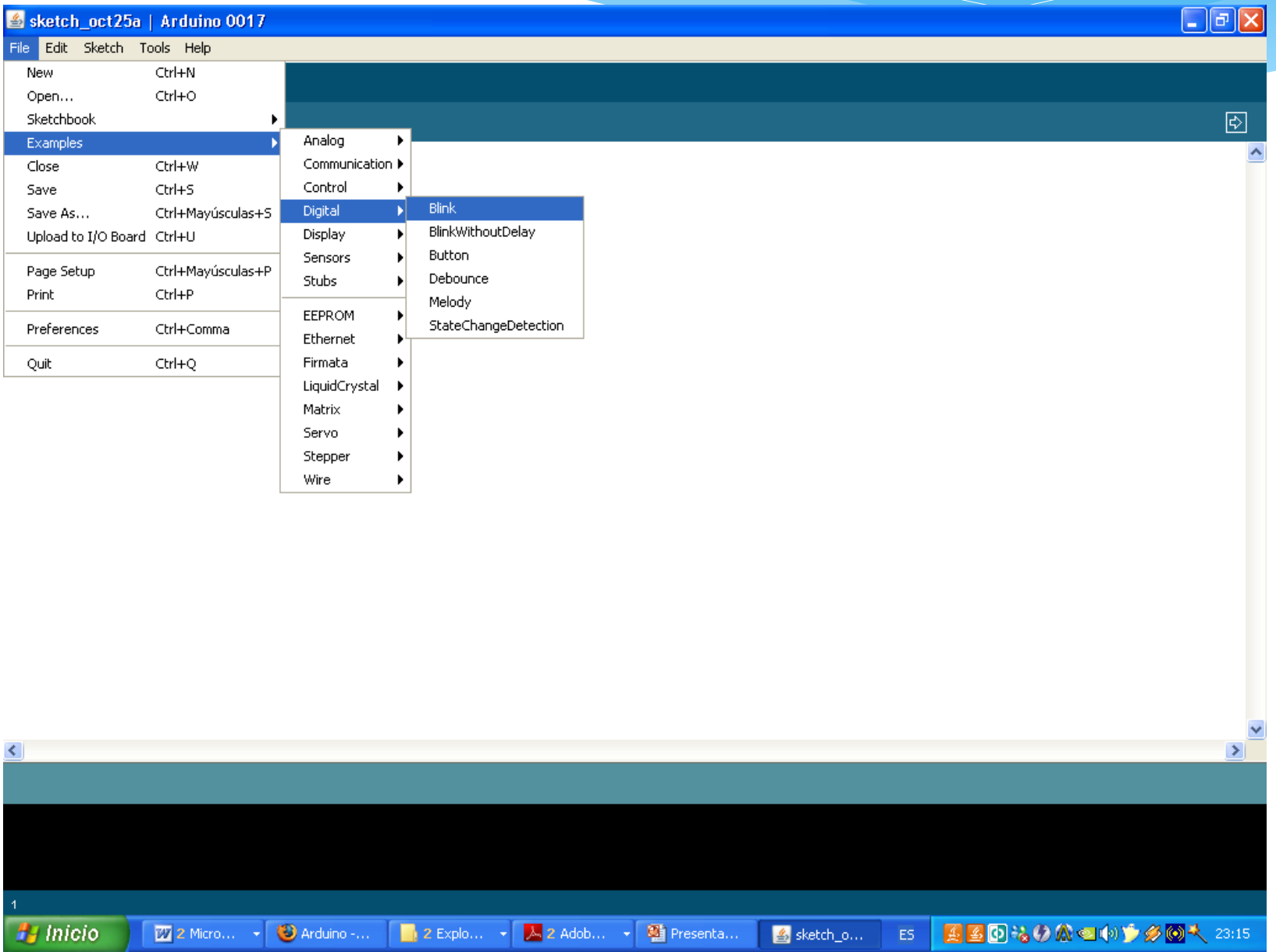
# *Configuración del puerto serie*

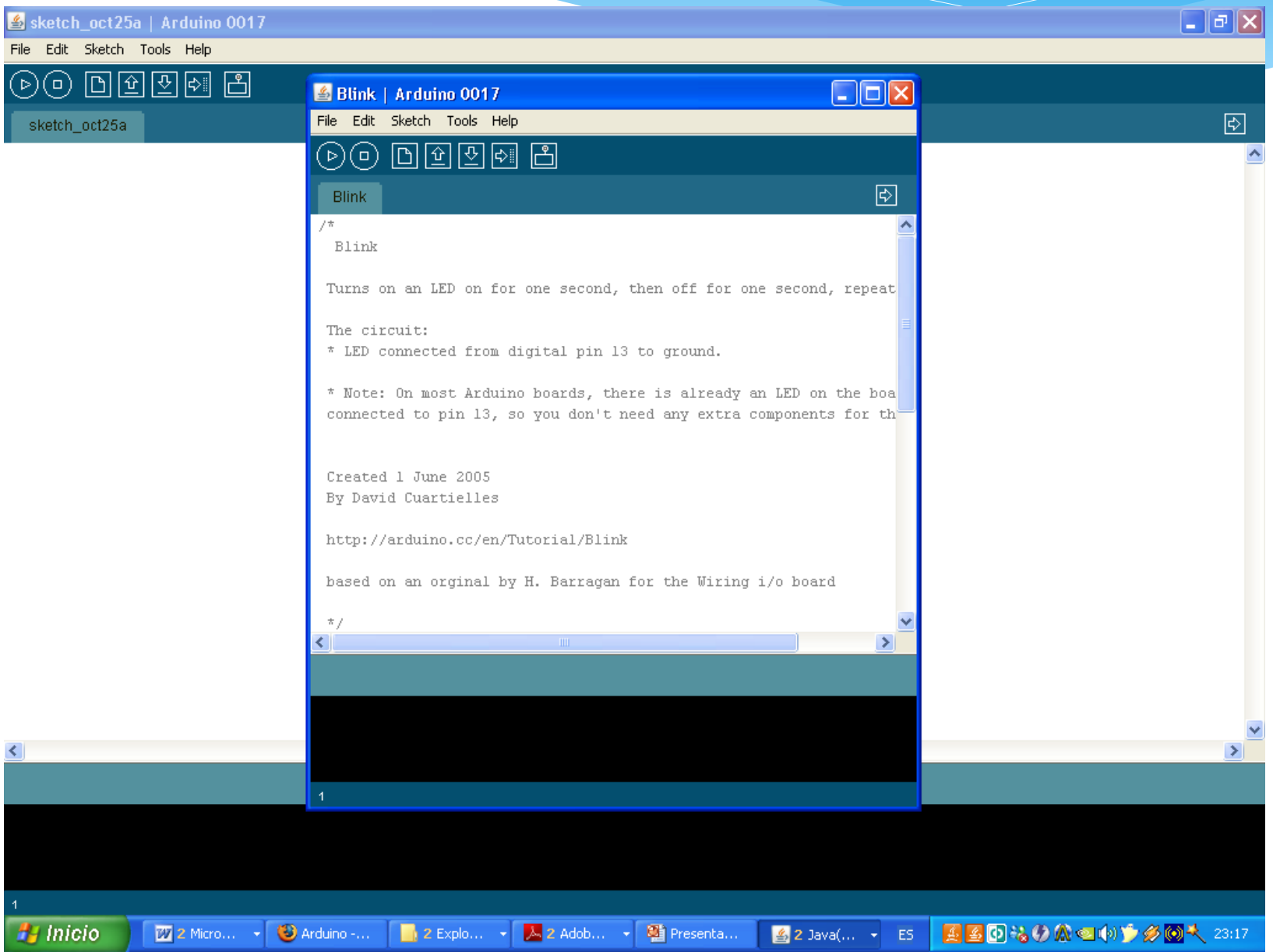


En Windows, si desconocemos el puerto al que está conectado nuestra placa podemos descubrirlo a través del “Administrador de dispositivos”

También debemos configurar la velocidad a la que la placa y el PC se comunican. Esto lo hacemos desde el menú “Serial Monitor Baud Rate”. El valor por defecto es de 115200 baudios



