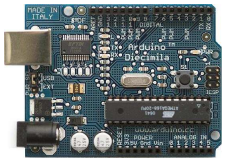
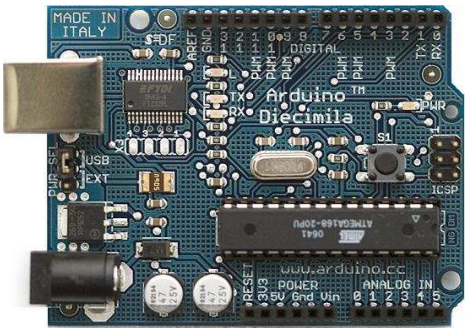
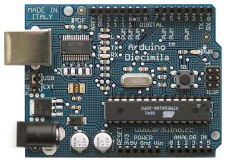


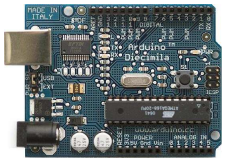


Prácticas con Arduino Nivel I

José Manuel Ruiz Gutiérrez



Índice de Aplicaciones practicas



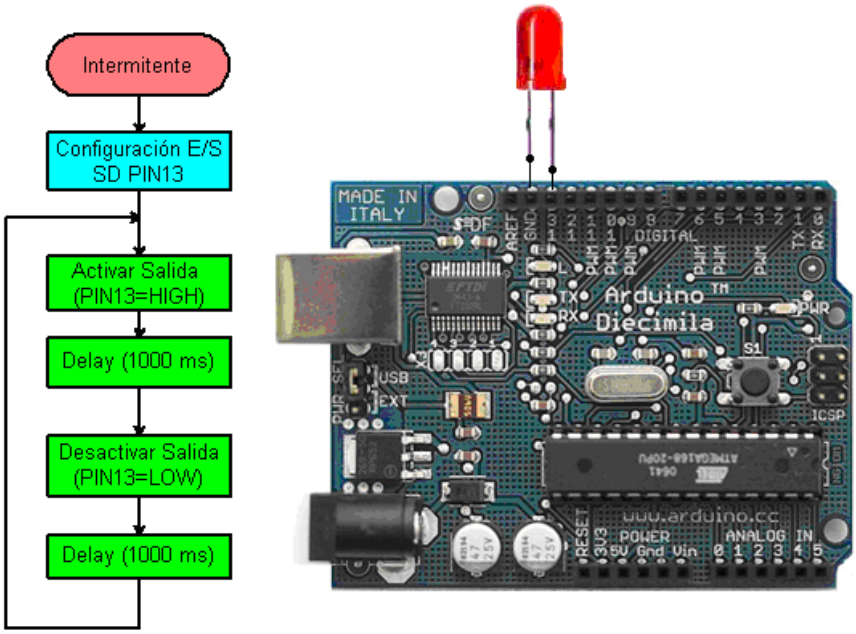
21. Timbre de llamada
22. Enciende y apaga un número de veces un LED
23. Control de un motor de cc con un transistor
24. Control de un motor de cc con el driver L293D
25. Control de un motor: velocidad variable y sentido de giro variable
26. Control de un motor: velocidad variable y sentido de giro variable (2ª opción)
27. Utiliza un relé para encender dispositivos de 220V

Nota: *Los materiales recogidos en este documento, listados de código y algunos textos explicativos han sido recogidos en la pagina Web oficial de Arduino (http://www.arduino.cc/es/ y http://www.arduino.cc), correspondiendo al autor de este documento la labor de compilación, traducción e incorporación de imágenes, organigramas y esquemas de funcionamiento.*

1. Intermitente

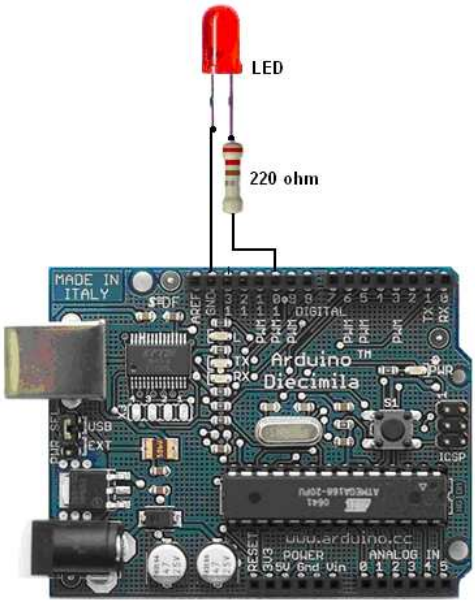
Se trata de realizar un ejercicio básico que consiste en encender y a pagar un led que conectamos en el PIN 13 de Arduino que lo configuramos como salida. El tiempo de encendido y apagado es de 1 segundo.

Organigrama y Esquema



Organigrama y esquema de conexionado con la tarjeta Arduino

Obsérvese que se ha colocado el diodo led sin resistencia en serie dado que el PIN13 de Arduino ya lleva incorporada una resistencia interior, en el caso de colocar el diodo LED en otra salida deberíamos colocar una resistencia de al entre 220 y 500 ohmios dependiendo del consumo de corriente del diodo



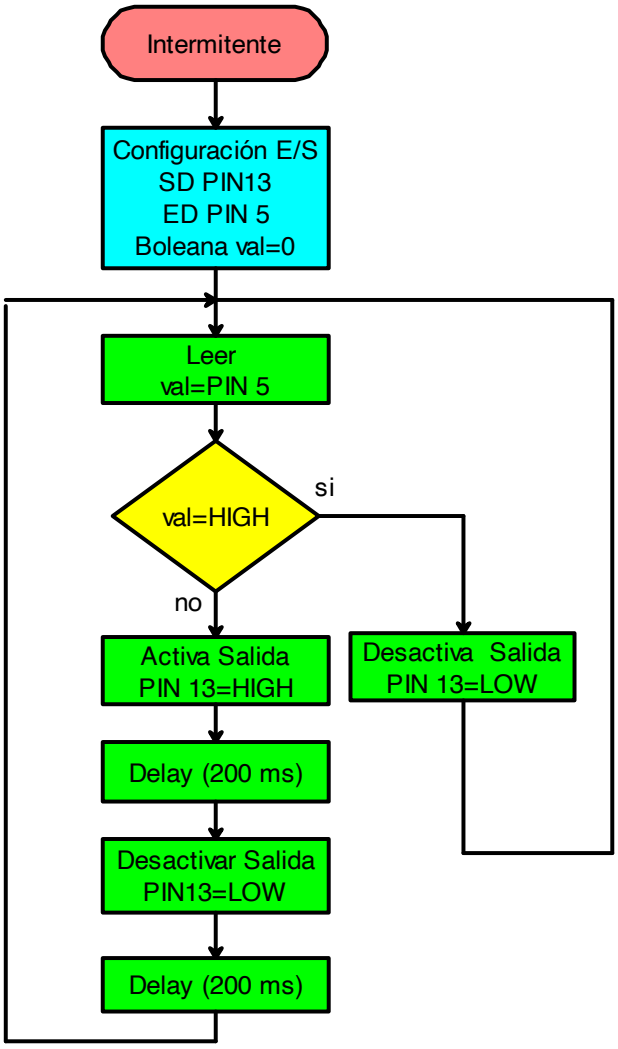
Conexionado a realizar en el caso de realizar la salida por el PIN 10

