Taller de electrónica creativa: Cocinando con Arduino

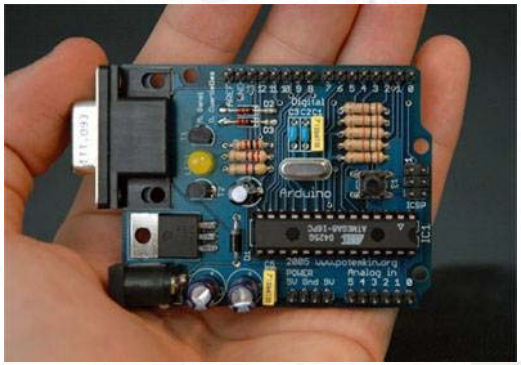
12, 13, 14 - 19, 20 y 21

de Junio de 2009

Yago Torroja
Igor González
Angela Ramos
y colaboradores

Introducción a Arduino

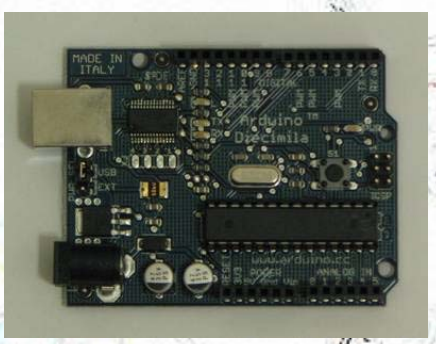
- Arduino es una plataforma open-source de desarrollo de protipos, basada en hardware y software fácil de usar.
- Está pensada para artistas, diseñadores, aficionados a la electrónica, y cualquiera interesado en crear objetos y entornos interactivos.



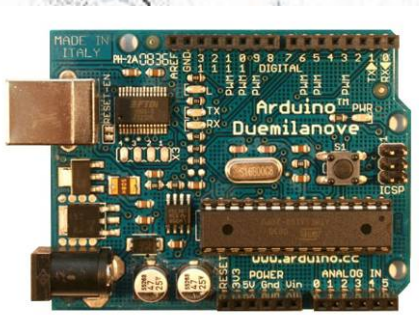
Arduino serie

Introducción a Arduino

Arduino NG



Arduino Diecimila

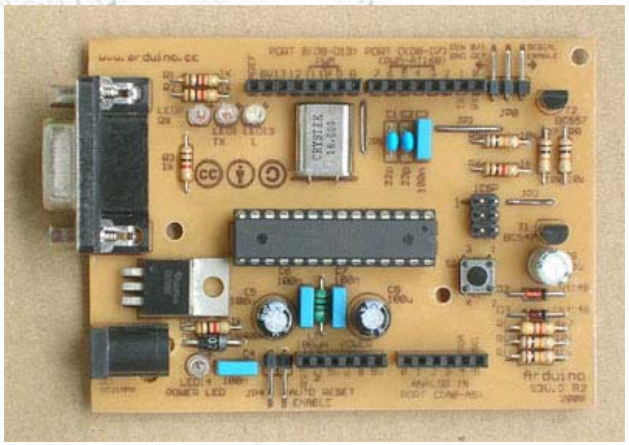


Arduino Duemilanove

- Arduino es una placa con un microcontrolador que permite conectar sensores y actuadores mediante sus entradas y salidas, analógicas y digitales.
- El microcontolador se programa utilizando un lenguaje propio de Arduino (basado en Wiring) y un entorno de desarrollo integrado (IDE) propio (basado en Processing).

Introducción a Arduino

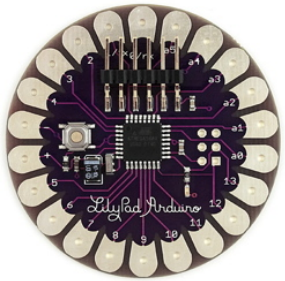
- Los proyectos desarrollados con Arduino pueden ser autónomos (stand-alone) o pueden conectarse con cualquier software a través del puerto serie (p.e. Flash, Processing, MaxMSP ...), bien por cable o por Xbee/ZigBit/etc ...
- Las placas se pueden ensamblar a mano o comprarse montadas. El software se puede descargar gratis de la web. Los esquemáticos (ficheros CAD) están disponibles bajo licencia open-source, por lo que se pueden modificar si es necesario.



Placas Arduino compatibles

Arduclema

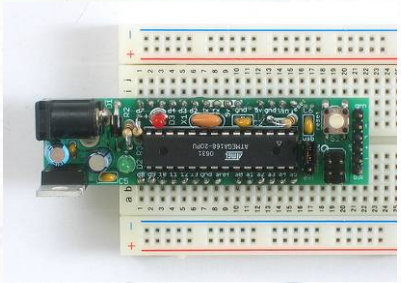
Lilypad



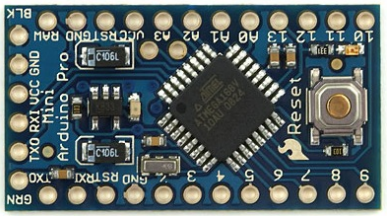
Arduino Pro

Arduino BT

Boarduino



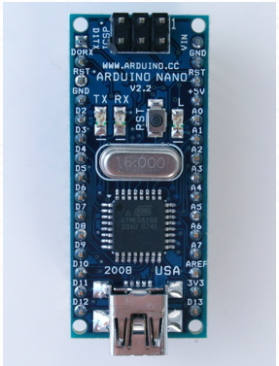
Arduino Pro Mini



Arduino MEGA

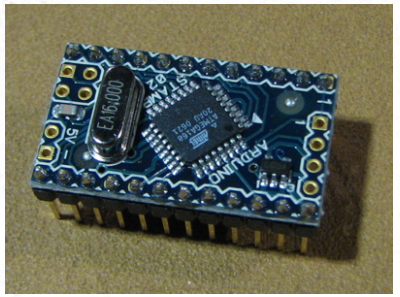
Freeduino

Arduino Nano



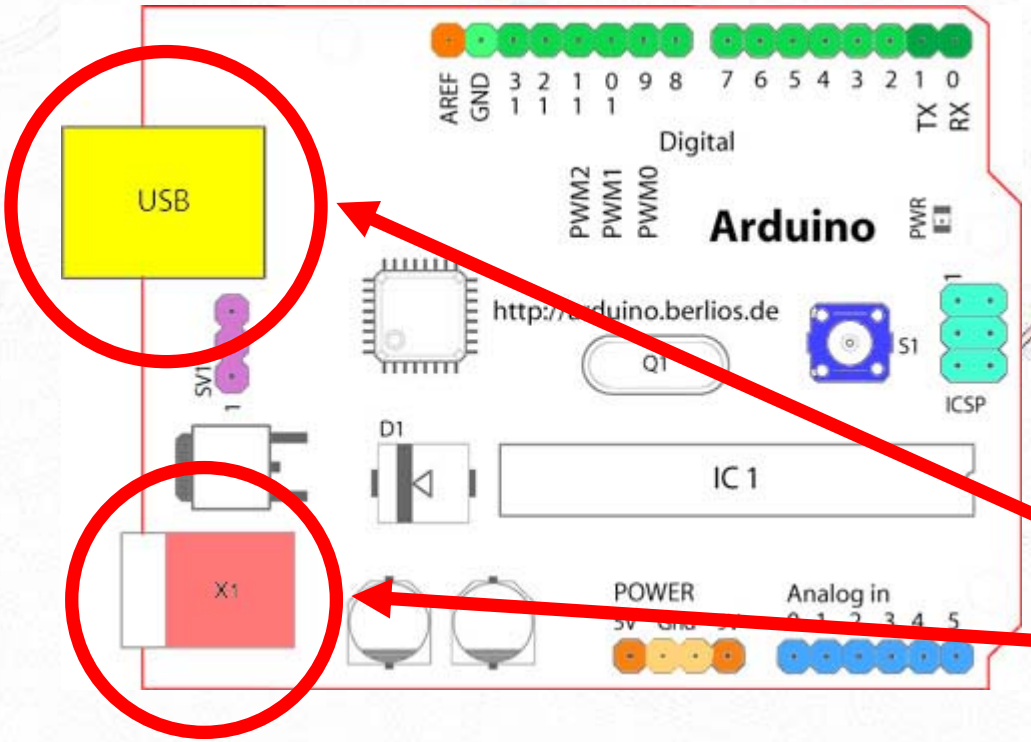
Seeduino

Arduino Mini

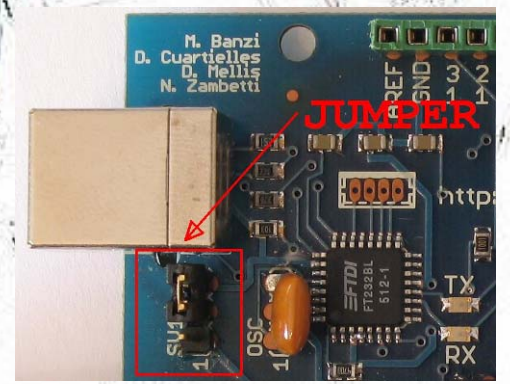


MEDIALAB PRADO

La placa Arduino - Alimentación



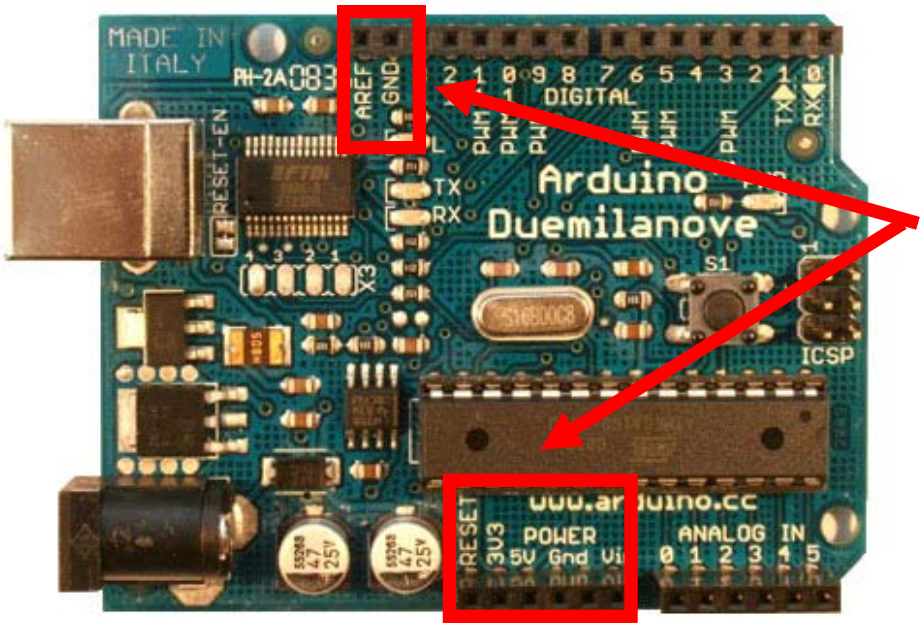
Dos alternativas:



- Puerto USB.
- Regulador de tensión: 5 a 15 V

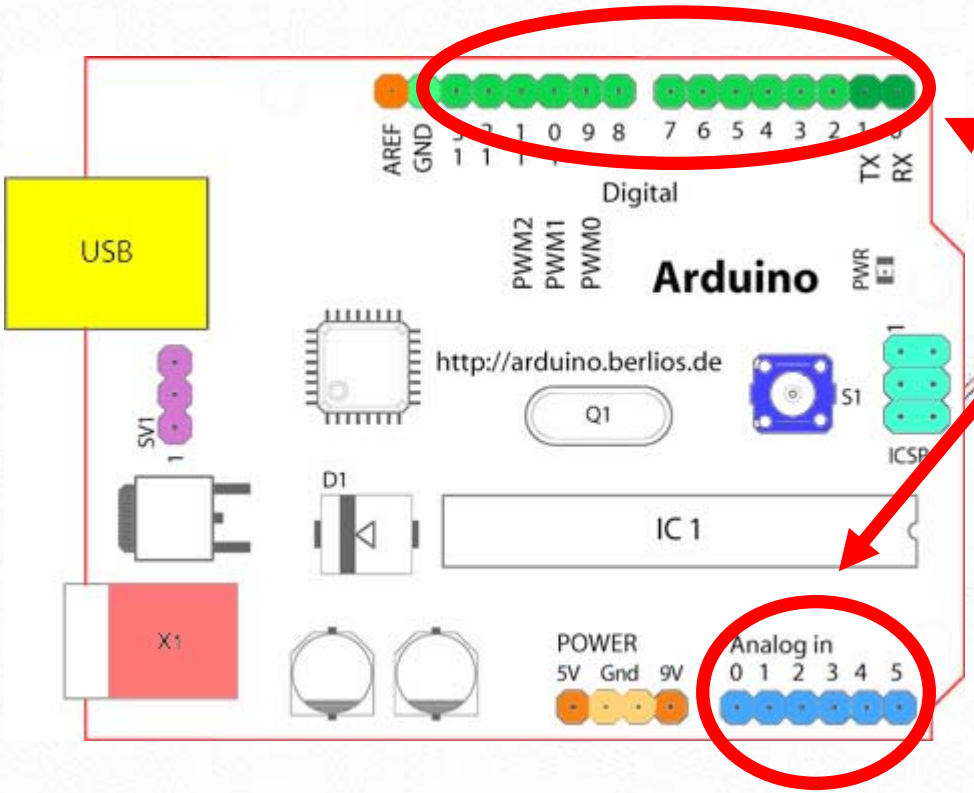
MEDIALAB PRADO

La placa Arduino - POWER



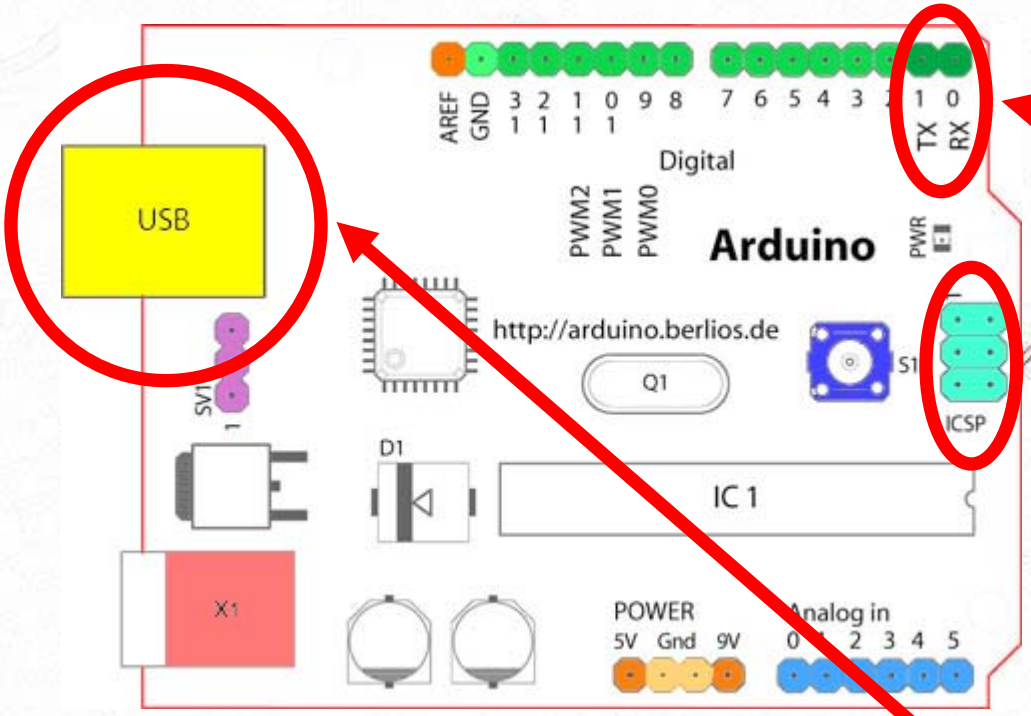
- **3V3** – 3,3 voltios
- **5V** – 5 voltios
- **GND** – 0 voltios
- **Vin** – Voltaje de alimentación externa
- **AREF** – Voltaje de referencia para entradas analógicas

La placa Arduino – E/S



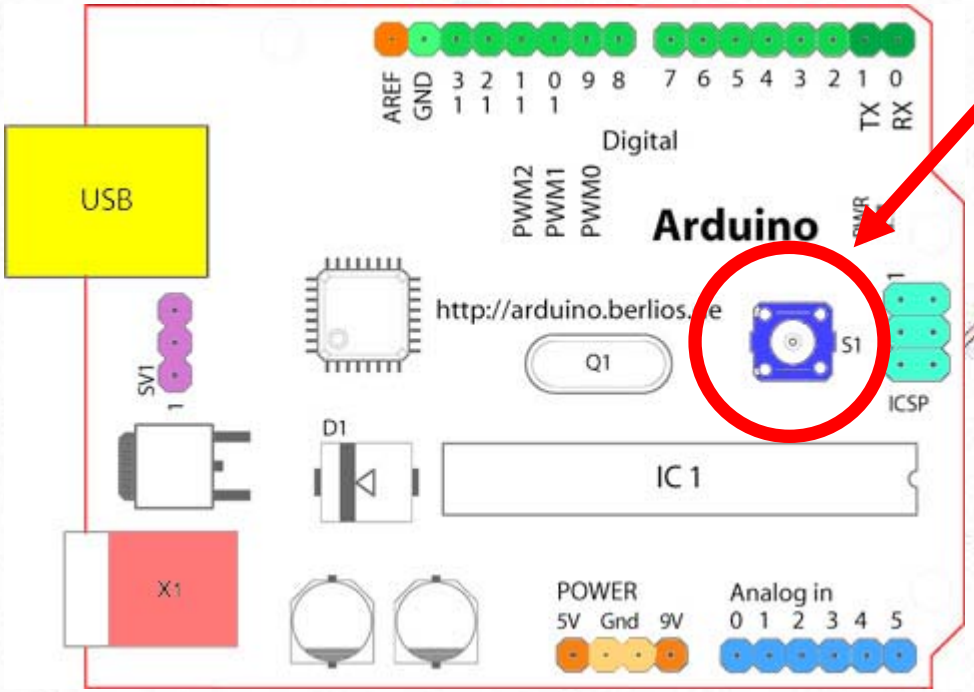
- 14 (hasta 20) pines de E/S digitales
- 6 entradas analógicas
- 6 salidas analógicas (PWM)

La placa Arduino - Comunicaciones



- Puerto serie: RX/TX
- ICSP
- Puerto USB (FTDI)

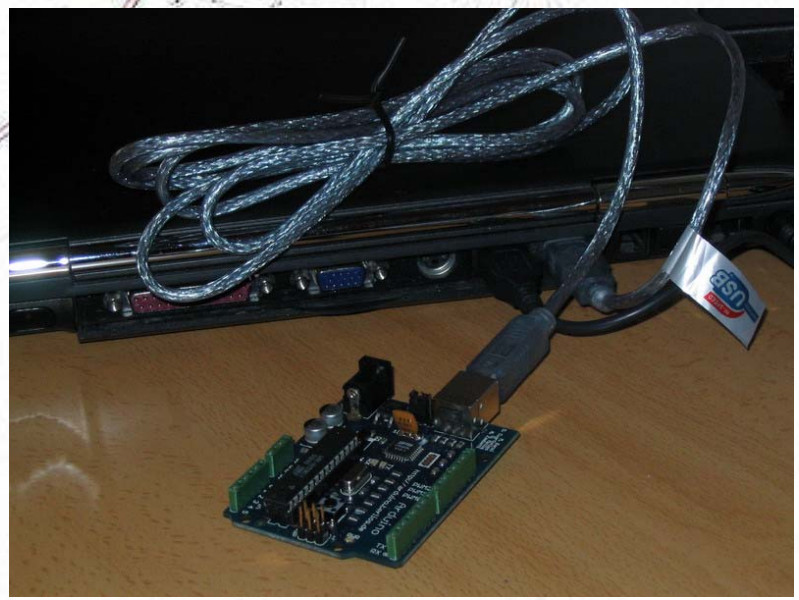
La placa Arduino - Varios



- Botón de reset
- Reloj a 16/20 Mhz
- Microcontrolador Atmega8/168 → 8/16 Kb
- Bootloader

Conexión Arduino-PC

- Instalación de los drivers FTDI.
- Cable USB tipo A

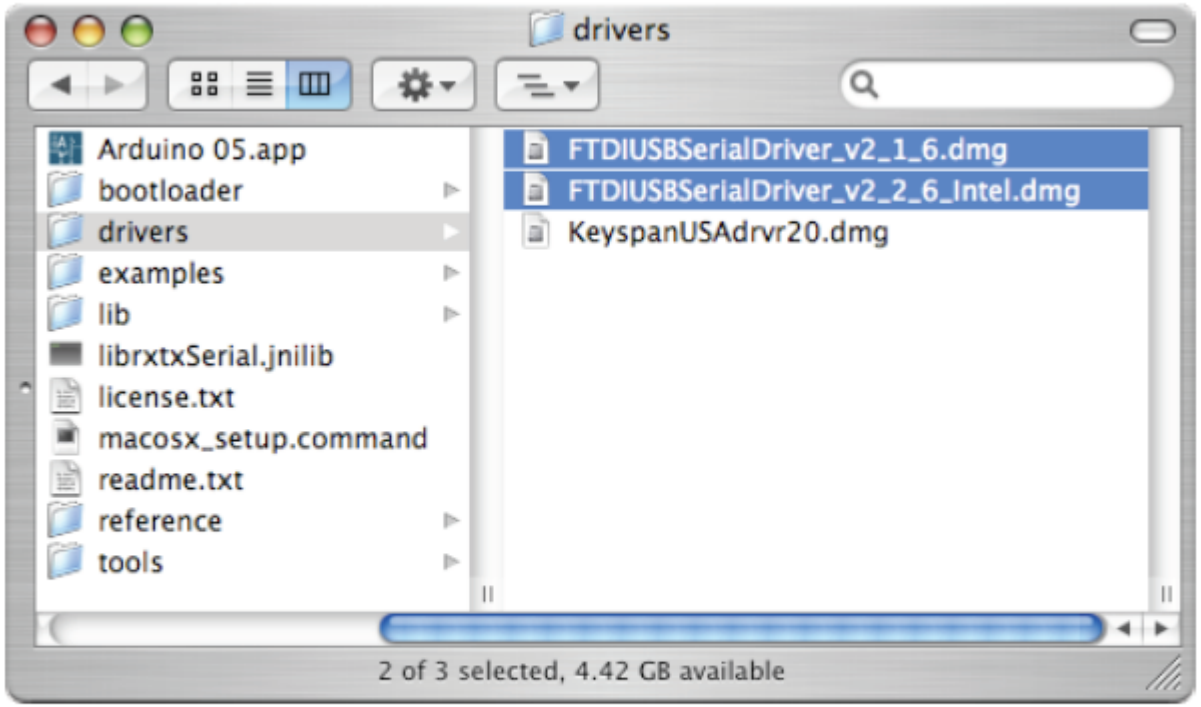


El entorno Arduino (IDE)

- Descarga de la última versión:
  - http://www.arduino.cc/en/Main/Software
- Instalación en el PC:
  - Windows: Drivers + descomprimir y ejecutar
  - Linux: Descomprimir el paquete y ejecutar.
  - MAC: Drivers + descomprimir y ejecutar
- Guía rápida:
  - http://www.arduino.cc/es/Metodolog%eda/GuiaRapida

El entorno Arduino (IDE) - Drivers MAC

Double-click on .dmg Installer



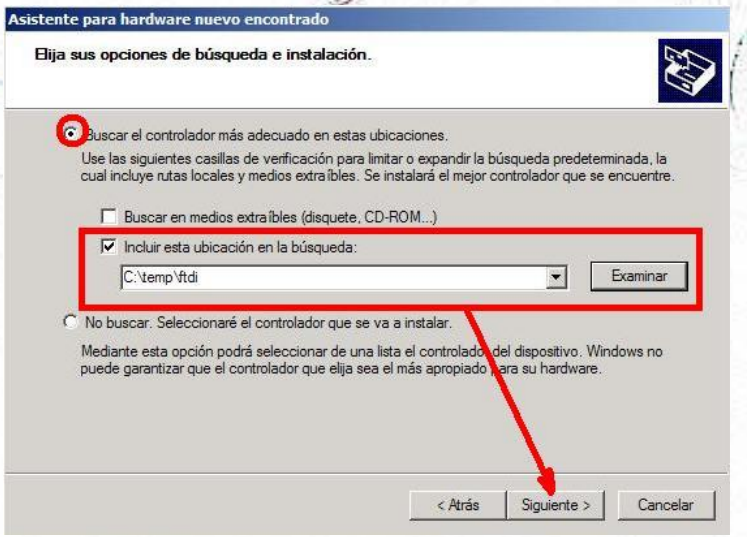
- v2_1_6 for PPC Macs
- v2_2_6 for Intel Macs

El entorno Arduino (IDE) - Drivers Windows

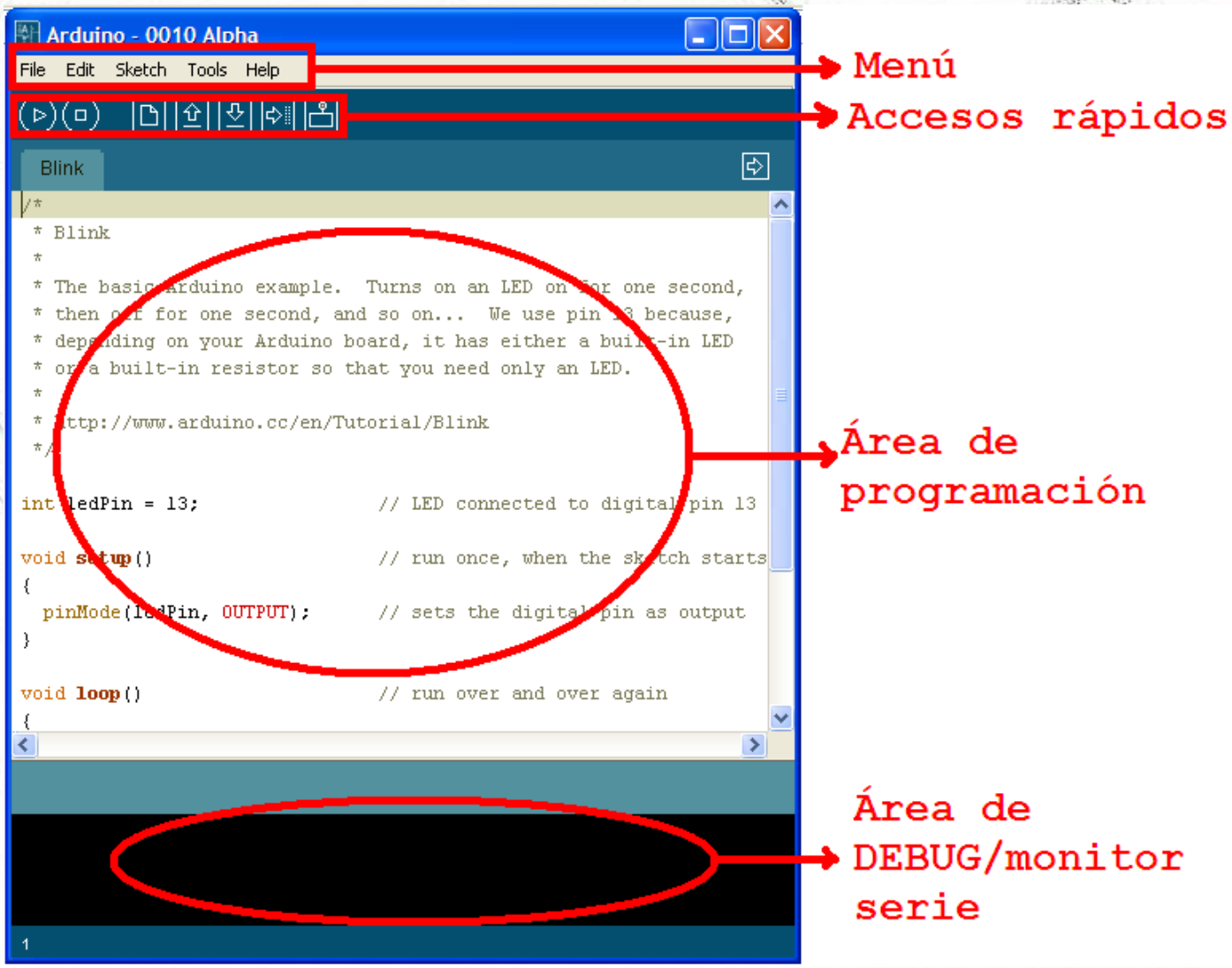
1

2

3



El entorno Arduino (IDE)