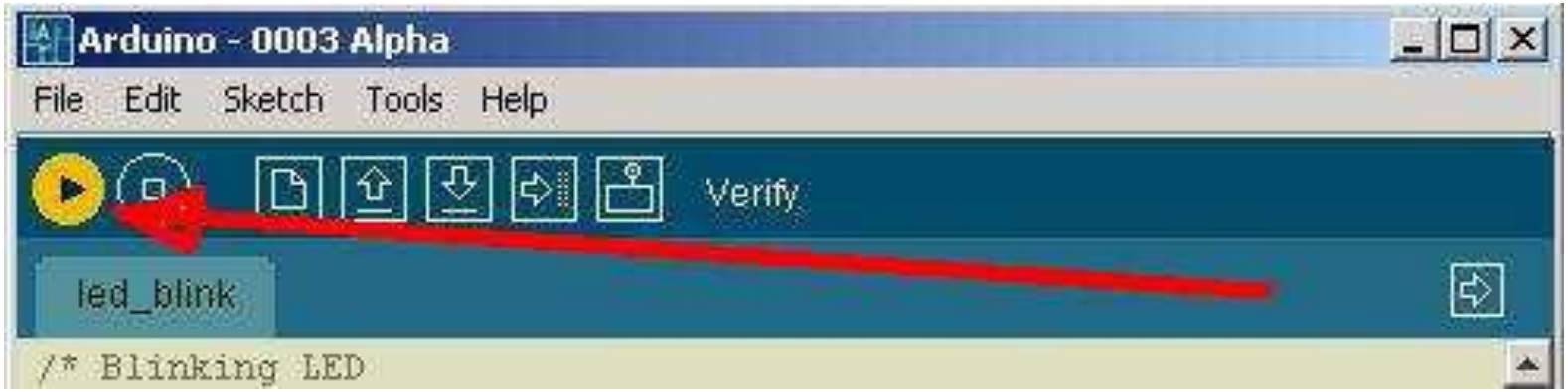

sketch_oct25a | Arduino 0017
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Mayúsculas+S
Upload to I/O Board Ctrl+U
Page Setup Ctrl+Mayúsculas+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Analog
Communication
Control
Digital
Display
Sensors
Stubs
EEPROM
Ethernet
Firmata
LiquidCrystal
Matrix
Servo
Stepper
Wire
Blink
BlinkWithoutDelay
Button
Debounce
Melody
StateChangeDetection
1
Inicio
2 Micro...
Arduino -...
2 Explo...
2 Adob...
Presenta...
sketch_o...
ES
23:15


sketch_oct25a | Arduino 0017
File Edit Sketch Tools Help
sketch_oct25a
Blink | Arduino 0017
File Edit Sketch Tools Help
Blink
/*
Blink
Turns on an LED on for one second, then off for one second, repeat
The circuit:
* LED connected from digital pin 13 to ground.
* Note: On most Arduino boards, there is already an LED on the boa
connected to pin 13, so you don't need any extra components for th
Created 1 June 2005
By David Cuartielles
http://arduino.cc/en/Tutorial/Blink
based on an orginal by H. Barragan for the Wiring i/o board
*/
1
1
Inicio
2 Micro...
Arduino -...
2 Explo...
2 Adob...
Presenta...
2 Java(...
ES
23:17

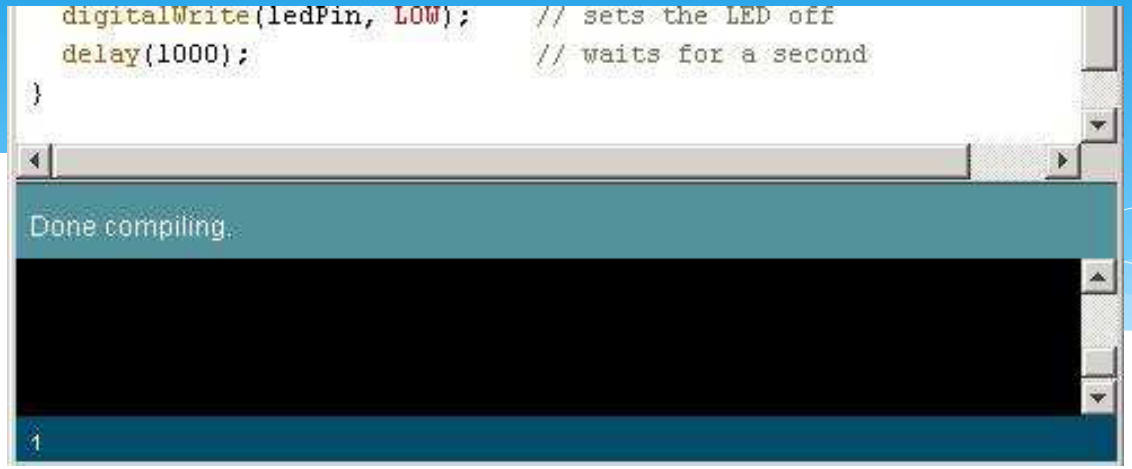
# SUBIENDO EL PROGRAMA DE EJEMPLO A LA PLACA ARDUINO

El ejemplo “led_blink” lo único que hace es parpadear un LED que esté colocado en el pin número 13 de la placa. Vamos a ver qué hay que hacer para subir el programa a la  placa Arduino.

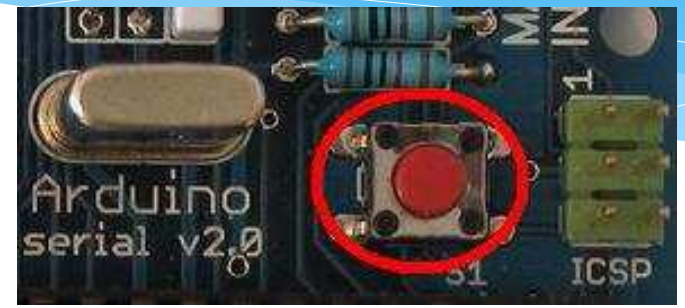
Primero comprobamos que el código fuente es el correcto. Para ello pulsamos el botón de verificación de código que tiene forma de triángulo inclinado 90 grados.



Si todo va bien deberá aparecer un mensaje en la parte inferior de la interfaz indicando “Done compiling”.



Una vez que el código ha sido verificado procederemos a cargarlo en la placa. Para ello tenemos que pulsar el botón de reset de la placa e inmediatamente después pulsar el botón que comienza la carga.

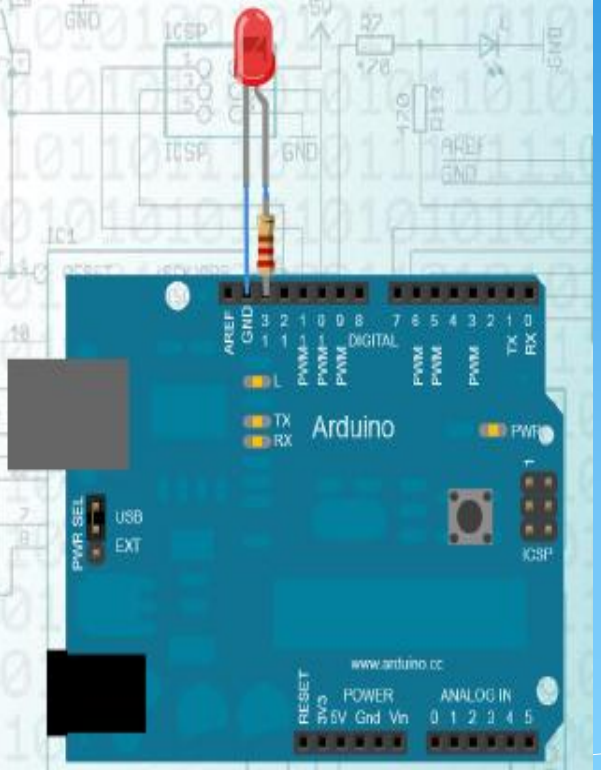



# “Hola Mundo”

```
void setup() {
  pinMode(13, OUTPUT);      //Pin digital 13 como output.
}

void loop() {
  digitalWrite(13, HIGH);   //Enciende el LED
  delay(1000);              //Espera 1000 milisegundos
  digitalWrite(13, LOW);    //Apaga el LED
  delay(1000);              //Espera 1000 milisegundos
}
```

# “Hola Mundo”

- Parpadeo de un LED
- Conectar el polo positivo del LED (el más largo) al pin 13 y el otro al GND.