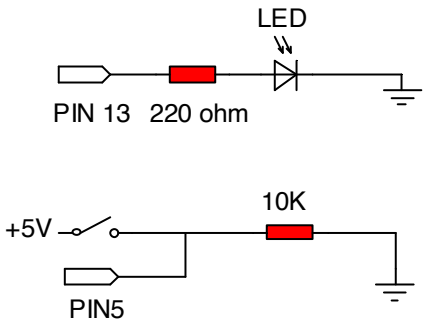
Programa

```
/*
 * Intermitente
 *
 * Ejemplo básico con Arduino. Encendido y apagado de un led
 * con una cadencia de 1 sg. usando el PIN 13 como salida
 * no es necesario usar una resistencia para el led
 * la salida 13 de Arduino la lleva incorporada.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */
int ledPin = 13;                // Definición de la salida en el PIN 13
void setup()                    //Configuración
{
  pinMode(ledPin, OUTPUT);      // designa la salida digital al PIN 13
}

void loop()                      // bucle de funcionamiento
{
  digitalWrite(ledPin, HIGH);   // activa el LED
  delay(1000);                  // espera 1 seg. (tiempo encendido)
  digitalWrite(ledPin, LOW);    // desactiva el LED
  delay(1000);                  // espera 1 seg. (tiempo apagado)
}
```

2. Alarma-1

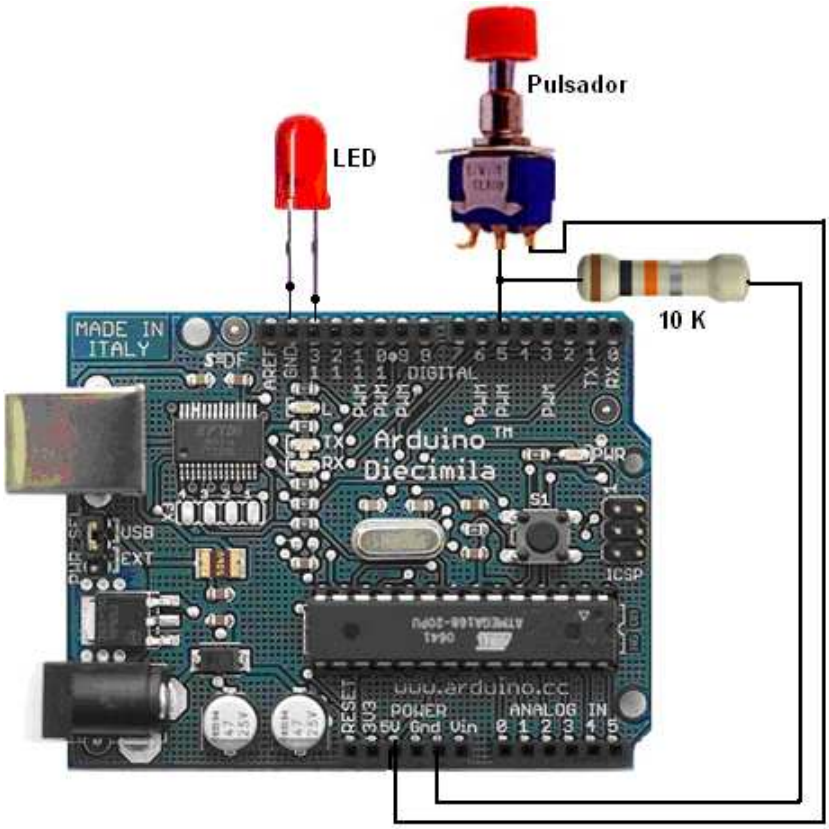
Cuando se pulsa el pulsador (entrada5 a “0”) se enciende y se apaga de forma intermitente la salida 13





Funcionamiento:

Cuando la E5 = 1 Entonces S13 = 0
Cuando la E5 = 0 Entonces S13 = 0-1 (Intermitente 200,200 ms)

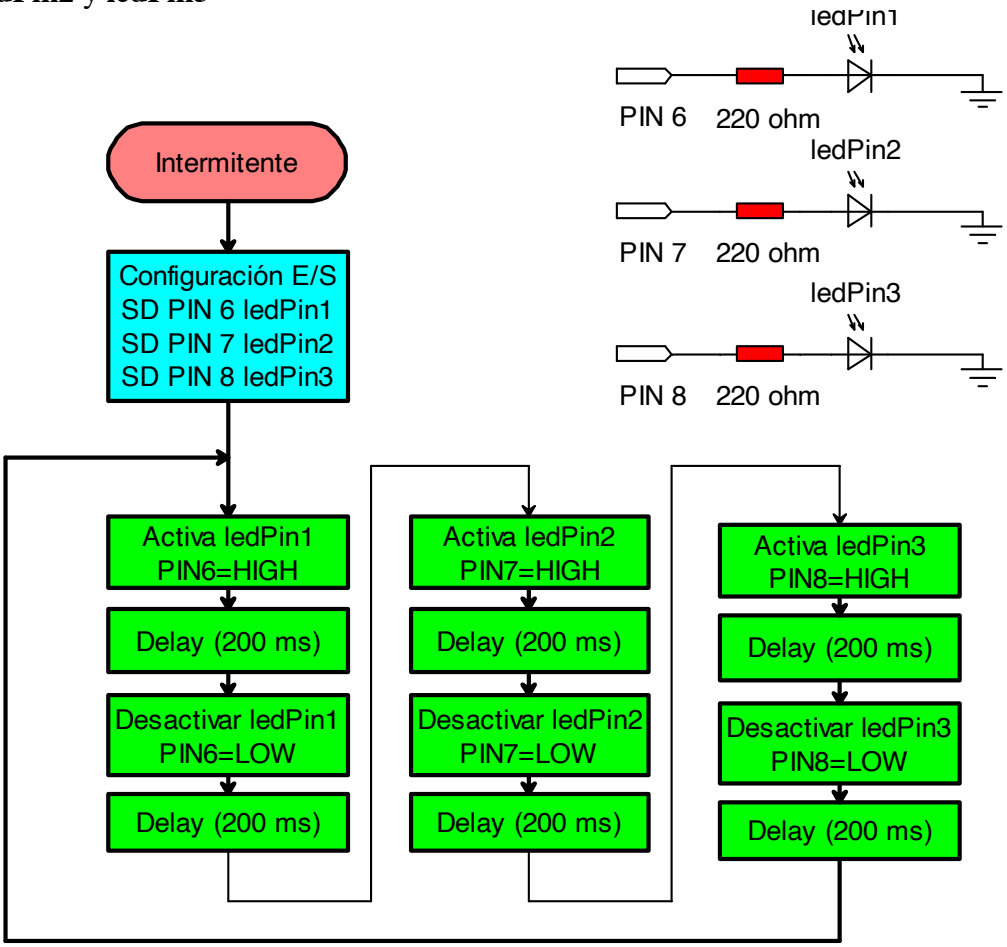


Programa:

```
int ledPin= 13; // choose the pin for the LED
int inPin= 5; // choose the input pin (for a pushbutton)
int val= 0; // variable for reading the pin status
void setup() {
pinMode(ledPin, OUTPUT); // declare LED as output
pinMode(inPin, INPUT); // declare pushbutton as input
}
void loop(){
val= digitalRead(inPin); // lee valor de entrada
if(val== HIGH) { // chequea si el valor leído es "1" (botón presionado)
digitalWrite(ledPin, LOW); // pone el LED en OFF
} else{
digitalWrite(ledPin, LOW); // parpadea el LED
delay(200);
digitalWrite(ledPin, HIGH);
delay(200);
}
}
```