Menú

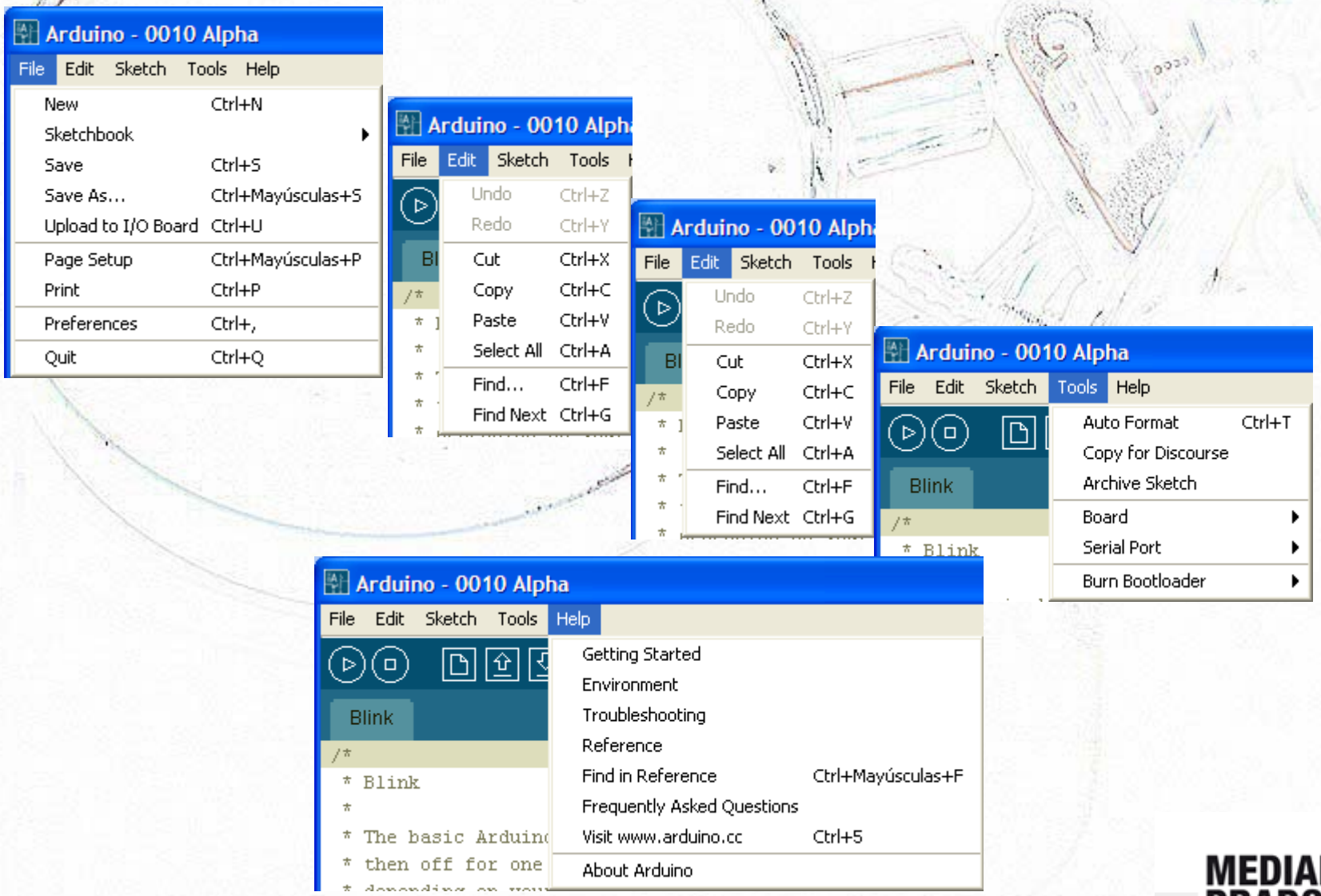
Accesos rápidos

Área de programación

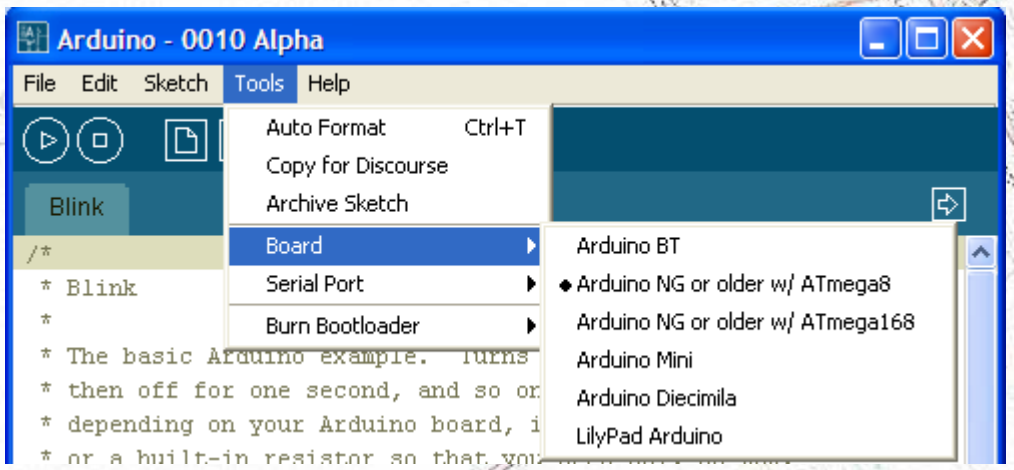
Área de DEBUG/monitor serie

MEDIALAB PRADO

El entorno Arduino (IDE) - Menu

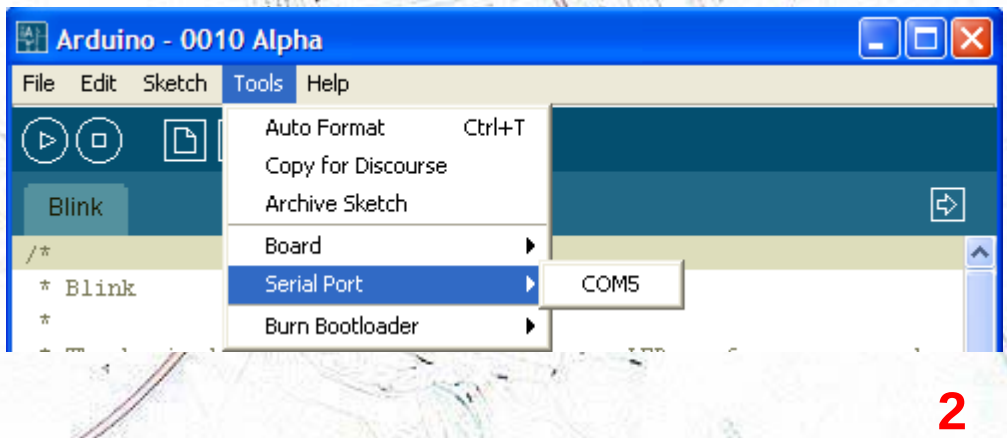


El entorno Arduino (IDE) - Placa



- Tools → Board: Seleccionar el tipo de placa

El entorno Arduino (IDE) - Puerto



- Mi Pc → Administrar → Administrador de de dispositivos → Puertos (COM & LPT)
- Tools → Serial Port → Seleccionar el puerto al que está conectada la placa.

El entorno Arduino (IDE) - Bloques

Bloque 0 – Comentarios (OPCIONAL)

Bloque 1 – Declaración de las variables que vamos a utilizar

Bloque 2 – Configuración inicial del programa

Bloque 3 – Bucle infinito que contiene el conjunto de instrucciones que se repiten constantemente

El entorno Arduino (IDE) - Subir un programa a la placa

Salidas Digitales

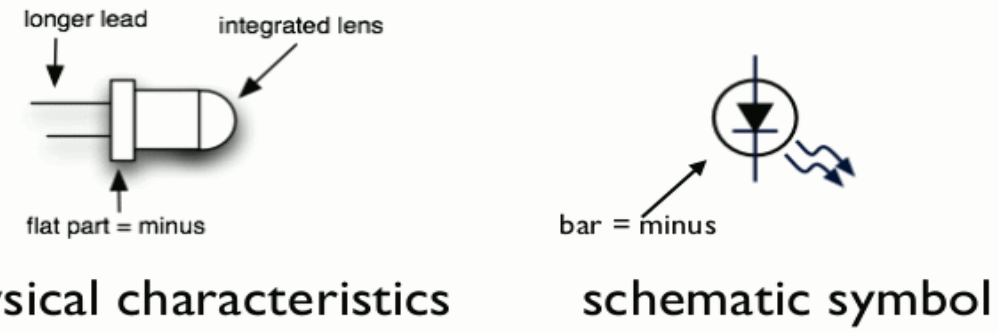
Los **pines digitales** se pueden emplear como salidas o entradas digitales, es decir, se puede escribir niveles altos (5V) o bajos (0V) de tensión a cada uno de los pines y viceversa, excepto los **pines 0 (TX) y 1 (RX)** que se emplean para la comunicación serie o comunicación de Arduino con otros dispositivos.

Comandos básicos:

- **pinMode(pin, modo)**, sirve para declarar un pin digital como entrada (INPUT) o como salida (OUTPUT). Los pines analógicos son, por defecto,de entrada.
- **delay(tiempo)**, sirve para parar los procesos de la placa durante tiempo milisegundos y delayMicroseconds(tiempo) sirve para parar los procesos de la placa durante tiempo microsegundos.
- **digitalWrite(pin, valor)**,sirve para escribir un valor al pin digital, el valor podrá ser 1 lógico (HIGH=5v) o 0 lógico (LOW=0v)
- **setup()** es la función de configuración de los pines de Arduino y sólo se ejecuta una vez, mientras que **loop()** se ejecuta una y otra vez hasta que apaguemos el sistema, o se gasten las baterías.

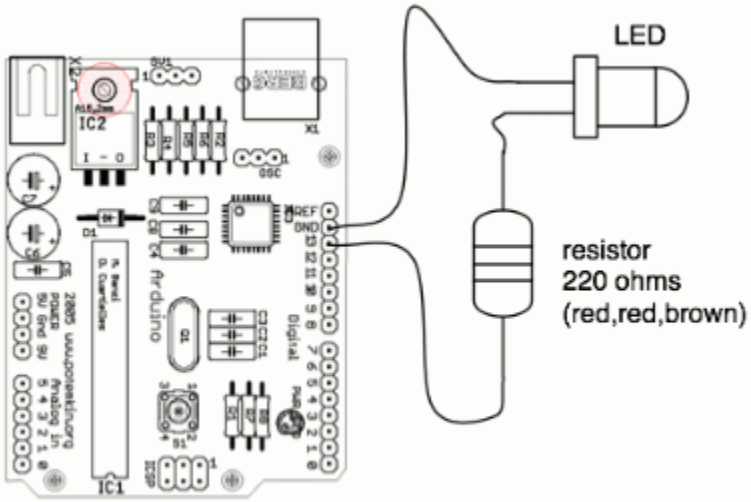
Salidas Digitales- Encender LEDs

Componentes: **Diodo LED**, polaridad, pata positiva la más larga, regla mnemotécnica del triángulo, transforma la electricidad en luz (actuador). Para que no se funda, debe ir acompañado por una resistencia.

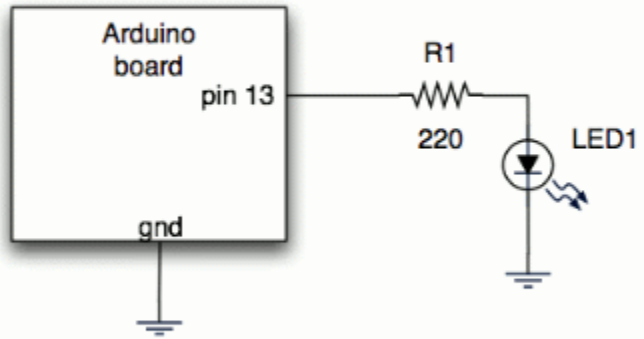


PIN13, tiene una resistencia por defecto para poder colocar un LED directamente. Es el que se utiliza en el ejemplo básico : “blink”. Para ello debemos acceder a través del menú File --> Sketchbook --> Examples --> Digital --> Blink (Parpadeo de un led conectado al pin13)

Salidas Digitales-Encender LEDs



wiring diagram

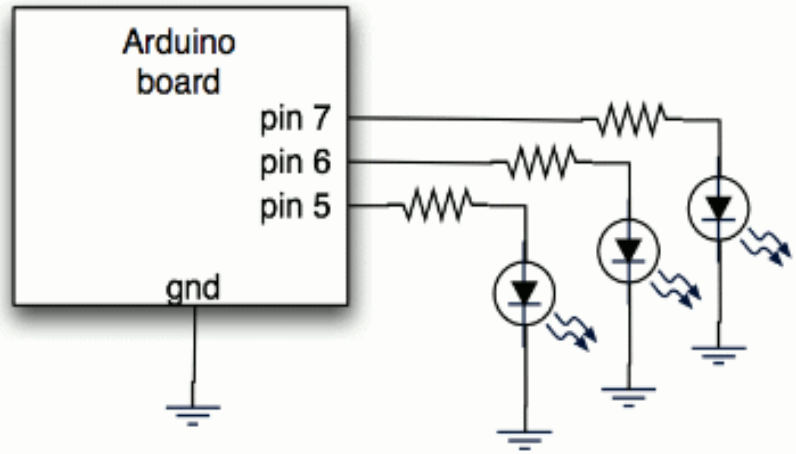


schematic

MEDIALAB PRADO

Salidas Digitales - Encender LEDs

```
int LedAPin =7;
int LedBPin =6;
int LedCPin =5;

void setup(){
  pinMode (LedAPin, OUTPUT);
  pinMode (LedBPin, OUTPUT);
  pinMode (LedCPin, OUTPUT);
}

void loop(){
  digitalWrite(LedAPin,HIGH);
  delay(1000);
  digitalWrite(LedAPin,LOW);
  digitalWrite(LedBPin,HIGH);
  delay(1000);
  digitalWrite(LedBPin,LOW);
  digitalWrite(LedCPin,HIGH);
  delay(1000);
  digitalWrite(LedCPin,LOW);
}
```



Salidas Digitales -Hacer sonar piezos

Componentes: el **piezo electrico** o "chicharra" es un componente que transforma un tren de pulsos en sonido. A bajo nivel transforma un cambio de voltaje en un movimiento físico de dos placas metálicas. También puede ser utilizado como Sensor de Vibraciones, para ello hay que situarlo en paralelo con una resistencia de 1M y conectarlo a una entrada analógica.