# Código del primer programa

```
int ledPin =  13

void setup()   {
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  digitalWrite(ledPin, HIGH)
  delay(1000)
  digitalWrite(ledPin, LOW)
  delay(1000)
}
```


FCE
Facultad de Ciencias
de la Electrónica

# Analog In - Entradas análogas

**Pins análogos** son entradas análogas. Reciben tensiones **entre** 5V y 0 voltios. Los pines análogos, al contrario de los digitales, no necesitan ser declarados como modo INPUT o OUTPUT .

**Conversión análogo digital o ADC (analog to digital converter)**

convertir tensiones de 0 a 5 voltios en números enteros que van del 0 al 1023. En otras palabras, representa la información en números de 10 bits.

Son AI: resistencias variables **:** potenciómetro - fotocélula - FSR

Comandos básicos

**analogRead(pin),** Lee o captura el valor de entrada del especificado pin análogo. Placa Arduino realiza una conversión análoga a digital de 10 bits. Esto quiere decir que mapeará los valores de voltaje de entrada, entre 0 y 5 voltios,

a valores enteros comprendidos entre 0 y 1023.

# Analog In - Comunicación Serial

Esto nos permite comunicarnos con un computador e intercambiar datos o siemplemente monitorear que pasa con el sketch que esta corriendo en Arduino.

Podemos visualizar los datos procedentes de la tarjeta usando **Monitorización del Puerto Serie** (último botón a la derecha).

A veces nos interesa poder mandar datos de los sensores hacia el computador o incluso poder mandar comandos desde el PC a Arduino. Por ejemplo, si queremos visualizar, la lectura de un potenciómetro. Si la comunicación en serie esta activada, no se podrán usar los pines 0 y 1 como entrada/salida digital.

Es recomendable dejar tiempos de espera entre los envíos de datos para ambos sentidos (uso por ejemplo de un delay(10)) ,ya que se puede saturar o colapsar el puerto.

# Analog In - Comunicación Serial

* Comandos básicos:

  **Serial.begin**

  configura el puerto serie a una velocidad determinada. Se expresa en bits por segundo. Va en el **setup().**

  **Serial.print**()

  Descompone un número obtenido de un sensor, por ejemplo, en símbolos ASCII y los lanza uno a uno por el puerto serie en modo de caracteres ASCII. Por ejemplo, el número 100 se representará con la secuencia de números ASCII: 49, 48, 48.

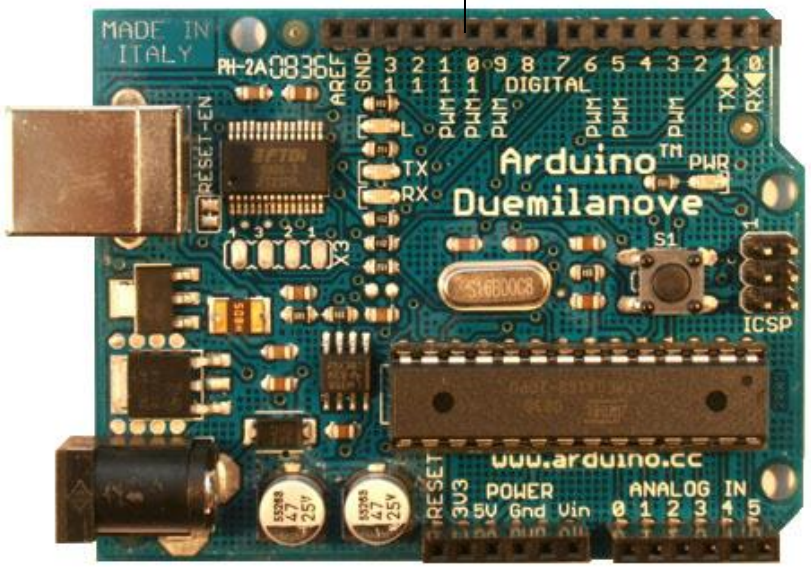
  **Serial.println():** lanza el valor 13, que quiere decir retorno de carro y el valor 10 que quiere decir fin o salto de línea por el puerto serie.

  **Serial.print(dato,BYTE):** lanza el valor dato por el puerto serie, en modo Byte o Binario.

## Analog Out - PWM

* PWM: 3, 5, 6, 9, 10 y 11.

PWM outputs de 8-bits utilizando función analogWrite().




Para que me sirve?

Manejo de intensidad: luz, T, sonido, etc.

# Analog Out - PWM

PWM (Pulse-width modulation) o serial de modulación por ancho de pulso.
Ancho de pulso, representa al ancho (en tiempo) del pulso con una modulación o cantidad de pulsos (estado on/off) por segundo.
El periodo es medido en segundos y la frecuencia en hertz

La señal PWM se utiliza como técnica para controlar circuitos analógicos, comúnmente usadas para el control de motores DC (si decrementas la frecuencia, la inercia del motor es mas pequeña y el motor se mueve mas lentamente), ajustar la intensidad de brillo de un LED, etc.

En Arduino la señal de salida PWM (pines 3, 5, 6, 9, 10 y 11) es una senal de frecuencia constante (30769 Hz) y que solo nos permite cambiar el "duty cycle" o el tiempo que el pulso esta activo (on) o inactivo (off), utilizando la función analogWrite().

## Analog Out - PWM

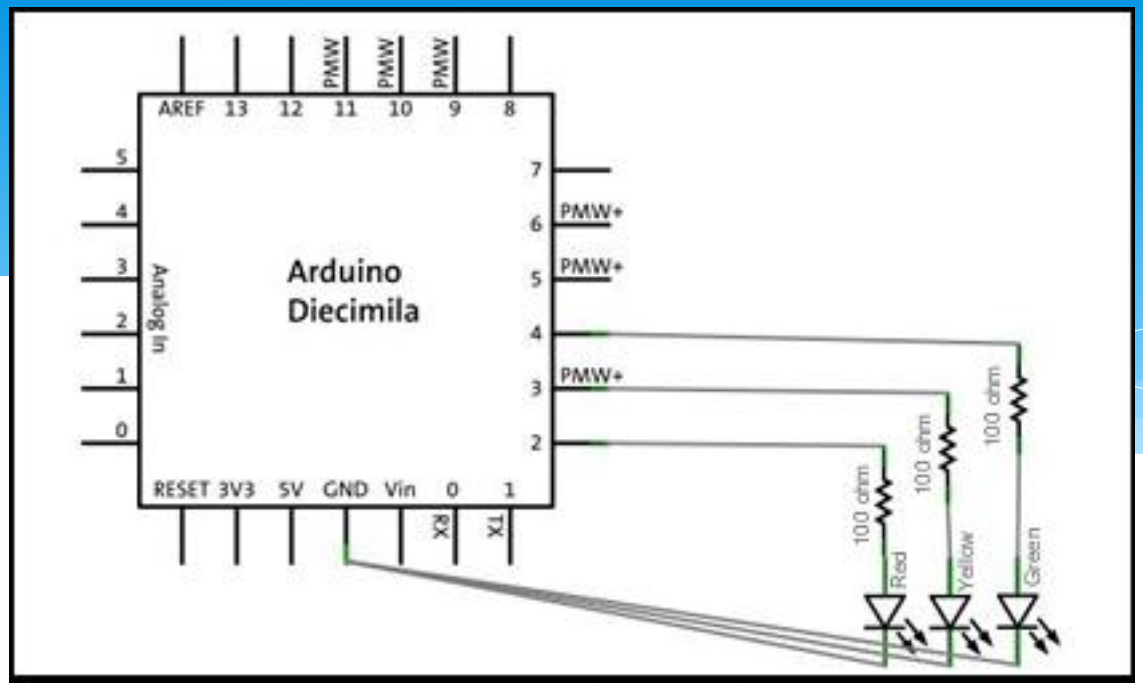
Con el siguiente código y con solo realizar modificaciones en los intervalos de tiempo que el pin seleccionado tenga valor HIGH o LOW, a través de la función digitalWrite (), generamos la señal PWM.

```
// señal PWM //

int digPin = 10; // pin digital 10

void setup() {

    pinMode(digPin, OUTPUT);    // pin en modo salida

}

void loop() {

    digitalWrite(digPin, HIGH); // asigna el valor HIGH al pin
    delay(500);                 // espera medio segundo
    digitalWrite(digPin, LOW);  // asigna el valor LOW al pin
    delay(500);                 // espera medio segundo

}
```

## Analog Out - PWM

El programa pone el pin a HIGH una vez por segundo, la frecuencia que se genera en dicho pin es de 1 pulso por segundo o 1 Hertz de pulso de frecuencia (periodo de 1 segundo) . Cambiando la temporización del programa, podremos cambiar la frecuencia de la señal. Por ejemplo, si cambiamos las dos lineas con delay(500) a delay(250), multiplicaremos la frecuencia por dos, de forma que estamos enviando el doble de la cantidad de pulsos por segundo que antes.,o sea que el LED ya no blinkea, sino que esta brillando al 50% de su brillo normal.