3. Secuencia Básica de 3 LEDs

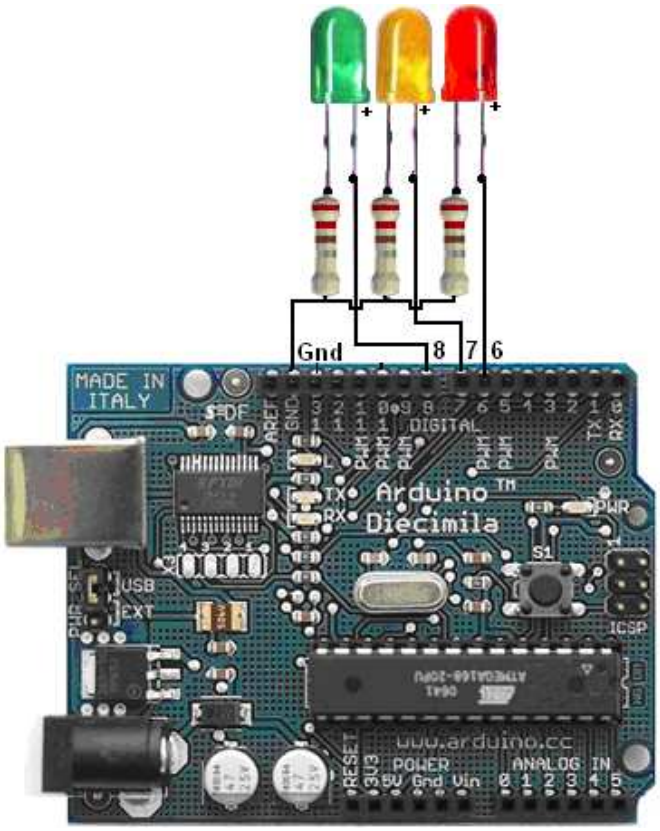
Se trata de encender y apagar 3 LEDs colocados en las salidas **6**, **7** y **8 (PIN6, PIN7 y PIN8)** con una cadencia de **200 ms**. Las variables asignadas a cada led son **ledPin1**, **ledPin2** y **ledPin3**



Programa

```
// Encendido y apagado de 3 LEDs

int ledPin1 = 6; // Define las salidas de los LED´s
int ledPin2 = 7;
int ledPin3 = 8;

void setup() { // Configura las SALIDAS
pinMode(ledPin1, OUTPUT); // declarar LEDs como SALIDAS
pinMode(ledPin2, OUTPUT);
```

```
pinMode(ledPin3, OUTPUT);
digitalWrite(ledPin1, LOW); // Apaga los LEDs
digitalWrite(ledPin2, LOW);
digitalWrite(ledPin3, LOW);
}

void loop(){ //Bucle de Funcionamiento
digitalWrite(ledPin1, HIGH); // Apaga y enciende los leds cada 200 ms
delay(200);
digitalWrite(ledPin1, LOW);
digitalWrite(ledPin2, HIGH);
delay(200);
digitalWrite(ledPin2, LOW);
digitalWrite(ledPin3, HIGH);
delay(200);
digitalWrite(ledPin3, LOW);
}
```



Montaje con la tarjeta Arduino

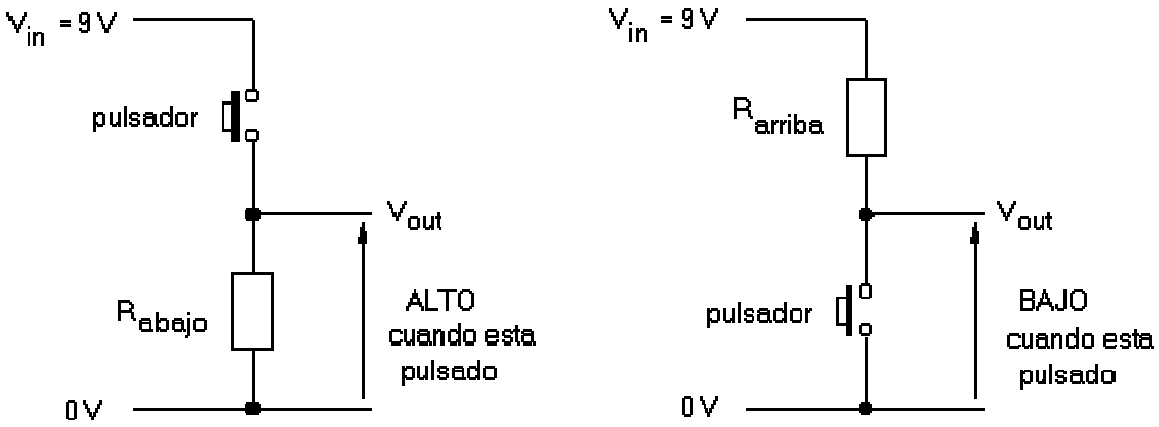
4. Lectura de un pulsador

Descripción del ejercicio

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.

Ejemplo:



La resistencia R_{abajo}(pull-down) en el primer circuito fuerza a Vout, llegando a nivel CERO, hasta que se acciona el pulsador. Este circuito entrega una tensión alta, cuando se presiona el pulsador. Un valor para la resistencia de 10 k es adecuada.