Salidas Digitales-Hacer sonar piezos

Básicamente, generamos un tren de pulsos (como en el ejemplo del parpadeo del LED) , pero con la frecuencia de cada una de las notas. El “tone” (o tono) está calculado en función de la inversa de la frecuencia de la nota.

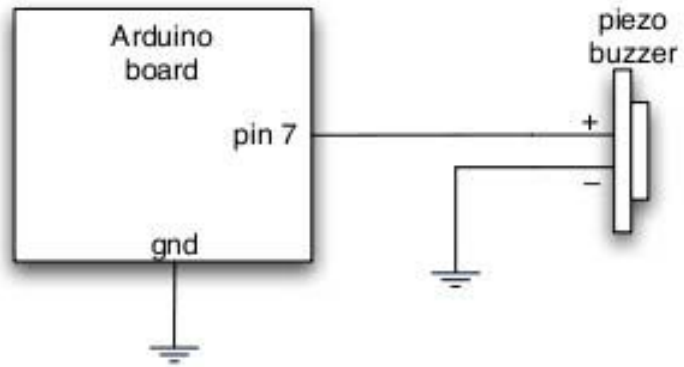
Link con valores de las frecuencias de las notas:

http://www.latecladeescape.com/w0/content/view/94/49/1/1/

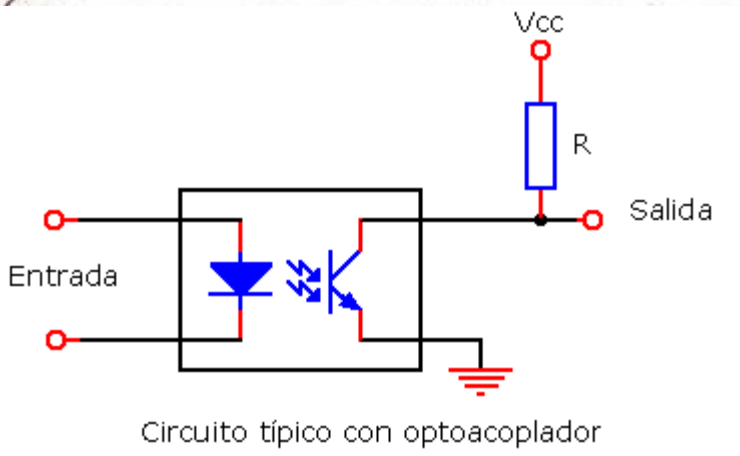
Por ejemplo para que suene la nota LA:

// nota a tono=1/440Hz=2,272ms=2272us frecuencia=440 Hz

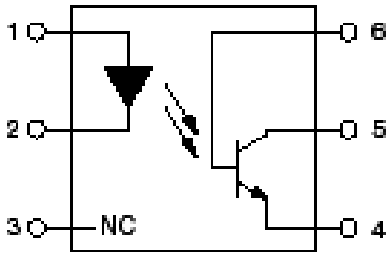
```
int speakerOut =7;

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

void loop() {
  for (int i = 0; i < 100; i++){
    digitalWrite(speakerOut,HIGH);
    delayMicroseconds(1136);
    digitalWrite(speakerOut, LOW);
    delayMicroseconds(1136);
  }
}
```



Salidas Digitales - Optoacoplador



Circuito típico con optoacoplador

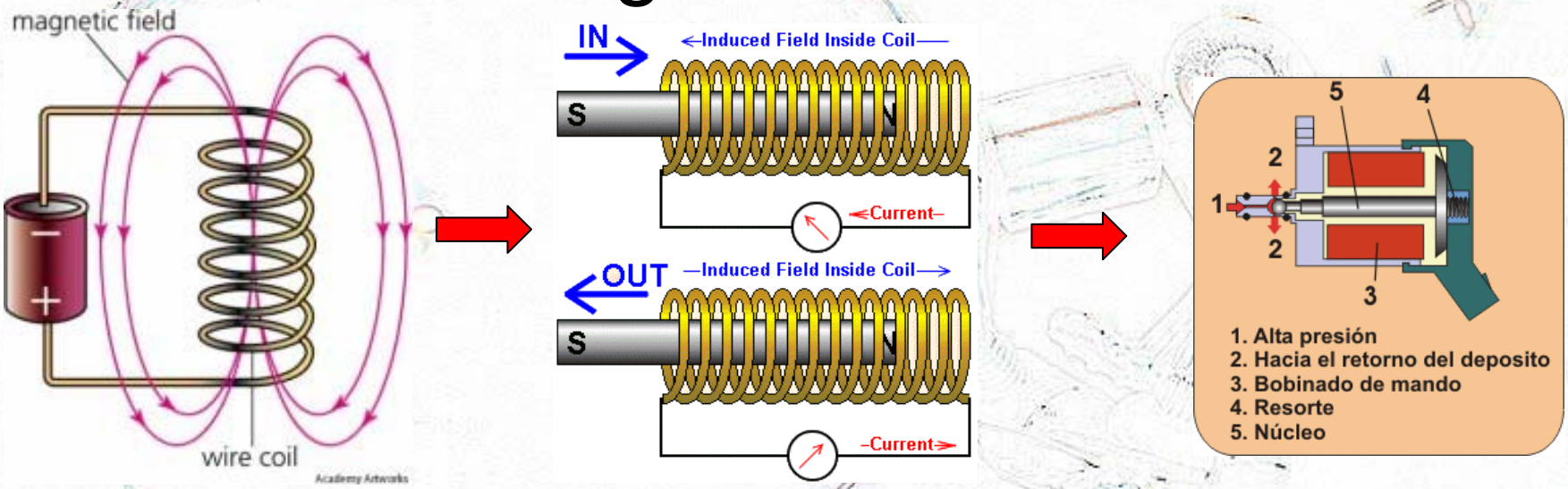


PIN 1. ANODE
2. CATHODE
3. NO CONNECTION
4. EMITTER
5. COLLECTOR
6. BASE

Circuito integrado 4N35

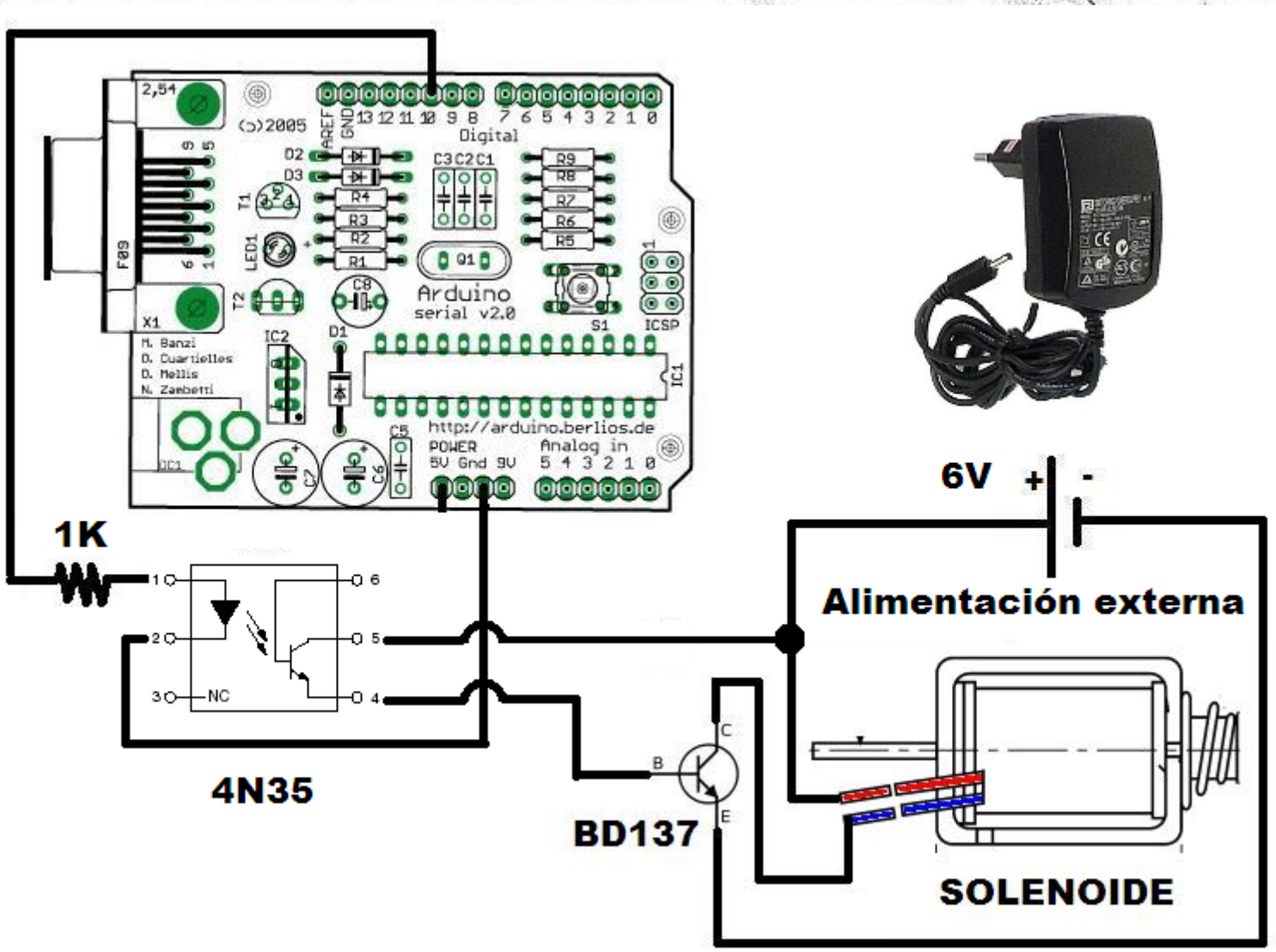
- Un optoacoplador es un dispositivo de emisión y recepción de luz que funciona como un interruptor activado mediante la luz. La mencionada luz es emitida por un diodo LED que satura un componente optoelectrónico, normalmente en forma de fototransistor. De este modo se combinan en un solo dispositivo semiconductor, un fotoemisor y un fotorreceptor cuya conexión entre ambos es óptica.
- Se suelen utilizar para separar circuitos que funcionan a distintos voltajes.

Salidas Digitales - Solenoide



- El solenoide es un alambre aislado enrollado en forma de hélice (bobina) por el que circula una corriente eléctrica. Cuando esto sucede, se genera un campo magnético dentro del solenoide. El solenoide con un núcleo apropiado se convierte en un imán (en realidad electroimán).
- Con la configuración apropiada el núcleo puede moverse, dando lugar a las más diversas aplicaciones.

Salidas Digitales – Haciendo música con un solenoide



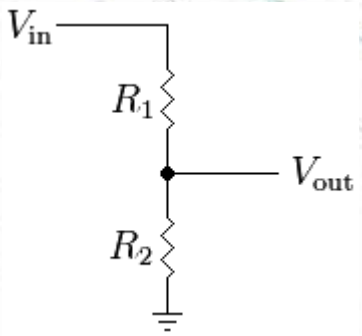
Salidas Digitales – Haciendo música con un solenoide

```
int solenoide = 10; // PIN del solenoide

void setup() {
  pinMode(solenoide, OUTPUT);
// Inicializa el pin 10 como salida digital
}

void loop() {
  digitalWrite(solenoide, HIGH);
   // Activa el solenoide
 delay (10);
  digitalWrite(solenoide, LOW);
 // Desactiva el solenoide
  delay (500);
}
```

```
int solenoide = 10; // PIN del solenoide

void setup() {
  pinMode(solenoide, OUTPUT);
// Inicializa el pin 10 como salida digital
}
void loop() {
  digitalWrite(solenoide, HIGH);
  delay (10);
  digitalWrite(solenoide, LOW);
  delay (500);
  digitalWrite(solenoide, HIGH);
  delay (10);
  digitalWrite(solenoide, LOW);
  delay (200);
}
```

Entradas digitales – Divisor de tensión resistivo

- Un divisor de tensión es una configuración de circuito eléctrico que reparte la tensión de una fuente entre una o más impedancias conectadas en serie.
- Un divisor resistivo es un caso especial donde ambas impedancias, son puramente resistivas.
- Se utiliza para leer valores de sensores.