Ahora cambia los números del LED en 1/4 del tiempo en que esta off. Has correr el sketch y veras que el brillo es de 25%. Esta técnica se llama **pulse width modulation (PWM)**, la idea es que el LED blinkea tan rápido que no te das cuenta, pero cambias su rango de brillo entre el tiempo on y el tiempo off. También se aplica a motores y es controlada con la instrucción *analogWrite()*




P1. Manejo de una salida digital. Intermitente (*BLINK).

P2. Secuencia Básica de 3 LEDs. (SEMAFORO)

http://www.youtube.com/watch?v=I3U9I7s5lMM

O

(*semaforo 2)

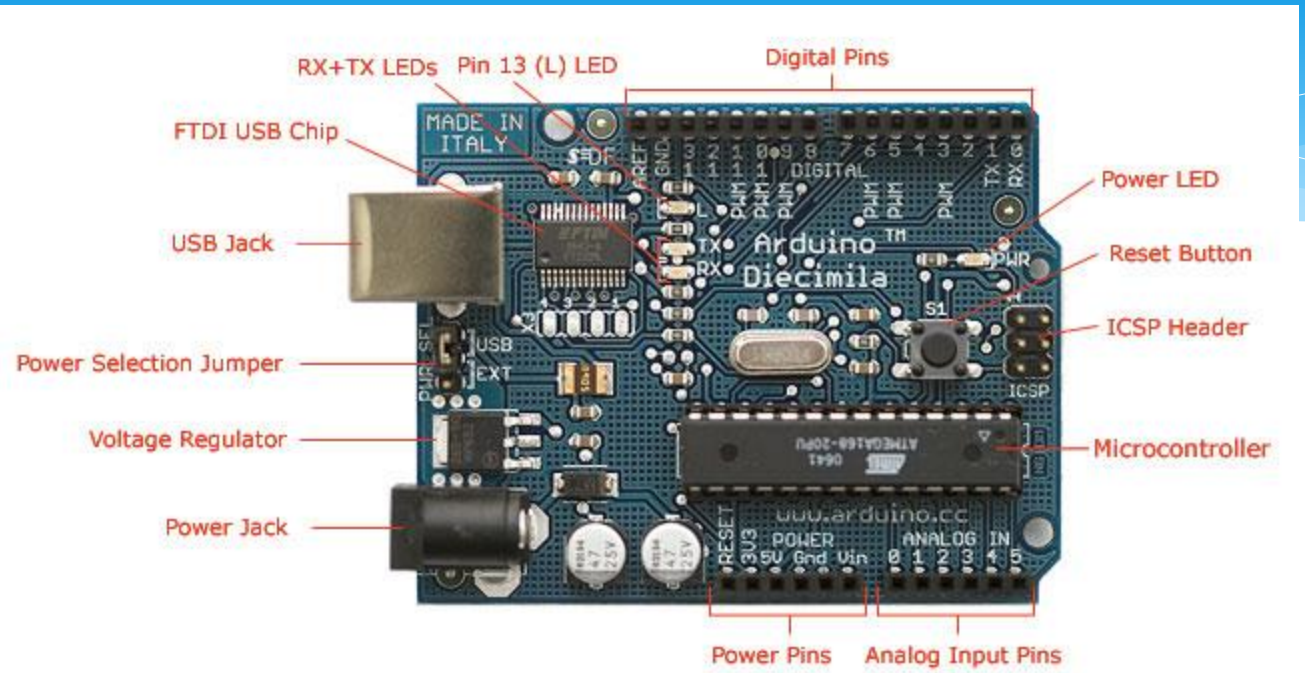
http://www.youtube.com/watch?v=wzqun5cktXk&feature=player_embedded

# Salidas/Entradas Digitales (DI - DO)

**Pines digitales** son salidas o entradas que reciben niveles altos (5V) o bajos (0V) de tensión y que son interpretados como un 1 o un 0 respectivamente.

Para controlar estas salida /entradas, se utilizan los siguientes comandos:

* pinMode(pin, mode) Configura el pin como entrada o salida pin corresponde al número del pin y mode puede ser INPUT o OUTPUT
* digitalWrite(pin, value) Escribe un 0 o un 1 (0 o 5V) en el pin especificado
* int digitalRead(pin) Lee el valor de un pin digital.




*Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.*

**Nota:** Los pines digitales 0 y 1 pueden ser utilizados como salidas o entradas siempre que se estén utilizando para la comunicación serial

# Salidas/Entradas Digitales (DI - DO)

**Pins digitales** (pines análogos no necesitan ser declarados como modo INPUT o OUTPUT) son salidas o entradas digitales, esto es niveles altos (5V) o bajos (0V) de tensión que pasan por cada uno de los pines excepto los **pines 0 (TX) y 1 (RX)** que se emplean para la comunicación en serie o comunicación de Arduino con otros dispositivos.

Para controlar estas salida /entradas, se utilizan ciertos comandos:

pinMode

digitalWrite

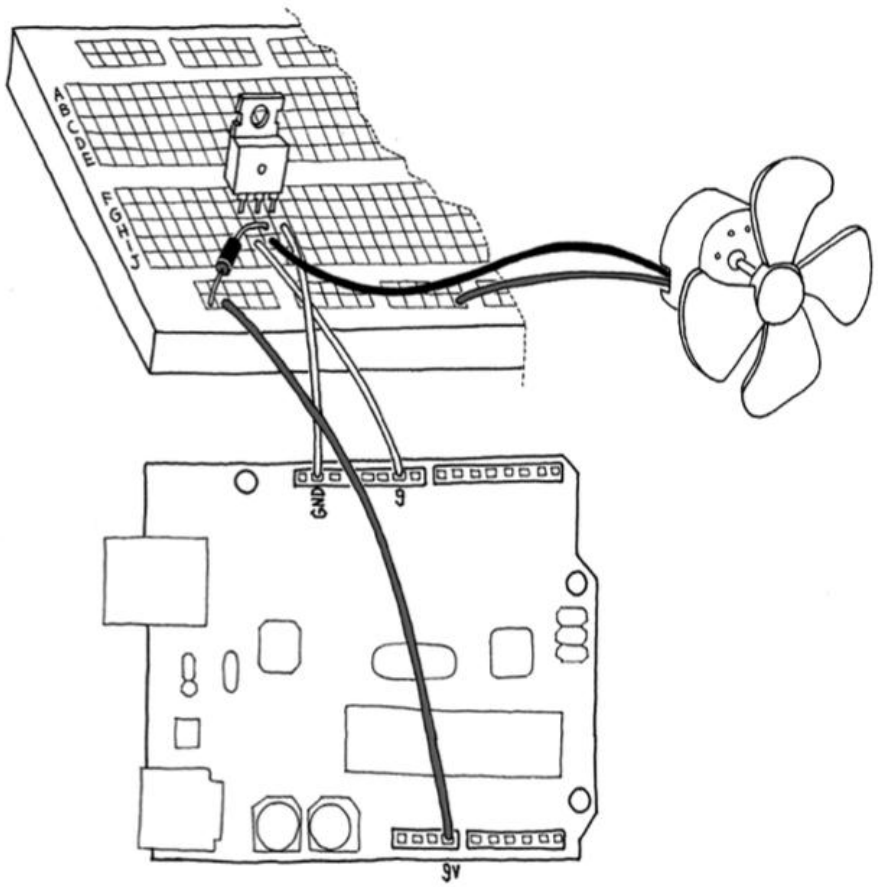
delay

digitalWrite(LED, LOW)

digitalWrite(LED, HIGH)