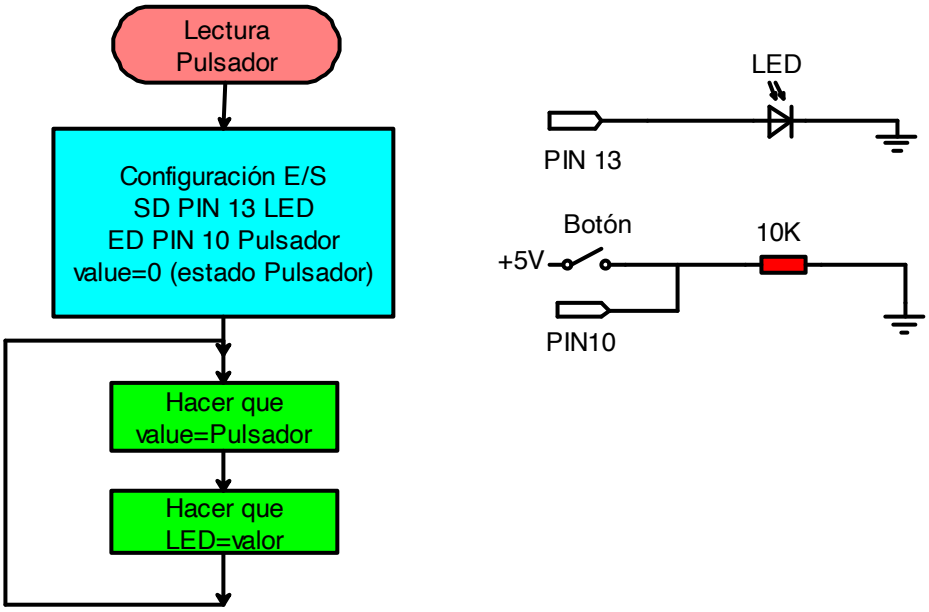
En el segundo circuito, la resistencia R_{arriba} (pull-up) fuerza a nivel ALTO a Vout, mientras no se actúe sobre el pulsador. Al presionar el pulsador, se conecta Vout directamente con 0 V. Es decir, este circuito entrega un nivel BAJO cuando se presiona el pulsador.

Elementos necesarios:

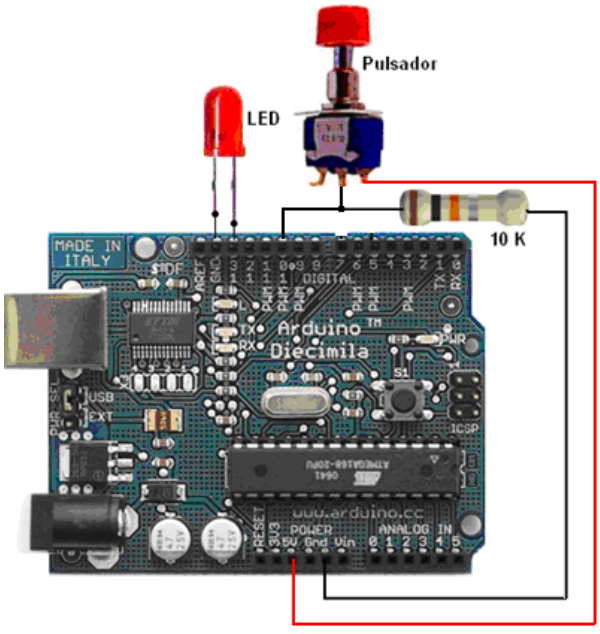
- Un resistencia de 1K Ohmios.
- Un pulsador.
- Un diodo LED
- Cables para realizar las conexiones.

Se utiliza una resistencia pull-dowm, junto con un pulsador, para conectarla 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 10 será de 0 voltios (LOW) en caso contrario será de + 5 voltios (HIGH).

En una placa protoboard debe haber una resistencia de 10K conectada entre el pin de entrada y tierra como se ve el esquema y foto inferiores.



Esquema



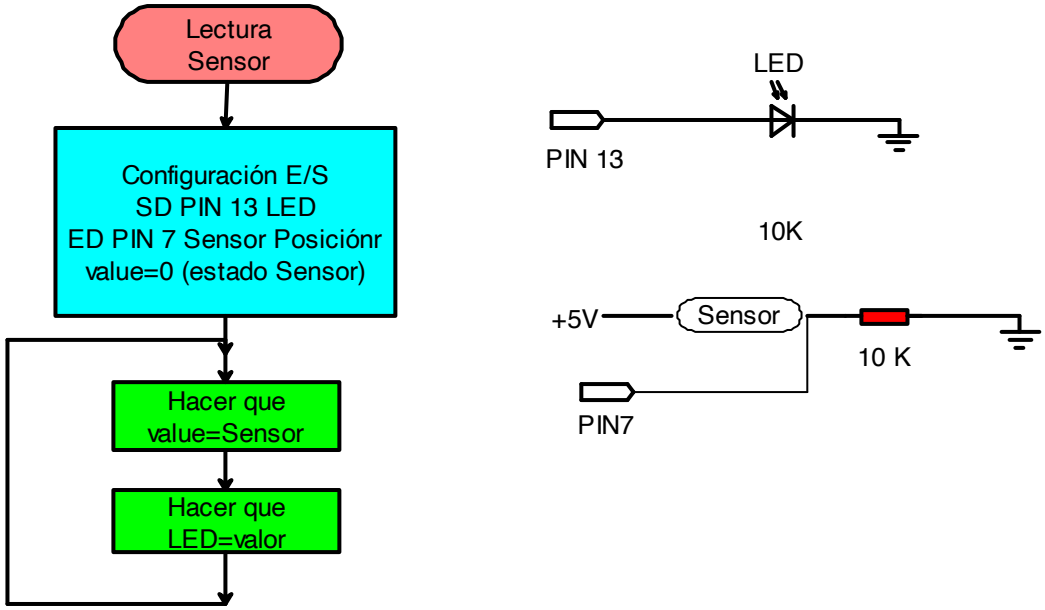
Código fuente

```
/* Pulsador
 * --------
 *
 * Detecta si el botón ha sido presionado o no
 * y enciende el LED en consecuencia.
 *
 * Massimo Banzi
 *
 */
 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);
}
```

5. Lectura de un sensor de inclinación

Descripción del ejercicio

El sensor de inclinación es un componente que puede detectar la inclinación de un objeto. Sin embargo, no deja de ser un pulsador activado por un mecanismo físico diferente. Este tipo de sensor es la versión ecológica de un interruptor de mercurio. Contiene una bola metálica en su interior que conmuta los dos pines del dispositivo de encendido a apagado, y viceversa, si el sensor llega a un cierto ángulo.

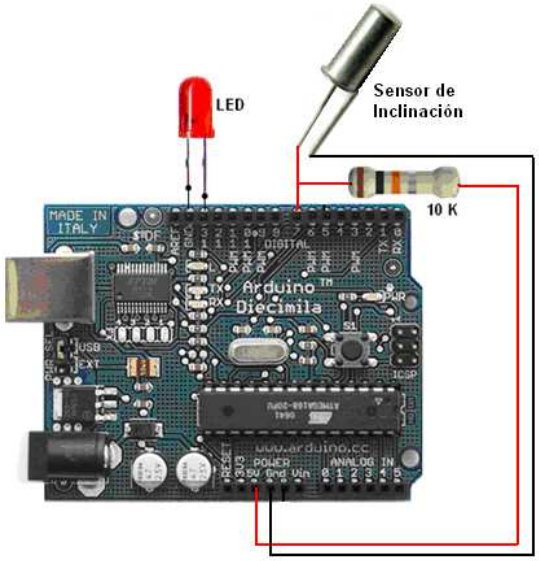


El código de ejemplo es el mismo que se empleó para el ejemplo del pulsador, pero sustituyendo este último por el sensor de inclinación. Usamos una resistencia de pull-up (de esta manera usamos la "activación a nivel bajo" para activar los pines) y conectamos el sensor al pin de entrada digital que leeremos cuando lo necesitemos.