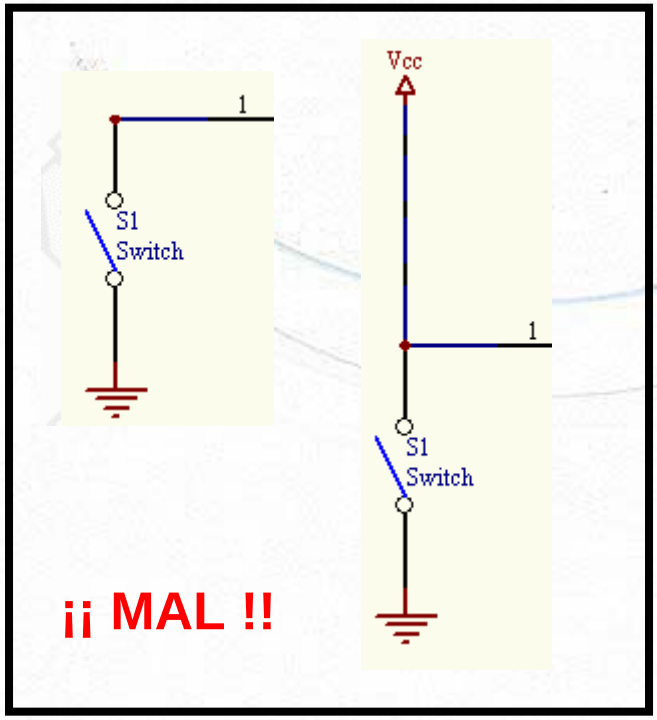


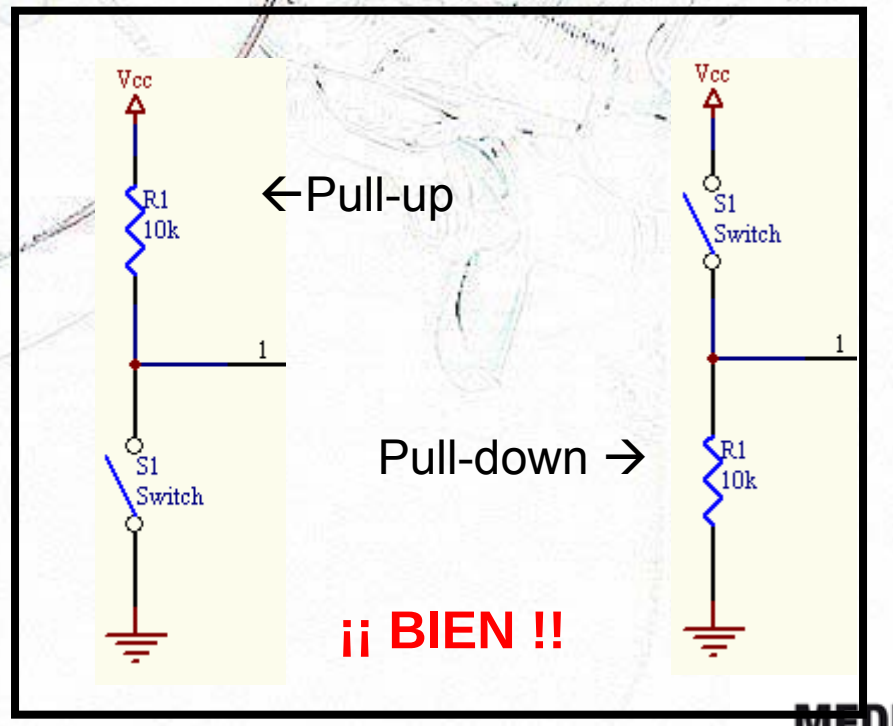
$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in}$$

• Arduino sólo puede leer voltajes, y muchos sensores son resistivos (varían la resistencia). Por eso hay que usar circuitos de este tipo para leer el valor de los sensores.

Entradas digitales – resistencia pull-up y pull-down

- Es un caso específico de divisor resistivo.
- Sirve para leer valores digitales sin que el valor de entrada sea indeterminado.



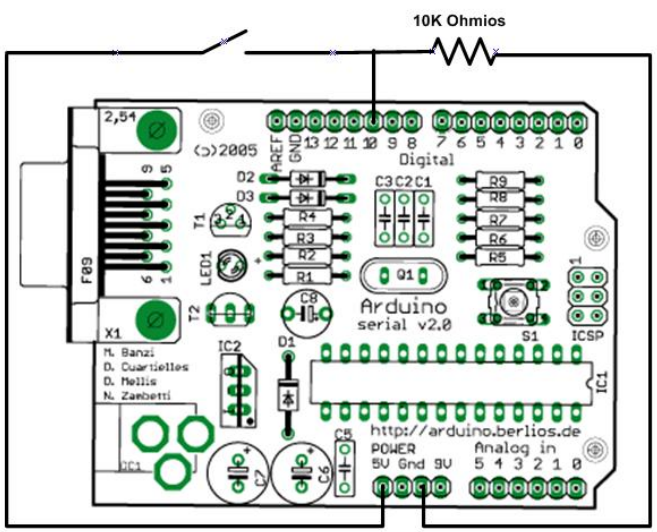


Entradas digitales

Comandos básicos:

- **digitalRead(pin)**,sirve para leer un valor del pin digital que señalemos, el valor podrá ser 1 lógico (HIGH=5v) o 0 lógico (LOW=0v)

Entradas digitales - Pulsador




```
int ledPin = 13; // PIN del LED
int inPin = 10;   // PIN del pulsador
int value = 0;    // Valor del pulsador

void setup() {
  pinMode(ledPin, OUTPUT);
// Inicializa el pin 13 como salida digital
  pinMode(inPin, INPUT);
// Inicializa el pin 10 como entrada digital
}

void loop() {
  value = digitalRead(inPin);
// Lee el valor de la entrada digital
  digitalWrite(ledPin, value);
}
```

- http://www.arduino.cc/es/Tutoriales/Pulsador

Entradas analógicas

Los **pines analógicos** se emplean como entradas analógicas, es decir, se puede recibir tensiones entre 5V y 0 voltios. Los pines analógicos, al contrario que los pines digitales, no necesitan ser declarados como modo INPUT (entrada) o OUTPUT (salida).

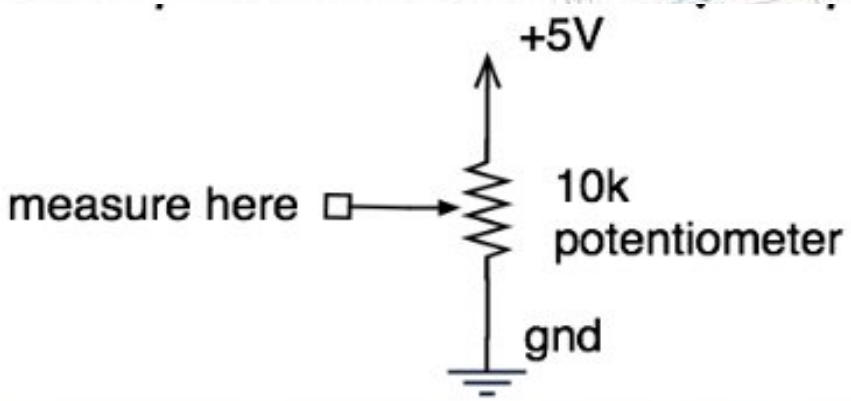
Conversión analógico-->digital(ADC) en Arduino: consiste en transformar un valor de tensión en un número que pueda ser comprendido por un dispositivo de lógica digital. Arduino puede convertir tensiones de 0 a 5 voltios en números enteros que van del 0 al 1023. En otras palabras representa la información en números de 10 bits (resolución).

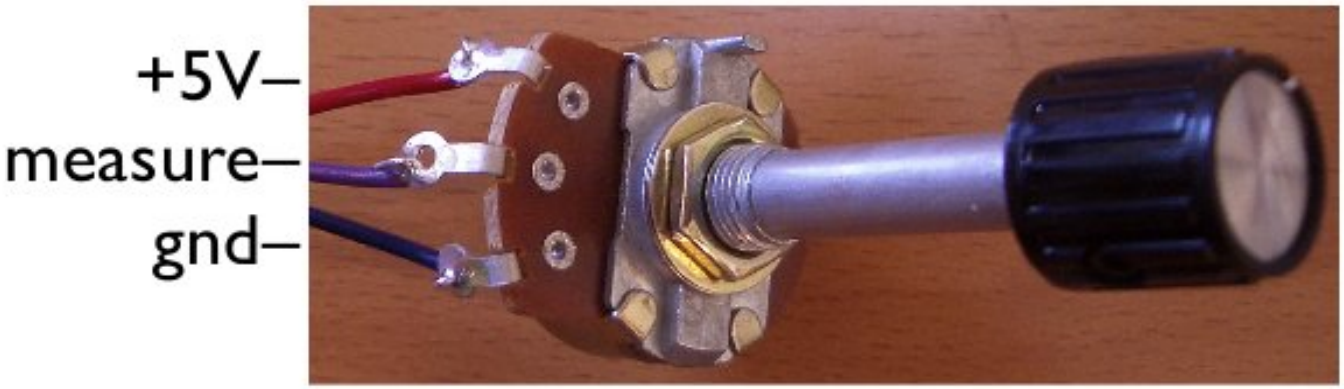
Comandos básicos:

- **analogRead(pin)**, Lee o captura el valor de entrada del especificado pin analógico, la tarjeta Arduino realiza una conversión analógica a digital de 10 bits. Esto quiere decir que mapeará los valores de voltage de entrada, entre 0 y 5 voltios, a valores enteros comprendidos entre 0 y 1023.

Entradas analógicas-Potenciómetro

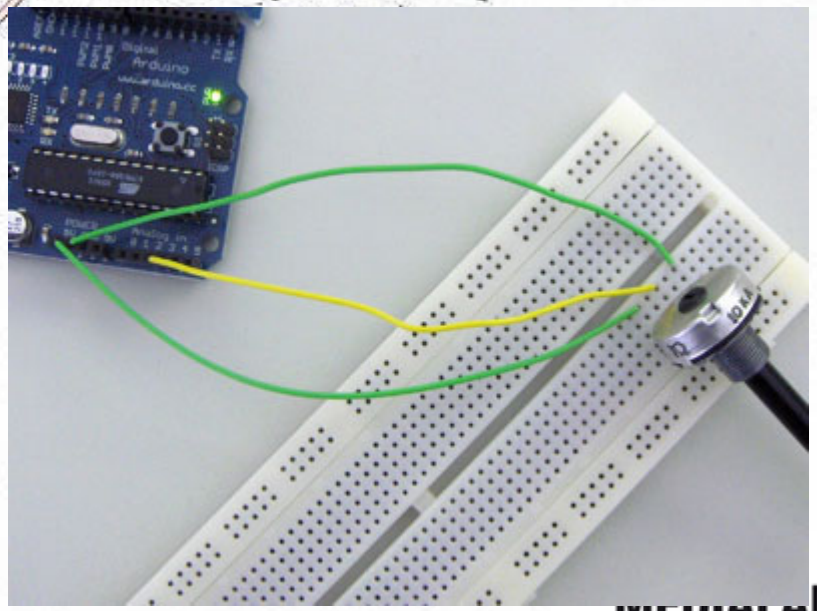
Componentes básicos: el **potenciómetro** es una resistencia que varía en función del giro mecánico de una de sus partes.





Entradas analógicas-Potenciómetro

```
int ledPin = 13;       // LED conectado a pin digital 13

int analogPin = 3;     // potentiómetro conectado a pin analógico 3

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

int threshold = 512; // valor de disparo o umbral (1024/2)

void setup() {

  pinMode(ledPin, OUTPUT);   // asigna modo salida el pin digital 13

}

void loop() {

  val = analogRead(analogPin);   // captura el pin de entrada
  if (val >= threshold) {
    digitalWrite(ledPin, HIGH);      // enciende el LED
  } else {
    digitalWrite(ledPin, LOW);       // apaga el LED
  }

}
```



Entradas analógicas-Comunicación Serie

Comunicación serie:

Dentro del interfaz Arduino, disponemos de la opción **"Monitorización del Puerto Serie"** (último botón a la derecha), que posibilita la visualización de datos procedentes de la tarjeta.

A veces nos interesa poder mandar datos de los sensores hacia el ordenador o incluso poder mandar comandos desde el PC a Arduino. Por ejemplo, si queremos visualizar, la lectura de un potenciómetro.

Si la comunicación serie está activada, no se podrán usar los pines 0 y 1 como entrada/salida digital.

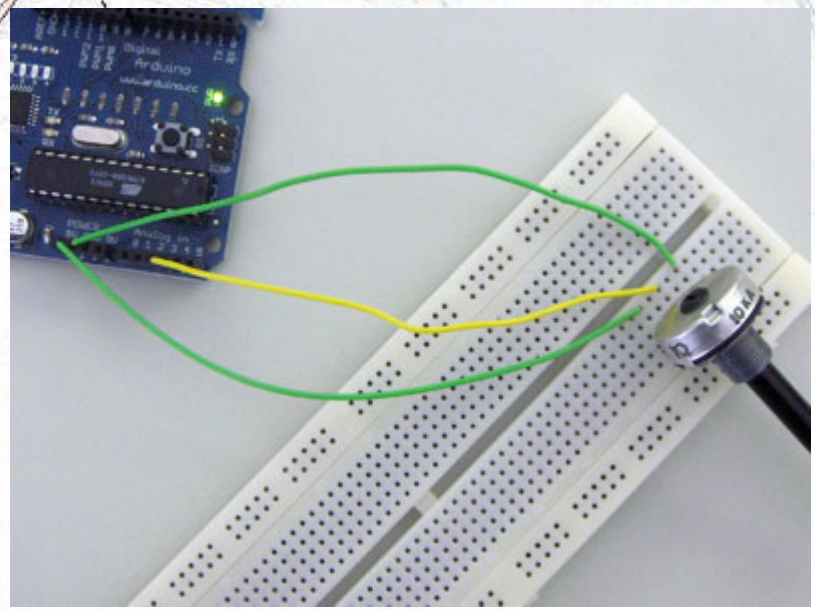
Es recomendable dejar tiempos de espera entre los envíos de datos para ambos sentidos (uso por ejemplo de un delay(10)) ,ya que se puede saturar o colapsar el puerto.