# Ejemplo en código

```
int ledPin = 13; // LED conectado al pin 13
int inPin = 7;   // pulsador conectado al pin 7
int val = 0;     // Variable para almacenar el valor leído

void setup()
{
 pinMode(ledPin, OUTPUT);    // configura el pin 13 como salida
 pinMode(inPin, INPUT);      // configure el pin 7 como entrada
}

void loop()
{
 val = digitalRead(inPin);   // Lee el valor del pin 7
 digitalWrite(ledPin,val);   // Enciende el LED si el pulsador está
   presionado
}
```




# Arduino conectado a un MOSFET (IRF250)

```
Int MotorPin = 9;

void setup()
{
 // configura el pin 9
como salida
 pinMode(MotorPin,
OUTPUT);
}
void loop()
{
 // Enciende el Motor

digitalWrite(ledPin,val);
}
```

# Manejo de una entrada digital, lectura de un pulsador

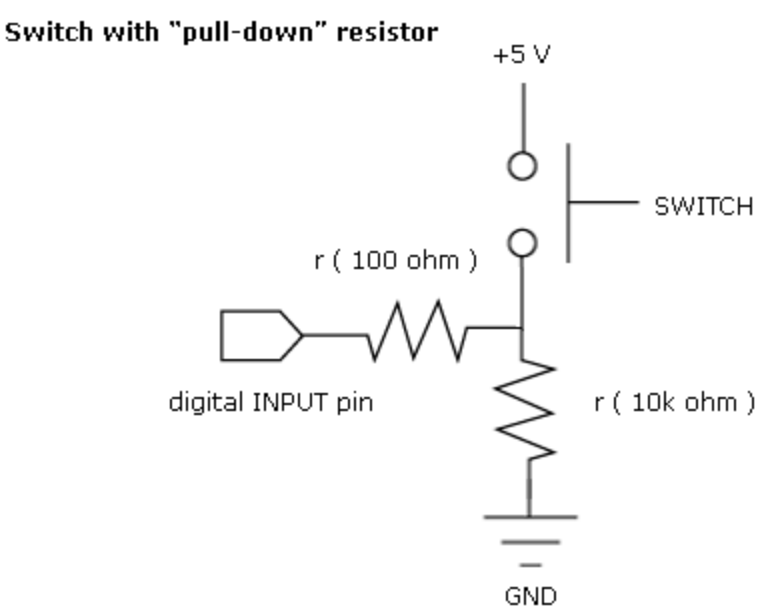
Descripción del ejercicio: Conectaremos un pulsador a la entrada digital 2 para luego leer si este se encuentra presionado utilizando el comando digitalRead (inPin)

El pulsador es un componente que conecta dos puntos de un circuito cuando es presionado.

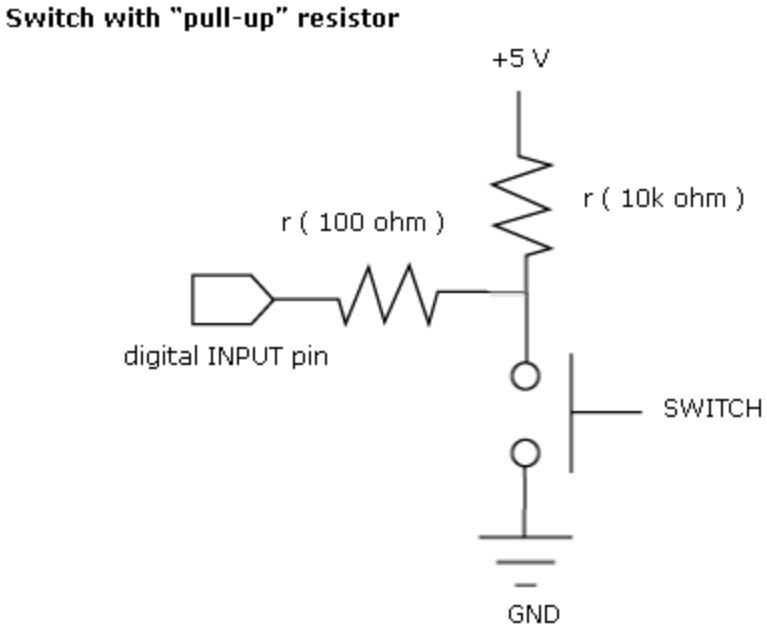
Para generar una señal de tensión con el pulsador, se necesita un divisor de tensión.

# Manejo de una entrada digital, lectura de un pulsador

Ejemplo de tipos de conexionado:




En alto cuando se presiona el botón




En bajo cuando se presiona el botón

# Manejo de una entrada digital, lectura de un pulsador

**Elementos necesarios:**

-Una resistencia de 10K Ohms.

-Una resistencia de 100 Ohms.

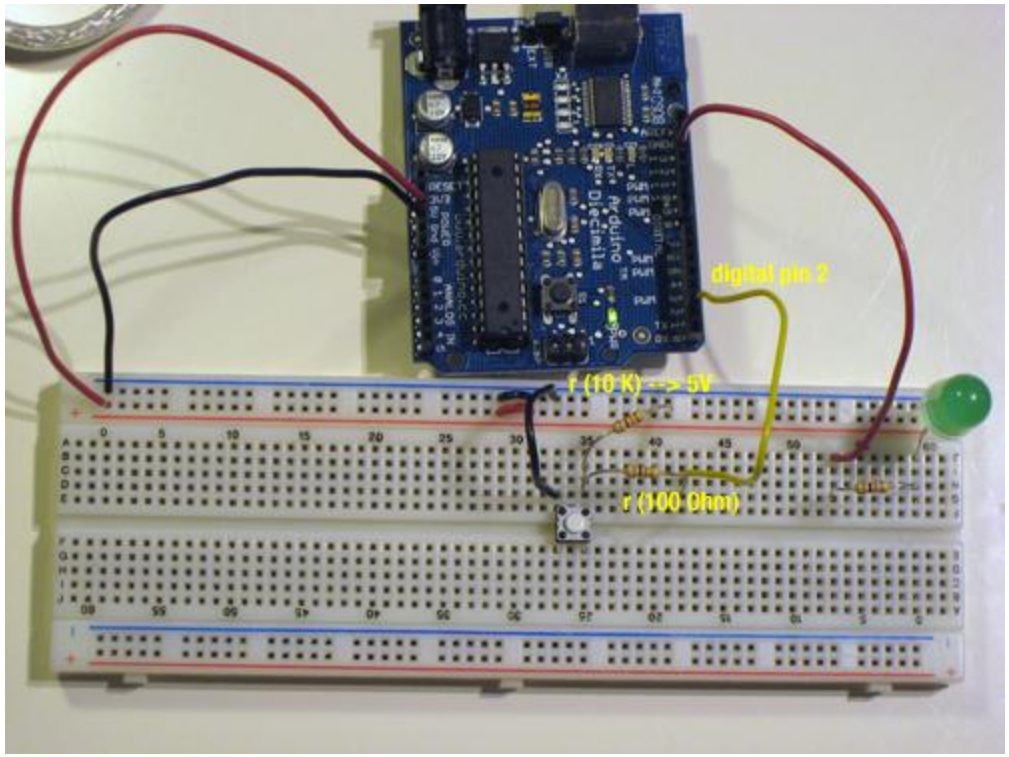
- Un pulsador.

- Un diodo LED (Utilizaremos el que está en la placa)

-Cables para realizar las conexiones.

Utilizaremos el esquema de conexión pull-down, junto con un pulsador, para conectar a un pin de entrada digital, y de esta forma, poder saber cuando el pulsador es presionado. Si el pulsador está presionado, el valor del pin 2 será de 0 voltios (LOW) en caso contrario será de + 5 voltios (HIGH).

# Manejo de una entrada digital, lectura de un pulsador




Recordemos que a diferencia de esta imagen utilizaremos el led incorporado en la placa.