Se ha utilizado una resistencia de 1K para montar la resistencia de pull-up y el sensor. Hemos elegido un sensor de inclinación de Assemtech.

Elementos necesarios:

- Una resistencia de 10K Ohmios.
- Un sensor de inclinación Assemtech.
- Un diodo LED.
- Cables para realizar las conexiones.



El esquema es el mismo que en el del ejemplo del pulsador

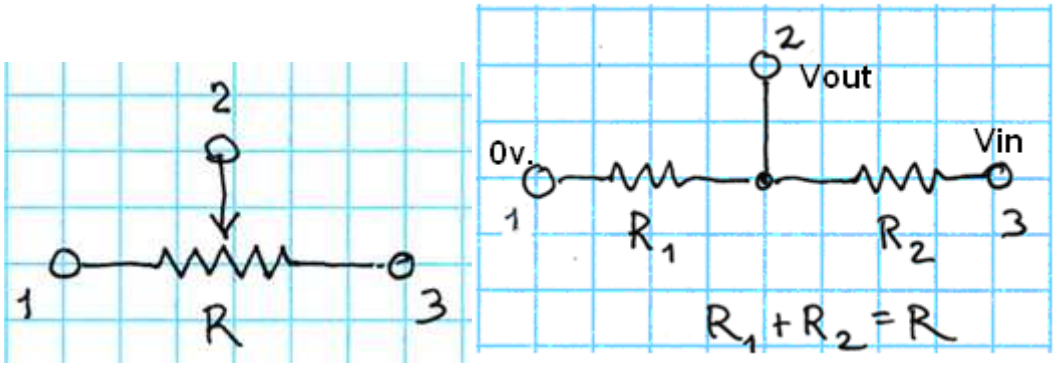
Código fuente

```
/* Sensor de Inclinación
 * -----------
 * Detecta si el sensor ha sido inclinado o no y
 * enciende la luz en consecuencia. Ten en cuenta que
 * al utilizar la "activación a nivel bajo" (mediante
 * una resistencia de pulls-up) la entrada se encuentra
 * a nivel bajo cuando el sensor se activa.
 *
 * (cleft) David Cuartielles for DojoCorp and K3
 * @author: D. Cuartielles
  */
 int ledPin = 13;          // PIN del LED
 int inPin = 7;            // 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 7 como entrada digital
}
void loop() {
  value = digitalRead(inPin);  // Lee el valor de la entrada digital
  digitalWrite(ledPin, value);

}
```

6. Potenciómetro: Lectura de señal Analógica

Descripción:

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. Como se muestra en el siguiente gráfico:



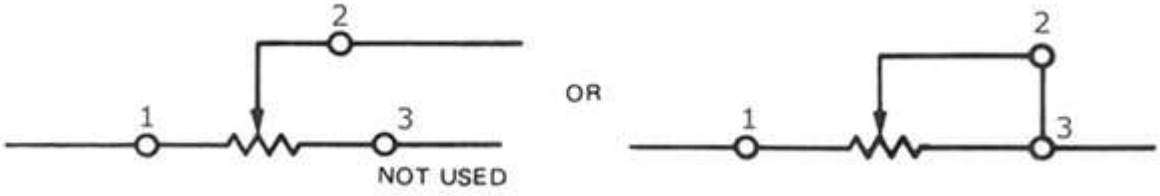
Vout=((R1/(R1+R2))*Vin (Aplicando la ley de Ohm)

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.

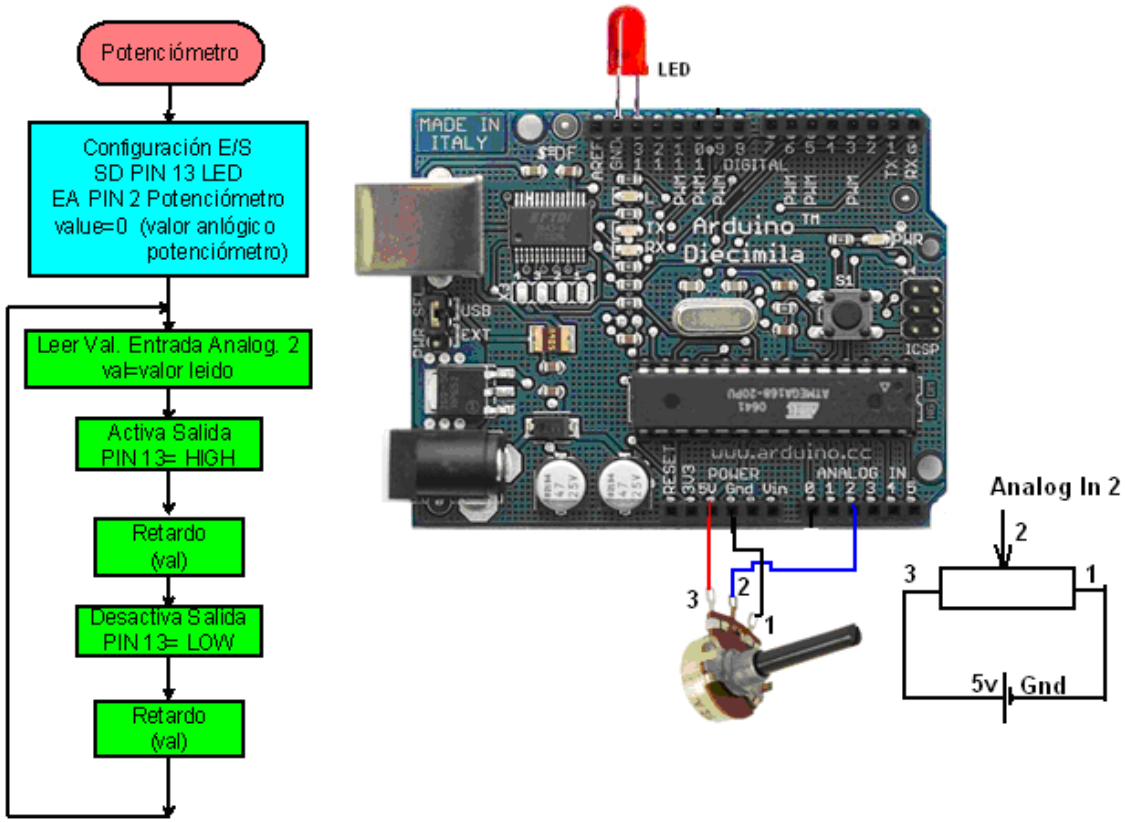
Se conectan en paralelo al circuito y se comporta como un divisor de tensión.

Un potenciómetro también puede ser usado como una resistencia variable (o reóstato) de dos terminales, en ese caso, se cortocircuitan dos de las tres patas del potenciómetro. Ejemplo:



Listado de componentes:

- 1 Potenciómetro de 10kΩ
- 1 Diodo LED
- Varios cables



Circuito:

Se conectan tres cables a la tarjeta Arduino. El primero va a tierra desde el terminal 1 del potenciómetro. El terminal 3 va a la salida de 5 voltios. El terminal 2 va desde la entrada analógica #2 hasta el terminal interno del potenciómetro.