Comandos básicos:

- **Serial.begin(velocidad)** sirve para configurar el puerto serie a una velocidad determinada. Ésta se expresa en bits por segundo. Va en el **setup().**
- **Serial.print(dato,DEC)**: 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 representaría 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 fín o salto de linea por el puerto serie.
- **Serial.print(dato,BYTE):** lanza el valor dato por el puerto serie, en modo Byte o Binario.

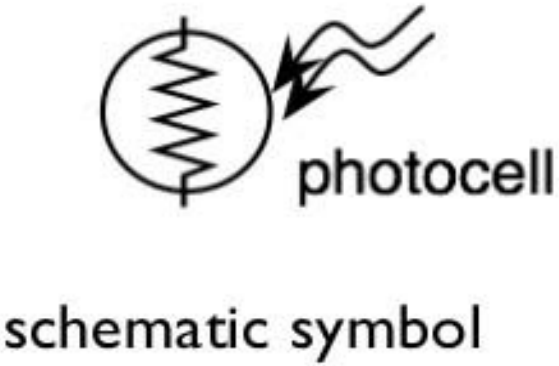
Entradas analógicas -Comunicación Serie

```
int ledPin = 13;       // LED conectado a pin digital 13

int analogPin = 3;   // potentiómetro conectado a pin analógico 3

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

int threshold = 512; // valor de disparo o umbral (1024/2)

void setup() {

  pinMode(ledPin, OUTPUT);   // asigna modo salida el pin digital 13
 Serial.begin(9600);
}

void loop() {

  val = analogRead(analogPin);  // captura el pin de entrada
  Serial.println(val,DEC);
   delay(10); //tiempo de espera entre los envíos de datos
 if (val >= threshold) {
    digitalWrite(ledPin, HIGH);    // enciende el LED
  } else {
    digitalWrite(ledPin, LOW);     // apaga el LED
  }

}
```



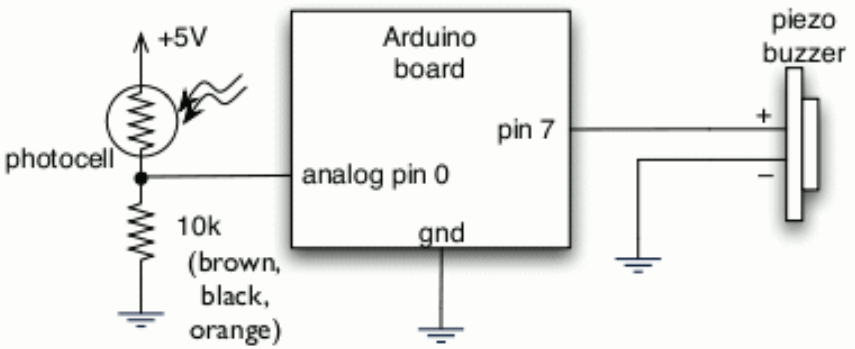
Entradas analógicas -LDR

Componentes: La **LDR** es un tipo de sensor resistivo, es decir, que varía su resistencia en función de la variación de alguna magnitud física. Resistencias que varían su valor con la luz. Se conectan con una configuración que llamamos de **Divisor de Tensión o Resistivo.**



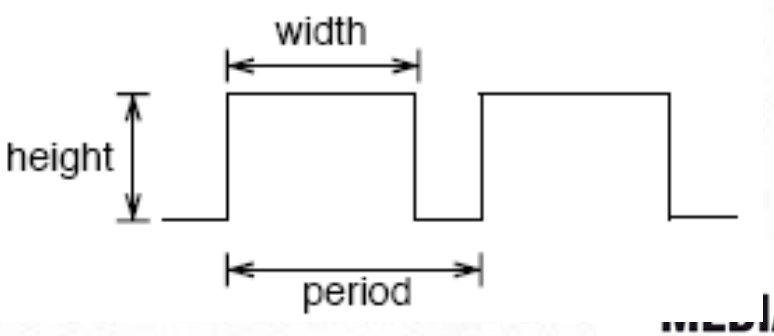
Entradas analógicas -LDR

```
int speakerOut =7;
int val=0;
int LDRPin=0;

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

void loop() {
  digitalWrite(speakerOut, LOW);
  val=val*2;
  //val=val/2;
  val=analogRead(LDRPin);
  digitalWrite(speakerOut,HIGH);
  delayMicroseconds(val);
  digitalWrite(speakerOut, LOW);
  delayMicroseconds(val);
}
```



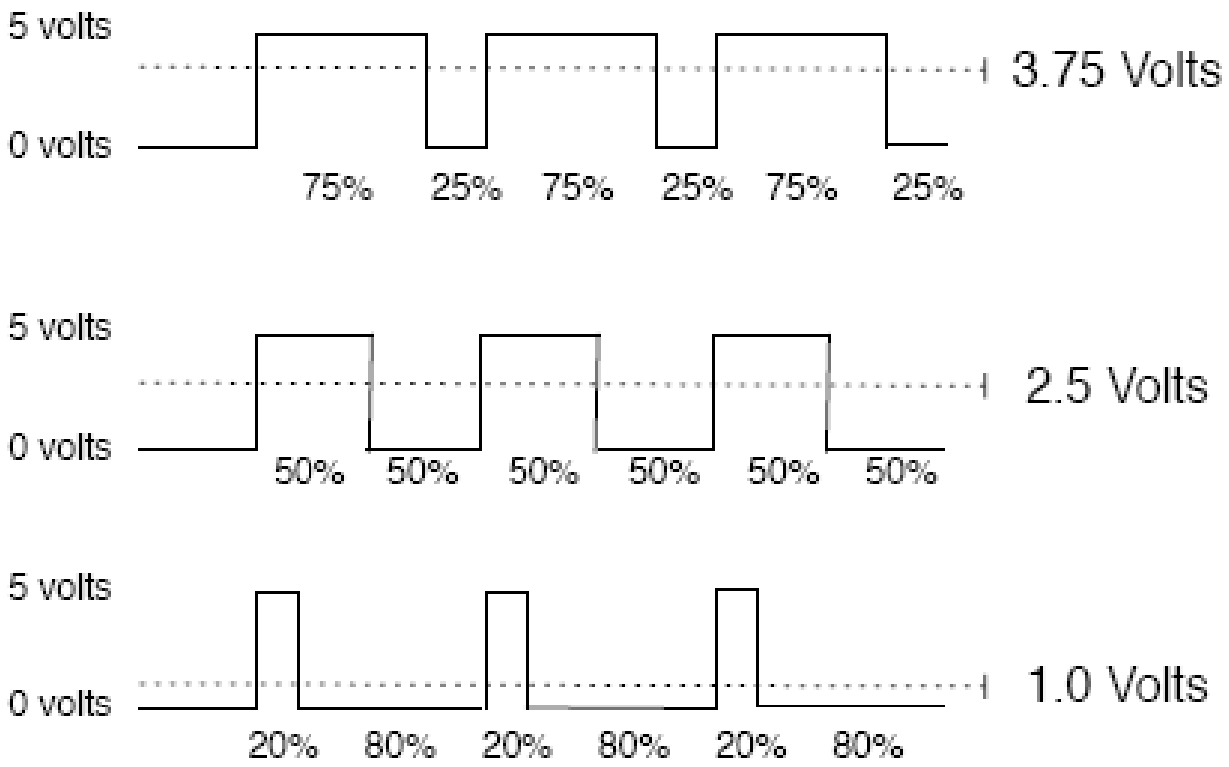
Salidas analógicas – PWM

- El chip Atmega de Arduino, como muchos microcontroladores, no puede generar una salida analógica, por lo que tiene que utilizar la técnica PWM (Pulse Width Modulation).
- PWM consiste en simular mediante una salida digital un salida analógica.
- Tres características de las señales PWM:
  - Altura (heigth)
  - Anchura (width)
  - Periodo (period)



Salidas analógicas – PWM

- El voltaje de salida es la media del tiempo que está a 5V con respecto del tiempo que está a 0V



Salidas analógicas

Arduino dispone de varios pines para generar salidas PWM, a través algunos de los pines digitales. Dependiendo del modelo de la placa y sobre todo del chip Atmega de que disponga la placa tendremos 3 o 6 salidas **PWM,** que están marcadas en la placa:

- Arduino serie, Arduino NG (chip Atmega8) → 3 pines digitales para PWM: **9, 10 y 11**.
- Arduino NG (chip Atmega168), Arduino Diecimilla → 6 pines digitales para PWM: **3, 5, 6, 9, 10 y 11.**

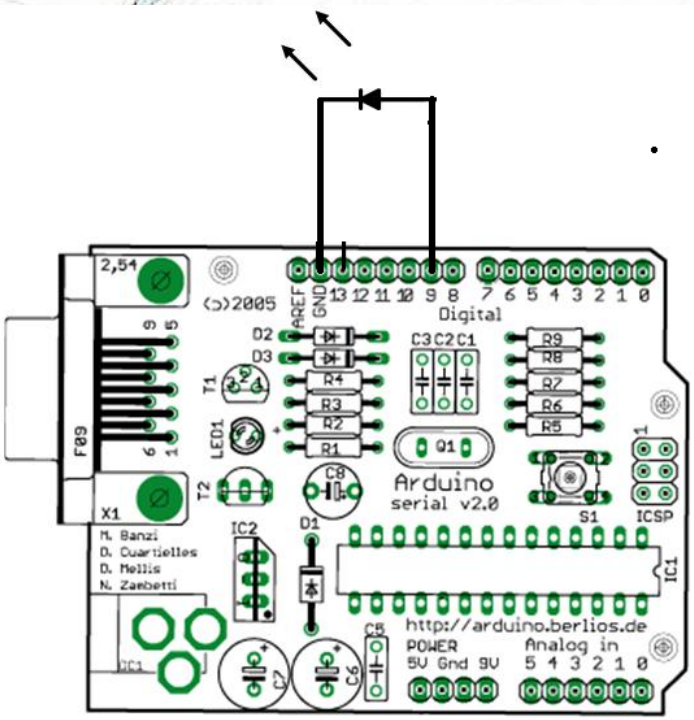
A diferencia de las entradas analógicas, en las que el conversor analógico digital nos daba un valor entre **0 y 1023**, para generar una salida digital el rango es de **0 a 255**. Donde **0** equivale a **0V** y **255** a **5V**

Los pines analógicos, al contrario que los pines digitales, no necesitan ser declarados como modo INPUT(entrada) o OUTPUT (salida).

Comandos básicos:

- **analogWrite(pin, value)**, Escribe el valor especificado en el pin **PWM** correspondiente. Dicho valor, como se ha mencionado, tiene que estar entre **0 y 255**.

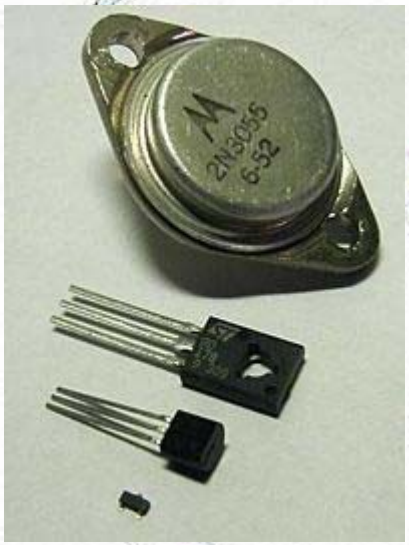
Salidas analógicas – Intensidad de un LED

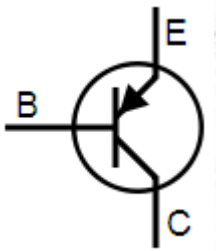



```
int valor = 0;    // variable que contiene el valor
int ledpin = 9;  // LED conectado al PIN 9

void setup() { } // No es necesario

void loop() {
  for(valor = 0 ; valor <= 255; valor +=5) {
// ilumina de menos a más
    analogWrite(ledpin, valor);
    delay(30);
// espera 30 ms para que el efecto sea visible
  }
  for(valor = 255; valor >=0; valor -=5) {
// ilumina de más a menos
   analogWrite(ledpin, valor);
   delay(30);
  }
}
```

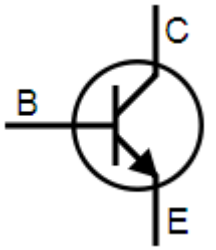
- http://www.arduino.cc/es/Ejercicios/AjustarLuz

Salidas analógicas – Transistor (BD137)





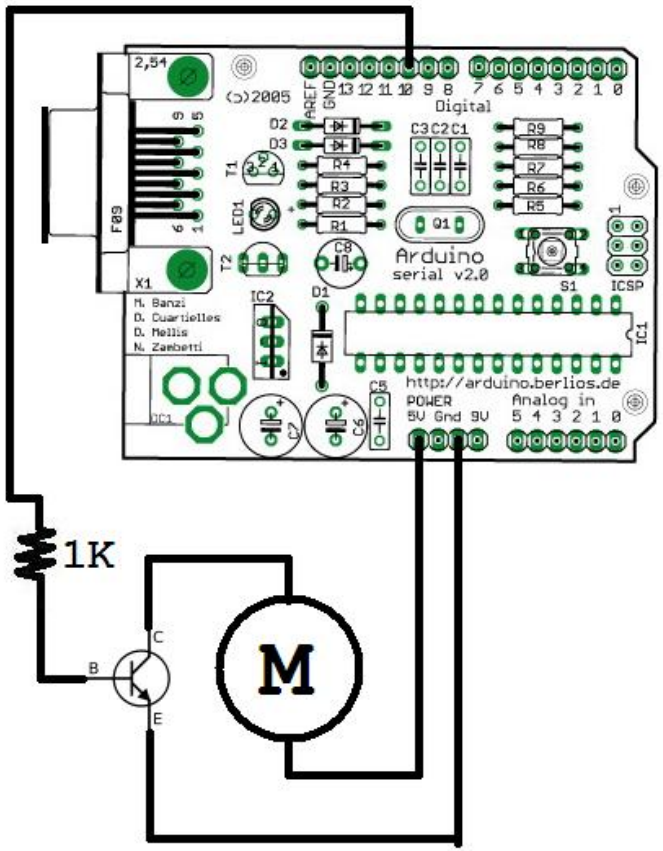
PNP



NPN

- El Transistor es un dispositivo electrónico semiconductor que cumple funciones de amplificador, oscilador, conmutador o rectificador.
- Formado por:
  - B: Base
  - C: Colector
  - E: Emisor
- De manera simplificada, la corriente que circula por el "colector" es función amplificada de la que se inyecta en el "emisor", pero el transistor sólo gradúa la corriente que circula a través de sí mismo, si desde una fuente de corriente continua se alimenta la "base" para que circule la carga por el "colector", según el tipo de circuito que se utilice.