# Código para Lectura del pulsador

```
int ledPin = 13; // Escoge el pin para el led
int inPin = 2; // Escoge el pin para el pulsador
int val = 0; // variable para leer el estado del pin

void setup() {
pinMode(ledPin, OUTPUT); // declara el LED como salida
pinMode(inPin, INPUT); // declara el pulsador como entrada
}

void loop() {
val = digitalRead(inPin); // Lee el valor del pulsador
if (val == HIGH) // verifica si la entrada es HIGH (Pulsador sin presionar)
{
 digitalWrite(ledPin, LOW); // apaga el LED
}
else
{  digitalWrite(ledPin, HIGH); // Enciende el LED
}
}
```

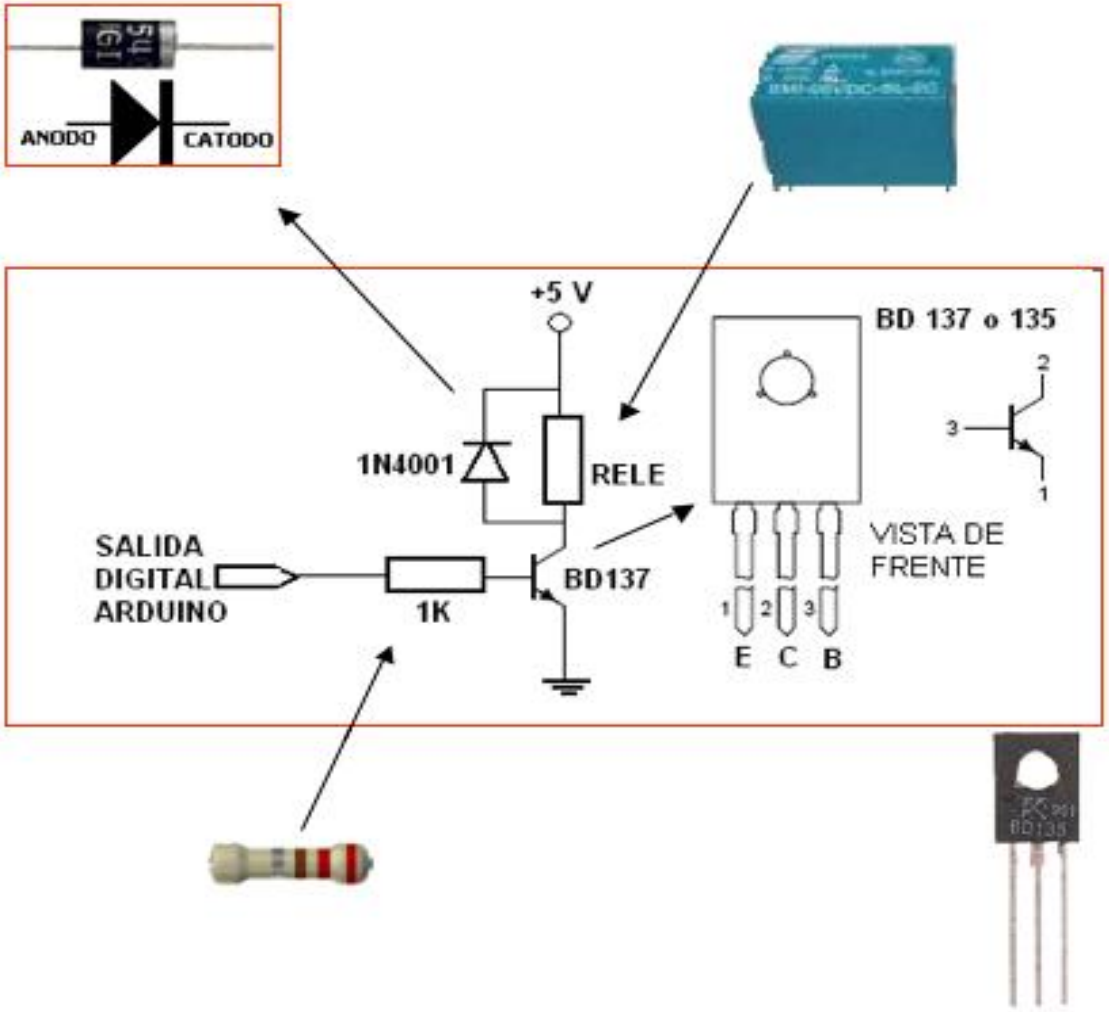
# Control de un relé

Este sencillo ejemplo enseña como encender una ampolleta de 220V de corriente alterna (AC) mediante un circuito de 5V de corriente continua (DC) controlado por Arduino. Se puede utilizar con cualquier otro circuito de 220V con un máximo de 10A (con el relé del ejemplo).

Rele

**Componentes:** el Relé es un dispositivo, que funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de un electroimán, se acciona un juego de uno varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.

# Práctica: Control de un relé




o transistor NPN

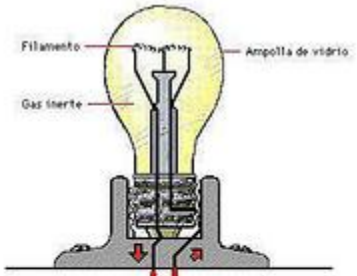
# codigo Control de un relé

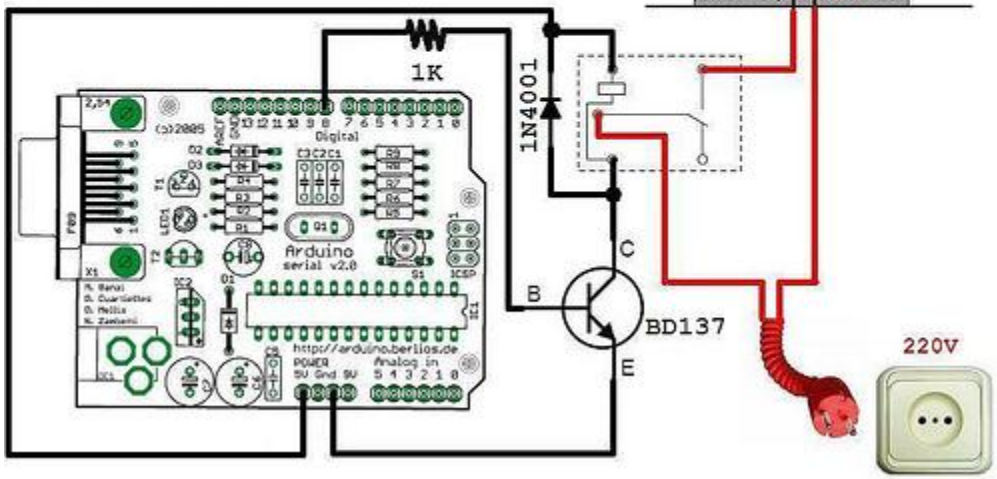
```
/* Enciende y apaga una ampolleta de 220V, cada 2 segundos,
mediante un relé conectado al PIN 8 de Arduino  */

int relayPin = 8;                    // PIN al que va conectado el

void setup()
{
 pinMode(relayPin, OUTPUT);          //Configura el Pin8 como sali
}

void loop()
{
 digitalWrite(relayPin, HIGH);   // ENCENDIDO
 delay(2000);
 digitalWrite(relayPin, LOW);    // APAGADO
 delay(2000);
 }
```

# Control de un relé


Relé

5V DC
220V AC

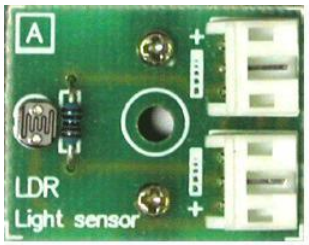


# Elementos analógos de entrada

Los sensores análogos a diferencia de los digitales entregan una salida continua, ya sea de voltaje o corriente. Es decir, no solo 0 o 5V , si no que también valores intermedios.

Cada sensor posee su propia escala por lo que probablemente tendrás que ocupar matemáticas para poder calcular el valor correcto de la medición.