Girando el dial o ajustando el potenciómetro, cambiamos el valor de la resistencia variable. Esto produce oscilaciones dentro del rango de 5 y 0 voltios, que son capturados por la entrada analógica.

Código:

```
/* Potenciómetro
 * ------------------
 *
 * enciende y apaga un LED conectado  al pin digital #13;
 * La cantidad de tiempo que el LED parpadeará depende del
 * valor obtenido mediante analogRead(). En este caso al pin 2 *
 *
 * Created 1 December 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
 *
 */
int potPin = 2; // seleccionar el pin de entrada analógico para el potenciómetro
int ledPin = 13; // seleccionar el pin de salida digital 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
}
void loop() {
  val = analogRead(potPin);    // lee el valor del sensor
  digitalWrite(ledPin, HIGH);  // enciende LED
  delay(val);   // detiene el programa por un tiempo "val"
  digitalWrite(ledPin, LOW);   // apaga el LED
  delay(val);          // detiene el programa por un tiempo "val"
}
```

7. El coche fantástico

Descripción del ejercicio

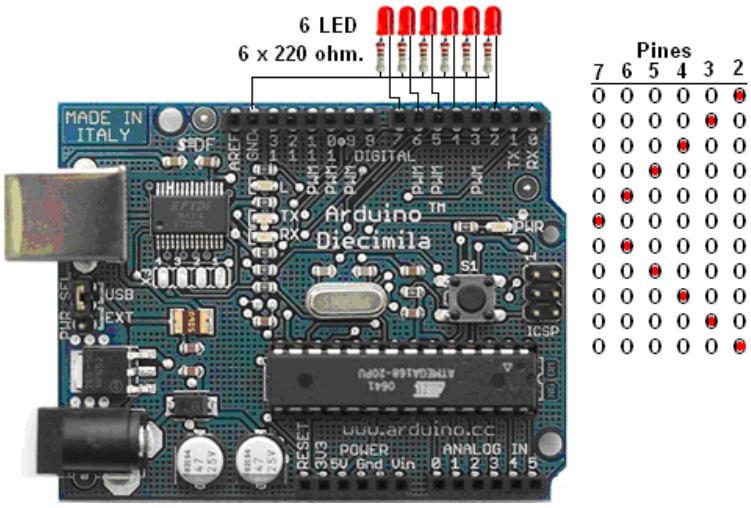
Hemos denominado este ejemplo como "El coche fantástico" en memoria de la serie de TV de los años 80 en la que el famoso David Hasselhoff tenía una máquina de IA conduciendo su Pontiac. El coche estaba equipado con gran cantidad de LED-s de todos los tamaños posibles que realizaban efectos parpadeantes.

De esta manera hemos decidido, con el objetivo de aprender programación secuencial y buenas técnicas para programar la placa E/S, sería interesante usar el coche fantástico como una metáfora.

Este ejemplo hace uso de 6 LED-s conectados a los PIN 2 a 7 de la placa mediante resistencias de 220 Ohmios. El primer código de ejemplo hace parpadear a los LED en secuencia de uno en uno, utilizando sólo las funciones di*gitalWrite(pinNum,HIGH/LOW*) y *delay(time)*. El segundo ejemplo muestra como usar una secuencia de control *for(;;;)* para hacer lo mismo, pero en menos líneas de código. El tercer y último ejemplo se centra en el efecto visual de apagar y encender los LED-s de una forma más suave.

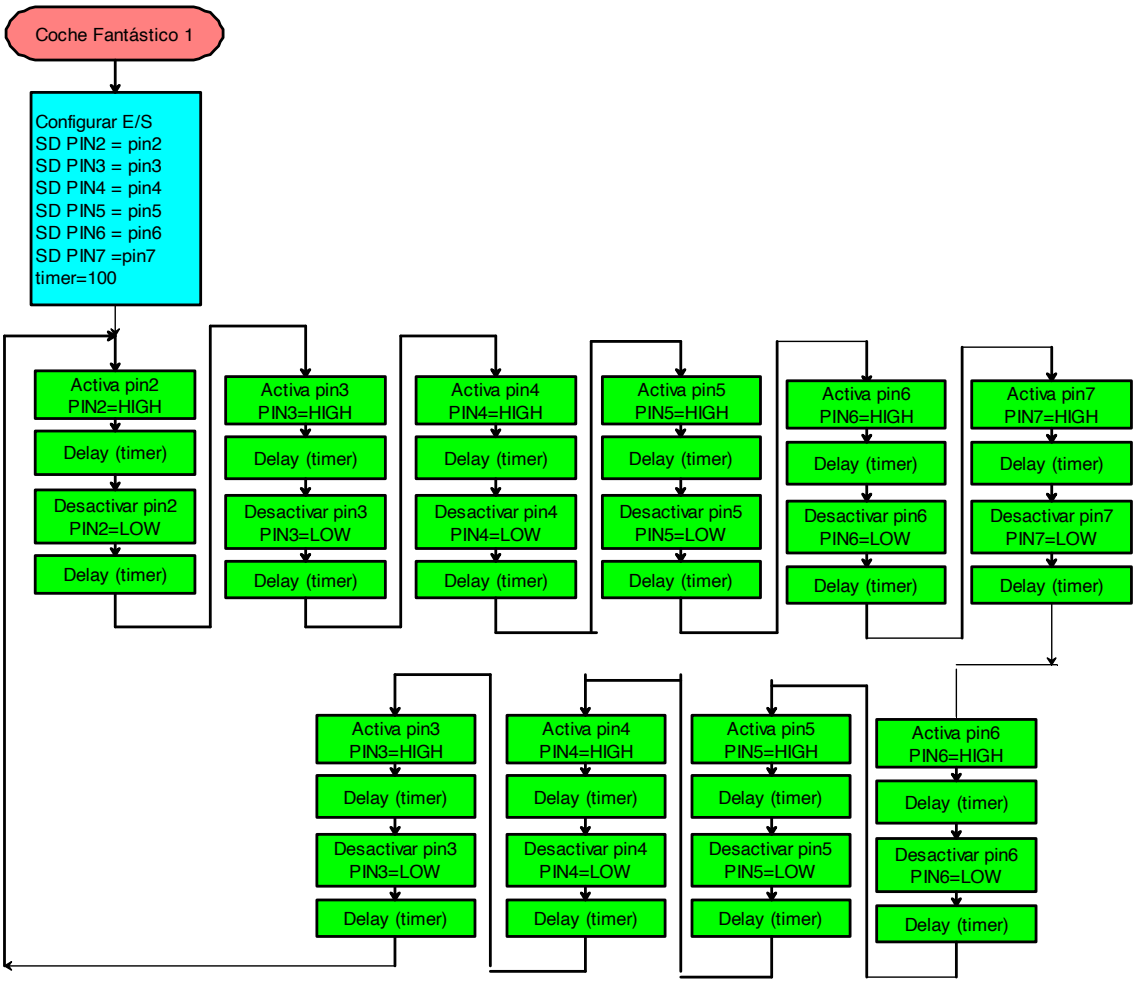
Elementos necesarios:

- 6 LED-s.
- 6 resistencias de 220 Ohmios.
- Una placa protoboard.
- Cables para realizar las conexiones



Esquema.