Salidas analógicas – Motor + BD137




```
int valor = 0;    // variable que contiene el valor
int motor = 10;  // motor conectado al PIN 10

void setup() { } // No es necesario

void loop() {
  for(valor = 0 ; valor <= 255; valor +=5) {
// sube la velocidad
    analogWrite(motor, valor);
    delay(30);
// espera 30 ms para que el efecto sea visible
  }
  for(valor = 255; valor >=0; valor -=5) {
// baja la velocidad
   analogWrite(motor, valor);
   delay(30);
  }
}
```

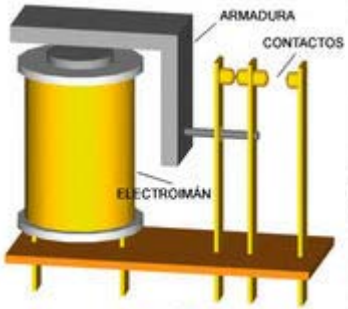
- http://www.arduino.cc/es/Ejercicios/Motores

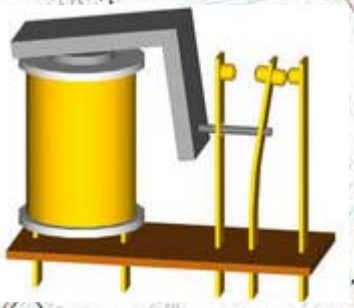
ANEXOS

- Si te da tiempo y quieres seguir cocinando, aquí tienes unas recetas extra ...

Salidas Digitales - Activar Bombillas

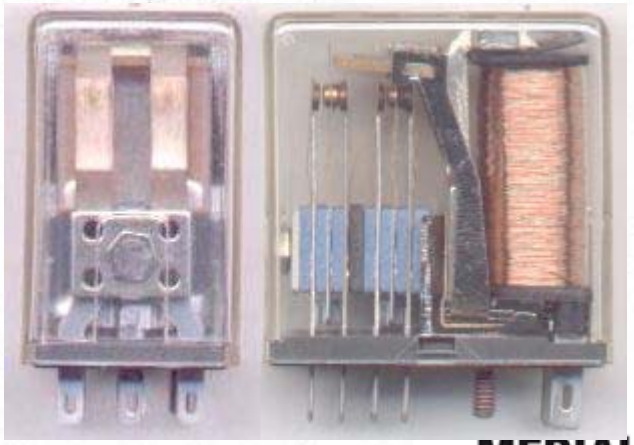
Componentes: el **Relé** es un dispositivo, que funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.





De esta forma, podremos separar dos circuitos que funcionen con voltajes diferentes. Uno a 5V (Arduino) y otro a 220V (la bombilla). Para nuestro ejemplo, utilizaremos un circuito de 220V con un máximo de 10A.

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

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

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

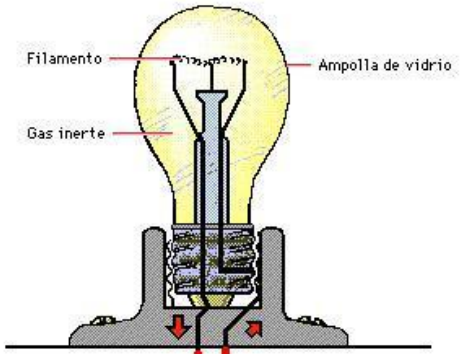


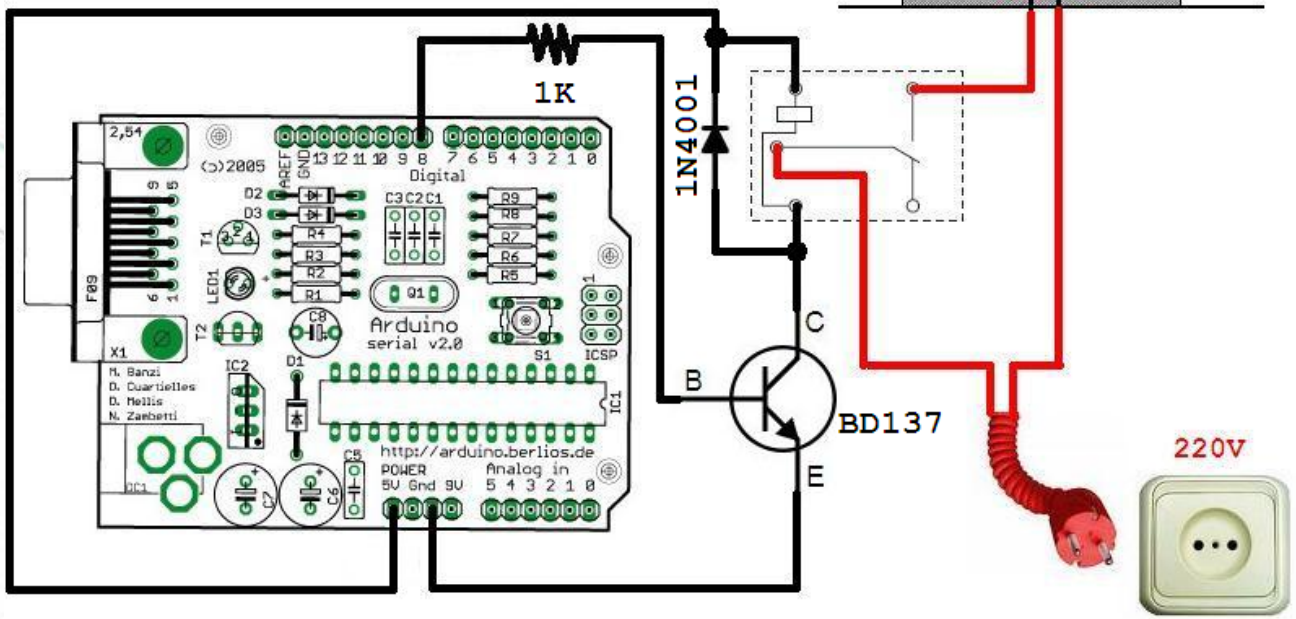
Salidas Digitales - Activar Bombillas

Como se ve en el esquema inferior hay dos circuitos. El del cableado NEGRO funciona a 5V de DC y el del cableado ROJO a 220V de AC.

Relé

5V DC
220V AC

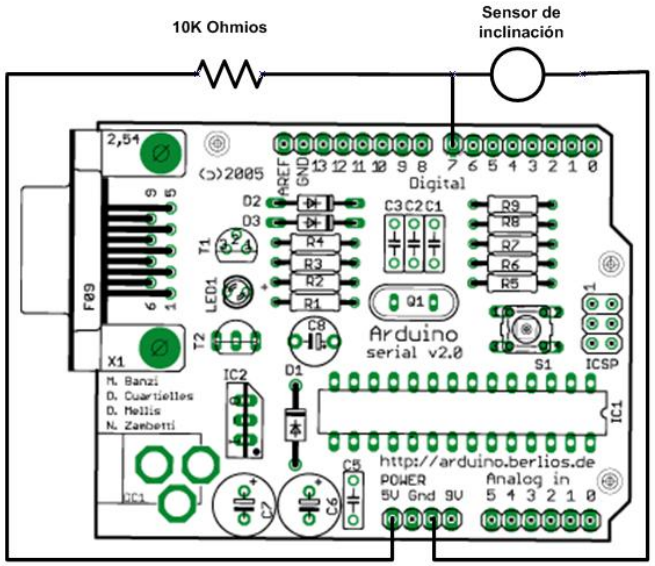




http://www.youtube.com/watch?v=TCLIGSijFtU

MEDIALAB PRADO

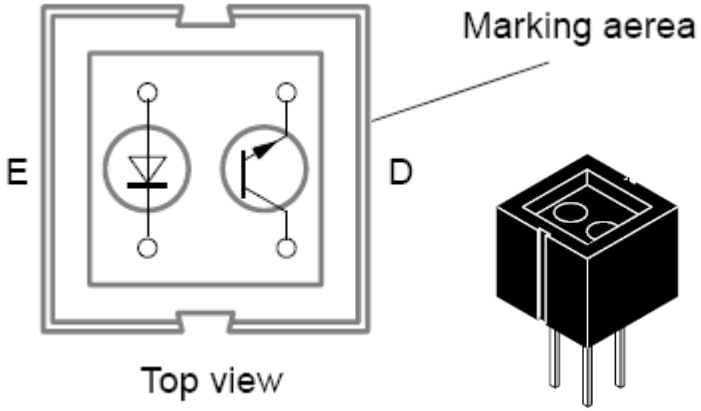
Entradas digitales – Sensor TILT

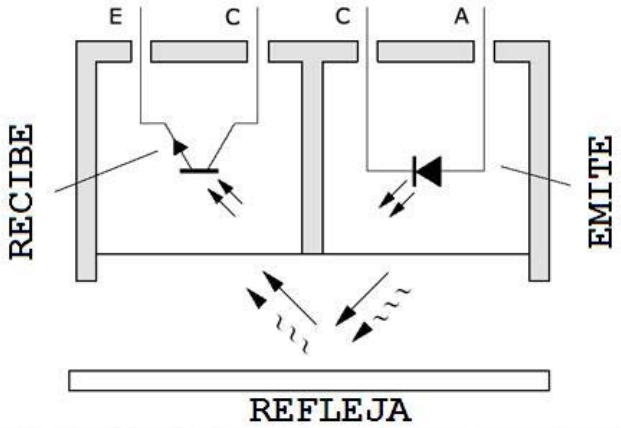



```
int ledPin = 13; // PIN del LED
int inPin = 7;    // PIN del sensor TILT
int value = 0;    // Valor del pulsador

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

void loop() {
  value = digitalRead(inPin);
// Lee el valor de la entrada digital
  digitalWrite(ledPin, value);
}
```

- http://www.arduino.cc/es/Tutoriales/SensorTilt

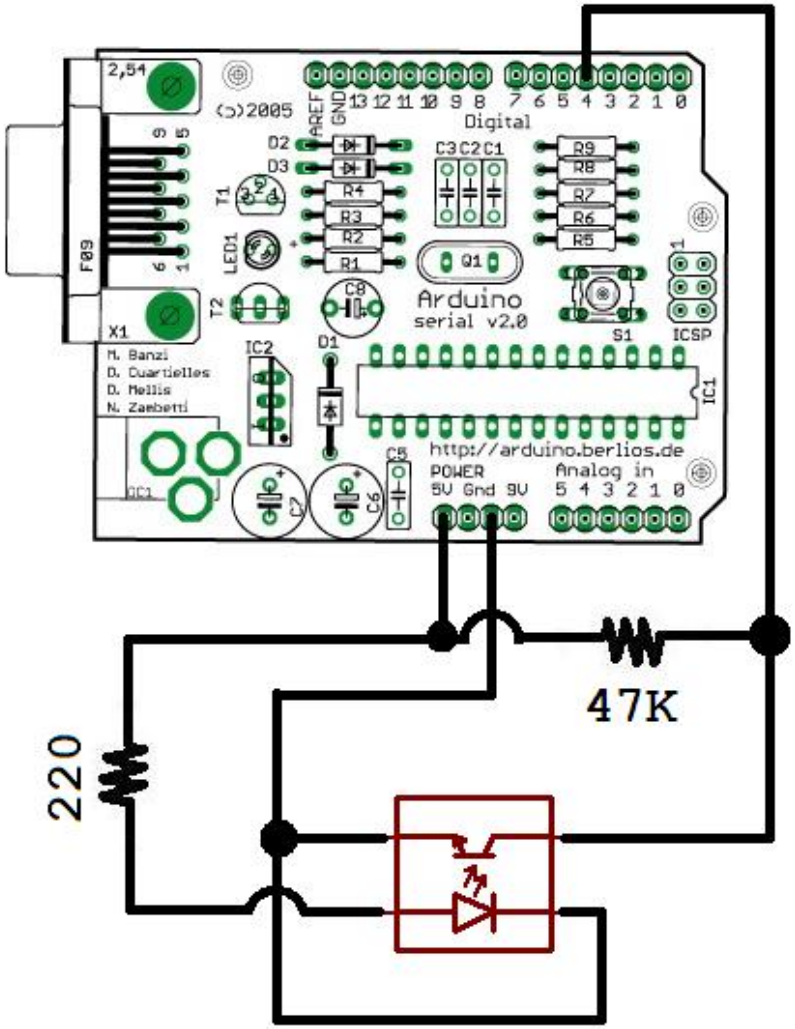
Entradas digitales – CNY70





- Sensor que detecta la reflexión de la luz a muy poca distancia.
- Formado por:
  - LED → emite luz.
  - Fototransistor → se activa si detecta luz.
- Se utiliza para robots sigue-lineas

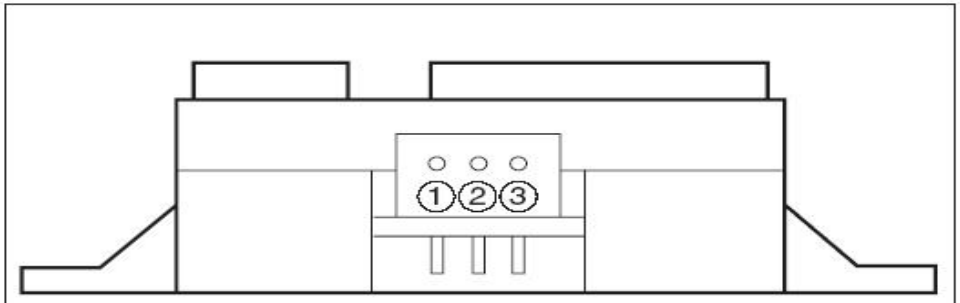
Entradas digitales – CNY70




```
int ledPin = 13; // PIN del LED
int inPin = 4;     // PIN del CNY70
int value = 0;    // Valor del CNY70

void setup() {
  pinMode(ledPin, OUTPUT);
// Inicializa el pin 13 como salida digital
  pinMode(inPin, INPUT);
// Inicializa el pin 4 como entrada digital
}

void loop() {
  value = digitalRead(inPin);
// Lee el valor de la entrada digital
  digitalWrite(ledPin, value);
}
```

Entradas analógicas -Sensor de distancia

Componentes: El sensor que vamos a utilizar en el ejemplo es el **GP2D120** de Sharp. Un sensor analógico de distancia que funciona con infrarrojos.