# Funciones para el manejo de entradas análogas

int analogRead(pin) Lee el valor de un pin analógico. Entrega un valor entre 0 y 1023 que representa 0 o 5V es decir 4.9mV por unidad.

```
int ledPin = 9;       // LED conectado al pin 9
int analogPin = 3;    // potenciómetro conectado al pin 3
int val = 0;          // variable para almacenar el valor
leído
void setup()
{
 pinMode(ledPin, OUTPUT);   // Configura el pin como
salida
}

void loop()
{
  val = analogRead(analogPin);     // Lee el pin de
entrada (valor entre 0 a 1023)
}
```

# Arduino y sus entradas analógicas

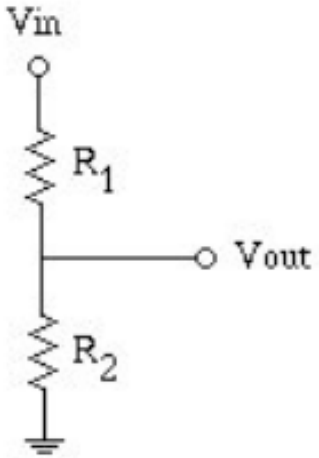
Arduino tiene un conversor análogo digital interno de 6 canales accesibles desde los pines marcados en la placa como ANALOG IN 0-5. Estos pueden leer un voltaje que se encuentre entre los 0 y los 5 voltios. La medición tiene una resolución de 10 bits (1024 valores) lo que nos daría una resolución de (5/1024) = 0.0048 =4.8 mv.

En este caso (ADC interno) para poder hacer la medición buscaremos la forma de implementar un divisor de voltaje con el sensor que estemos trabajando, el valor de la resistencia de división deberá estar en el mismo orden de magnitud que pueda dar la máxima lectura del sensor.

# Divisores de tensión para acoplar sensores

En electrónica, un divisor de voltaje (o tensión o potencial) es un circuito lineal simple que produce un voltaje como output (Vout) que es una fracción de su voltaje como input (Vin). División de voltaje se refiere a la partición del voltaje.

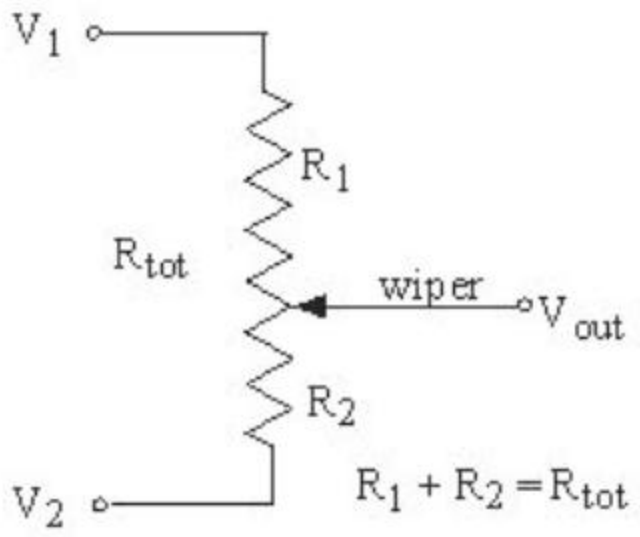
**Voltage Divider**



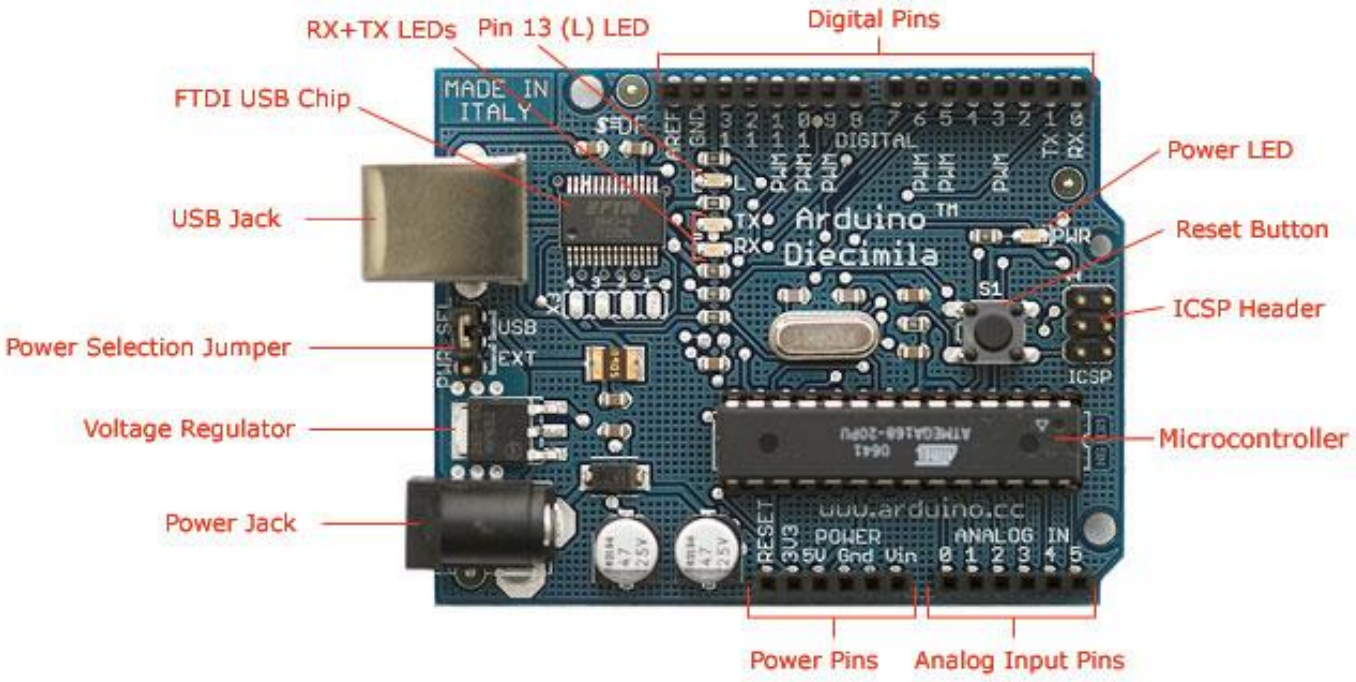

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$

# Divisores de tensión para acoplar sensores

Un ejemplo simple de divisor de voltaje, consiste en dos resistencias en serie o un potenciómetro. Es comúnmente utilizado para crear un voltaje de referencia y también puede ser usado como un atenuador de a baja frecuencia. Para poder leer estos cambios en la resistencia, los pondremos en un circuito (divisor de voltaje) y pasaremos una corriente a través de ellos de tal forma que podemos medir el cambio de voltaje resultante sobre el sensor. ya hecha la medición tendremos que convertir ese voltaje a un valor digital, para este trabajo se utiliza el componente electrónico llamado conversor analógico digital (ADC).



# Arduino y sus entradas analógicas




*Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.*

**Nota:** Las entradas analógicas al igual que las digitales soportan valores de 0 a 5V.

# Potenciómetro para simular una entrada analógica

* El potenciómetro es un dispositivo electromecánico que consta de una resistencia de valor fijo sobre la que se desplaza un contacto deslizante, el cursor, que la divide eléctricamente.

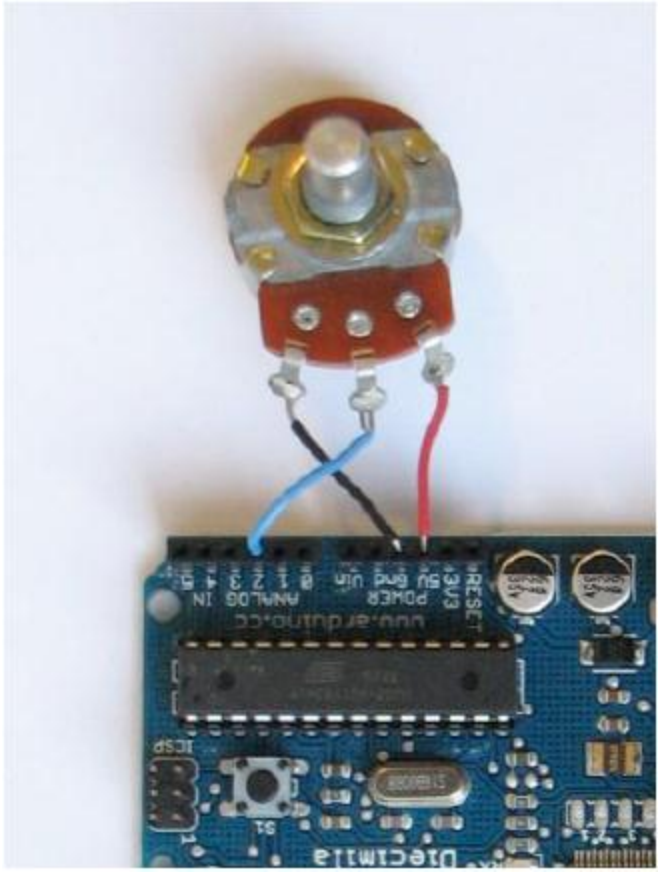
  Vout=((R1/(R1+R2))*Vin (Aplicando la ley de Ohm)

* Un potenciómetro es especificado por su resistencia total, R, entre los terminales externos 1 y 3; El movimiento del cursor origina un cambio en la resistencia medida entre el terminal central, 2, y uno cualquiera de los extremos.