Ejemplo 1



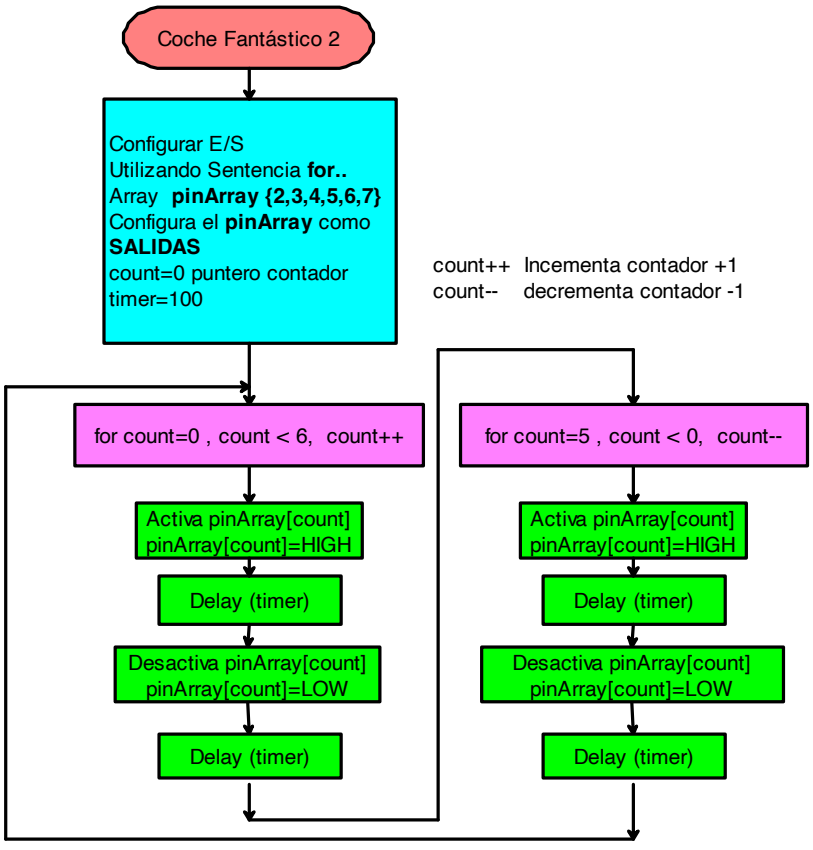
Código fuente

```
/* El coche fantástico 1

 * ---------------------
 *
 * Básicamente una extensión del LED parpadeante.
 *
 *
 * (cleft) 2005 K3, Malmo University
 * @author: David Cuartielles
 * @hardware: David Cuartielles, Aaron Hallborg
 */
int pin2 = 2;          // PIN-es de los LED
```

```
 int pin3 = 3;
 int pin4 = 4;
 int pin5 = 5;
 int pin6 = 6;
 int pin7 = 7;
 int timer = 100;        // Temporizador

void setup(){

  pinMode(pin2, OUTPUT);  // Configuración de los PIN-es como salida
  pinMode(pin3, OUTPUT);
  pinMode(pin4, OUTPUT);
  pinMode(pin5, OUTPUT);
  pinMode(pin6, OUTPUT);
  pinMode(pin7, OUTPUT);

}

void loop() {
   digitalWrite(pin2, HIGH); // Enciende y apaga secuencialmente LED-s
   delay(timer);
   digitalWrite(pin2, LOW);
   delay(timer);
   digitalWrite(pin3, HIGH);
   delay(timer);
   digitalWrite(pin3, LOW);
   delay(timer);
   digitalWrite(pin4, HIGH);
   delay(timer);
   digitalWrite(pin4, LOW);
   delay(timer);
   digitalWrite(pin5, HIGH);
   delay(timer);
   digitalWrite(pin5, LOW);
   delay(timer);
   digitalWrite(pin6, HIGH);
   delay(timer);
   digitalWrite(pin6, LOW);
   delay(timer);
   digitalWrite(pin7, HIGH);
   delay(timer);
   digitalWrite(pin7, LOW);
   delay(timer);
   digitalWrite(pin6, HIGH);
```

```
  delay(timer);
  digitalWrite(pin6, LOW);
  delay(timer);
  digitalWrite(pin5, HIGH);
  delay(timer);
  digitalWrite(pin5, LOW);
  delay(timer);
  digitalWrite(pin4, HIGH);
  delay(timer);
  digitalWrite(pin4, LOW);
  delay(timer);
  digitalWrite(pin3, HIGH);
  delay(timer);
  digitalWrite(pin3, LOW);
  delay(timer);
}
```