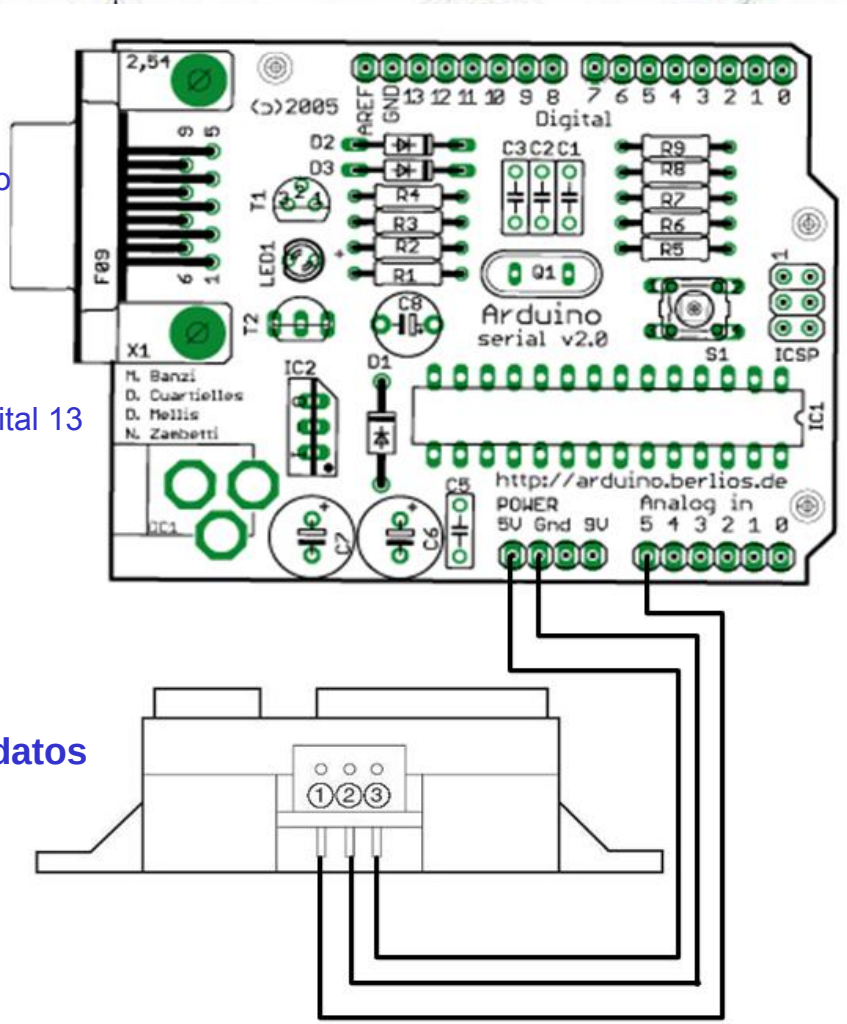
PIN	SIGNAL NAME
1	V_O
2	GND
3	V_{CC}

GP2D120-8

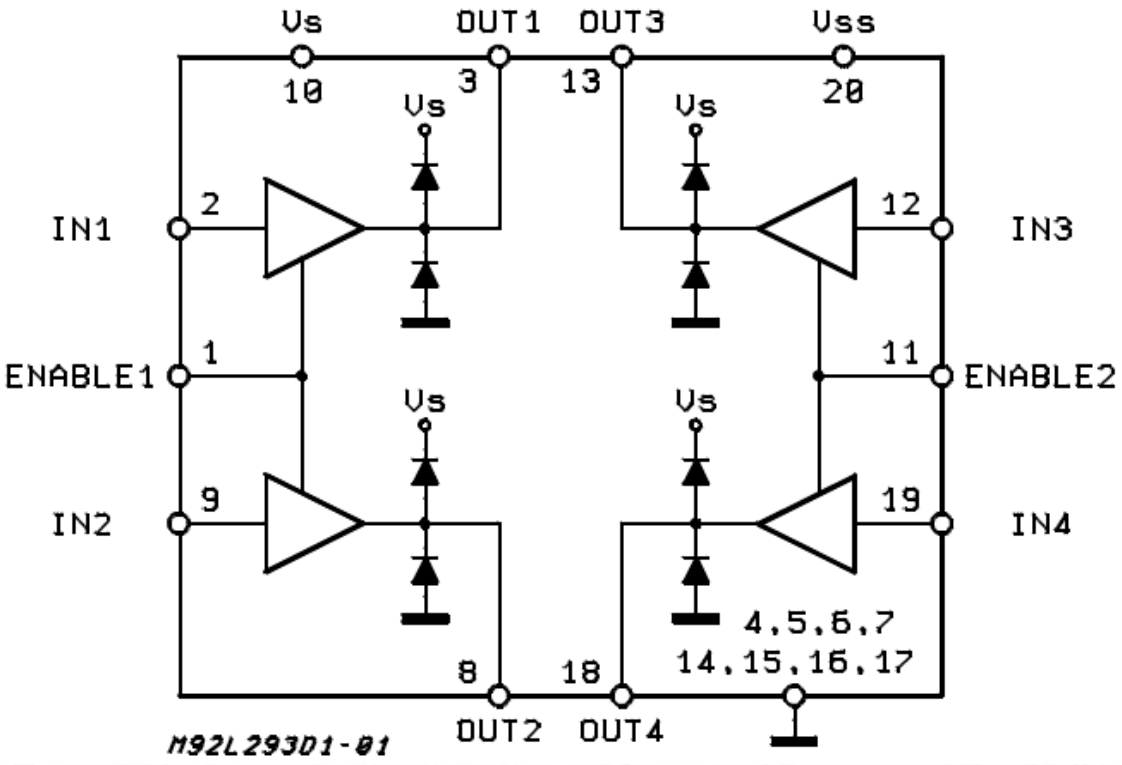
La conexión del sensor a la placa Arduino es muy sencilla. Tan sólo hay que conectarlo a la alimentación de la placa (Vcc y GND) y la señal que proporciona el sensor (Vo) a la entrada analógica 5 de la placa Arduino.

Entradas analógicas -Sensor de distancia

```
int ledPin = 13;      // LED conectado a pin digital 13

int analogPin = 5; // sensor conectado a pin analógico 5

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

int threshold =  100 ; // valor de disparo o umbral

void setup() {

  pinMode(ledPin, OUTPUT);   // asigna modo salida el pin digital 13
 Serial.begin(9600);
}

void loop() {

  val = analogRead(analogPin); // captura el pin de entrada
Serial.println(val,DEC);
delay(10); //tiempo de espera entre los envíos de datos
 if (val >= threshold) {
    digitalWrite(ledPin, HIGH);    // enciende el LED
  } else {
    digitalWrite(ledPin, LOW);     // apaga el LED
  }

}
```

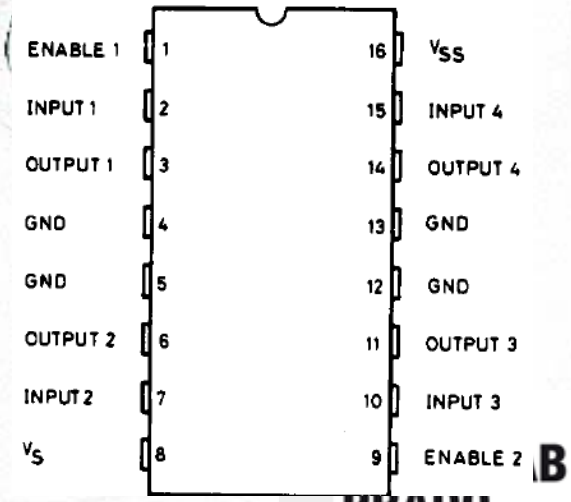


Salidas analógicas – Driver L293



Esquema del puente H

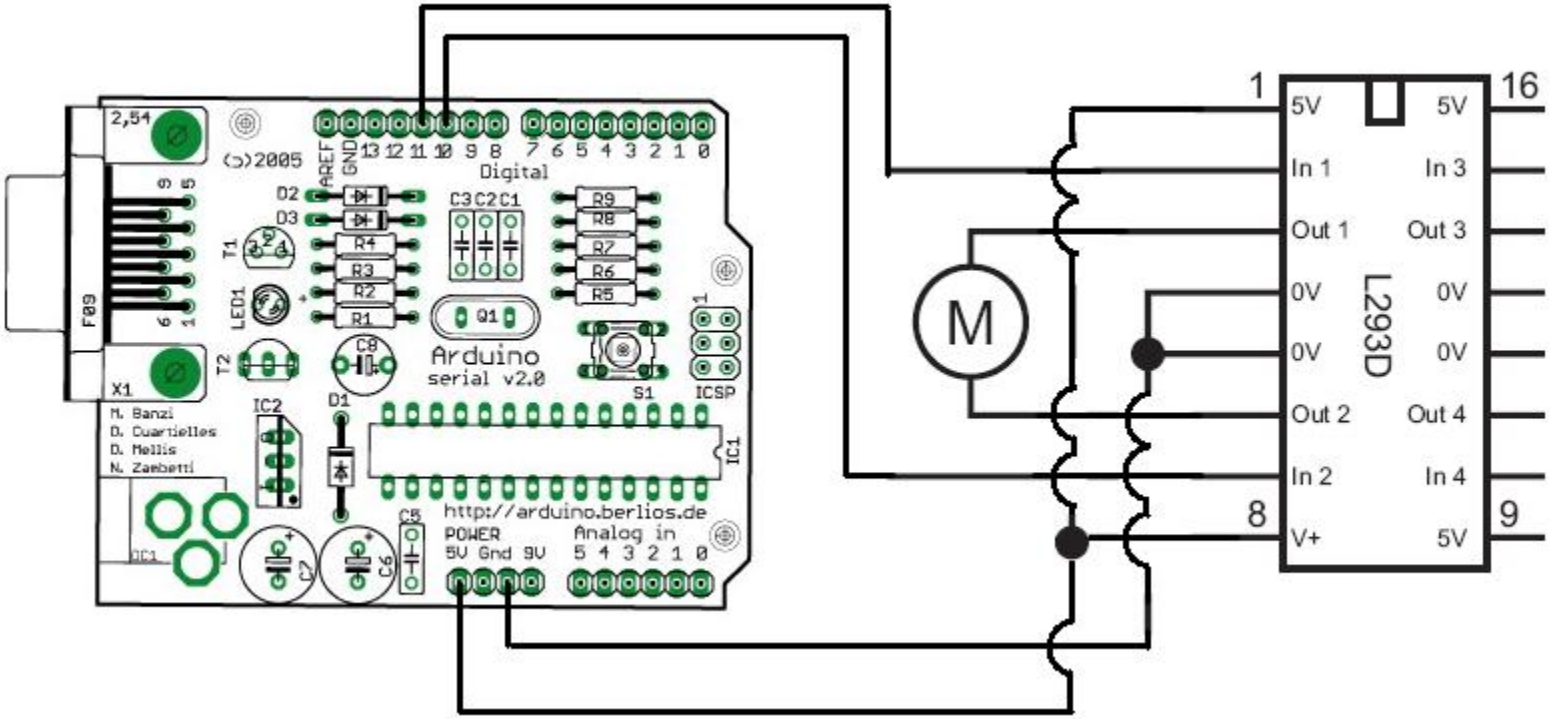
- Circuito integrado que permite controlar dos motores basado en el puente H.



Pines del L293 →

Salidas analógicas – Motor + L293D

- Vamos a controlar tanto la velocidad de giro como el sentido de dicho giro mediante el driver L293D.
- Es muy importante tener en cuenta que, **siempre**, una de las dos salidas tiene que estar a 0V si no queremos provocar un cortocircuito.



Salidas analógicas – Motor + L293D

```
int valor = 0;                  // variable que contiene el valor
int motorAvance = 10;           // Avance motor --> PIN 10
int motorRetroceso = 11;        // Retroceso motor --> PIN 11

void setup() { }                // No es necesario

void loop() {
  analogWrite(motorRetroceso, 0);          // Motor hacia delante ... sube la velocidad
  for(valor = 0 ; valor <= 255; valor+=5) {
    analogWrite(motorAvance, valor);
    delay(30);
  }
  for(valor = 255; valor >=0; valor-=5) {   // Motor hacia delante ... baja la velocidad
   analogWrite(motorAvance, valor);
   delay(30);
  }
  analogWrite(motorAvance, 0);             // Motor hacia detrás ... sube la velocidad
  for(valor = 0 ; valor <= 255; valor+=5) {
    analogWrite(motorRetroceso, valor);
    delay(30);
  }
  for(valor = 255; valor >=0; valor-=5) {   // Motor hacia detrás ... baja la velocidad
   analogWrite(motorRetroceso, valor);
   delay(30);
  }
}
```