  Este cambio de resistencia puede utilizarse para medir desplazamientos lineales o angulares de una pieza acoplada al cursor.

  Se conectan en paralelo al circuito y se comporta como un divisor de tensión.

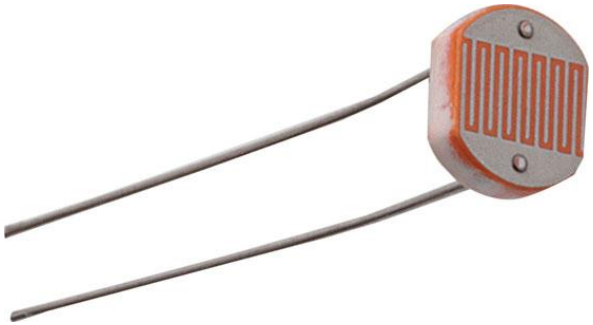
# Potenciómetro como una entrada analógica



```
int potPin = 2; // Pin A/I pot
int ledPin = 13; // Pin D/O LED
int val = 0; // variable guarda valor del
sensor

void setup()
{
 pinMode(ledPin, OUTPUT); // ledPin output
}

void loop()
{
 val = analogRead(potPin); // lee valor del
sensor
 digitalWrite(ledPin, HIGH); // enciende LED
 delay(val); //detiene el programa por un
tiempo
 // determinado por la variable
 digitalWrite(ledPin, LOW); // apaga el LED
 delay(val); //detiene el programa por un
tiempo
 // determinado por la variable
}
```

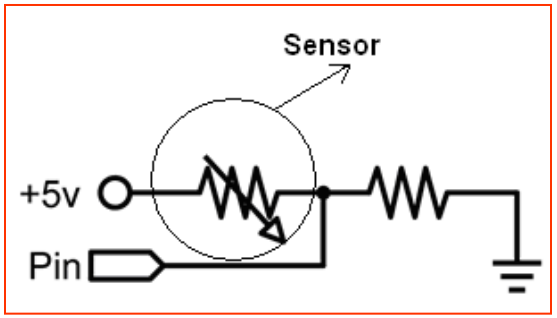
# Midiendo Luz con Arduino

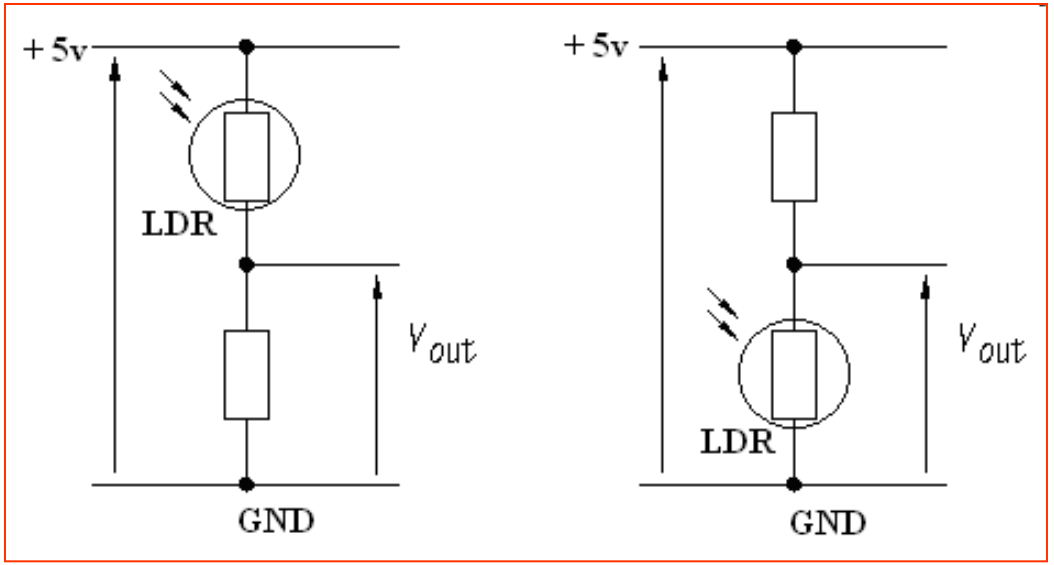
* Una fotocélula (o LDR -Light-Dependent Resistor) es una resistencia que varia su intensidad según la luz que recibe. Cuanta más luz menos resistencia ejerce a la corriente



* Una LDR considera un sensor de luminosidad, se pueden usar para apagar las luces es de día o encenderlas cuando es de noche. Estas variables son de tipo analógico, como medir el sonido, la inclinación, la presión o el desplazamiento.

# Práctica: Midiendo Luz con Arduino




Listado de componentes:

* 1 LDR
* 1 Resistencia de 1kΩ
* Un par de cables

**Nota:** El LDR del Kit ya viene con la resistencia por lo que no necesitamos agregar nada adicional

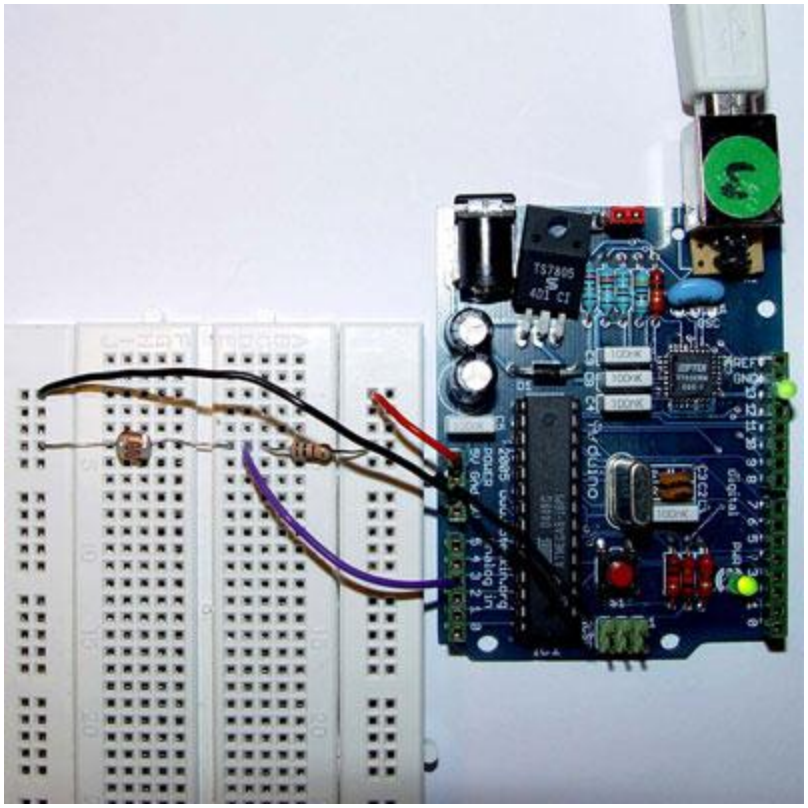
# Midiendo Luz con Arduino

* En set de codigos Al_lee LDR

```
int LightPin = 3; // selecciona el pin de entrada para el sensor de luz

int ledPin = 13; // selecciona el pin para el LED

int val = 0; // variable para almacenar el valor capturado desde el sensor

void setup() {

  pinMode(ledPin, OUTPUT);  // declara el ledPin en modo salida
  serial.begin(9600);
}

void loop() {

  val = analogRead(LightPin);    // read the value from the sensor
  digitalWrite(ledPin, HIGH);  // enciende el LED
  delay(val);                  // detiene el programa por un tiempo
  digitalWrite(ledPin, LOW);   // apaga el LED
  delay(val);                  // detiene el programa por un tiempo
  serial.println ("Sensor de luz: \d", val);
}
```

# Midiendo Luz con Arduino

Recuerda encender el monitor serial para leer los valores del LDR




# *Muchas Gracias*