Ejemplo 2

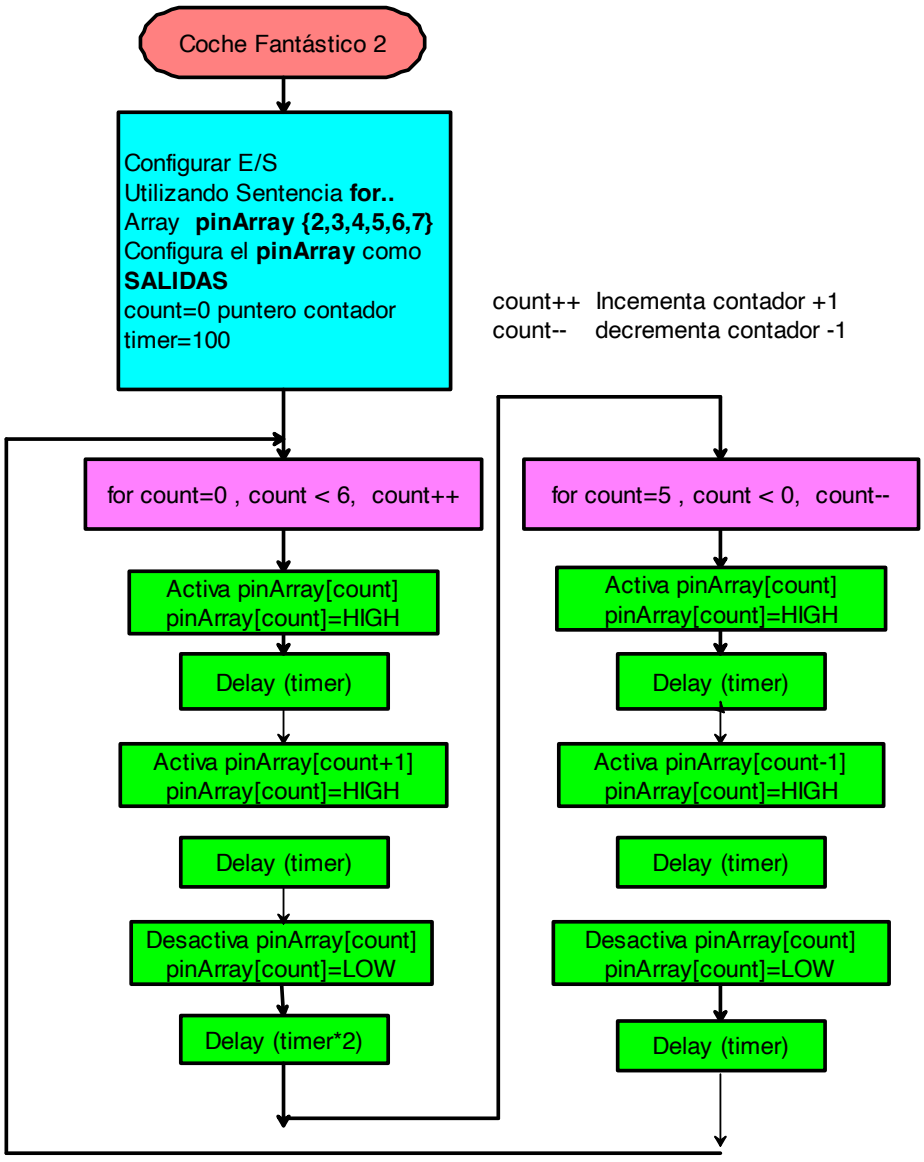


En este caso las luces se encenderán y apagaran todas en un sentido y luego , acabada la secuencia en sentido contrario.

```
/* El coche fantástico 2
 * ---------------------
 *
 * Reduciendo las líneas de código usando un for(;;).
 *
 *
 * (cleft) 2005 K3, Malmo University
 * @author: David Cuartielles
 * @hardware: David Cuartielles, Aaron Hallborg
 */
 int pinArray[] = {2, 3, 4, 5, 6, 7};    // Define el array de pines
 int count = 0;                          // Contador
 int timer = 100;                        // Temporizador
void setup(){
  for (count=0;count<6;count++){ // Configuramos todos los PIN-es
    pinMode(pinArray[count], OUTPUT);
  }
}
 void loop() {      // Enciende y apaga secuencialmente los LED-s
  for (count=0;count<6;count++) { // utilizando la secuencia de control for(;;)
   digitalWrite(pinArray[count], HIGH);  // Recorrido de ida
   delay(timer);
   digitalWrite(pinArray[count], LOW);
   delay(timer);
  }
  for (count=5;count>=0;count--) {
   digitalWrite(pinArray[count], HIGH);      // Recorrido de vuelta
   delay(timer);
   digitalWrite(pinArray[count], LOW);
   delay(timer);
  }
}
```

Ejemplo 3

En este caso el efecto que se crea es una estela visual muy vistosa.




```
/* El coche fantástico 3

 * -------------------
 *
 * Este ejemplo se centra en el efecto visual.
 *
```

```
 *
 * (cleft) 2005 K3, Malmo University
 * @author: David Cuartielles
 * @hardware: David Cuartielles, Aaron Hallborg
 */
int pinArray[] = {2, 3, 4, 5, 6, 7};          // PIN-es
int count = 0;                                // Contador
int timer = 30;                               // Temporizador

void setup(){

  for (count=0;count<6;count++) {      // Configuramos todas los PIN-es de golpe
    pinMode(pinArray[count], OUTPUT);
  }

}

void loop() {
  for (count=0;count<5;count++) {     // Enciende los LED creando una estela visual
    digitalWrite(pinArray[count], HIGH);
    delay(timer);
    digitalWrite(pinArray[count + 1], HIGH);
    delay(timer);
    digitalWrite(pinArray[count], LOW);
    delay(timer*2);
  }
  for (count=5;count>0;count--) {
    digitalWrite(pinArray[count], HIGH);
    delay(timer);
    digitalWrite(pinArray[count - 1], HIGH);
    delay(timer);
    digitalWrite(pinArray[count], LOW);
    delay(timer*2);
  }

}
```

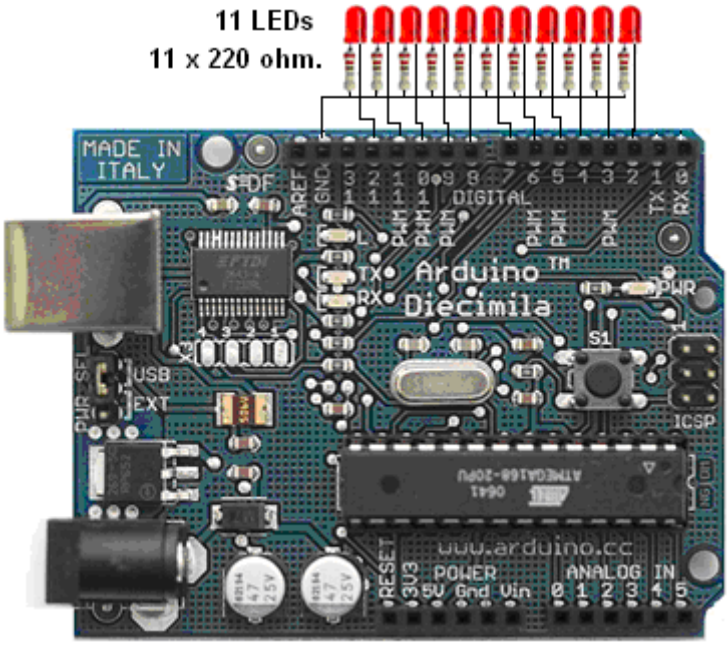
8. Estrella fugaz

Descripción del ejercicio

Este ejercicio muestra como realizar un rayo de luz, o más poéticamente, una estrella fugaz, moviéndose a través de una línea de LED-s. Podremos configurar tanto la velocidad de de la estrella, así como la longitud de la cola. No es muy elegante porque la cola brilla con la misma intensidad que la estrella, y al final, parecerá como si un rayo sólido cruzase la línea de LED-s.

Elementos necesarios:

- 11 LED-s.
- 11 resistencias de 220 Ohmios.
- Una placa protoboard.
- Cables para realizar las conexiones.



Esquema

¿Cómo funciona?

Hay que conectar 11 LED-s a los pines digitales de la placa a través de resistencias de 220 Ohmios tal y como se muestra en la imagen superior.

El programa comienza encendiendo LED-s hasta que llegue al número de LED-s establecido para la cola. En ese momento seguirá encendiendo LED-s hacia la izquierda (si se monta tal y como se muestra en la fotografía inferior), para mantener el movimiento de la estrella, al mismo tiempo que apaga LED-s por la derecha, para asegurarnos de que vemos la cola. De otra forma seguiría encendiendo LED-s hasta encenderlos todos. Esto ocurre cuando el tamaño de la cola es igual o mayor que el número de LED-s.

El tamaño de la cola debería ser relativamente pequeño en comparación con el número de LED-s de forma que podamos ver la estrella.

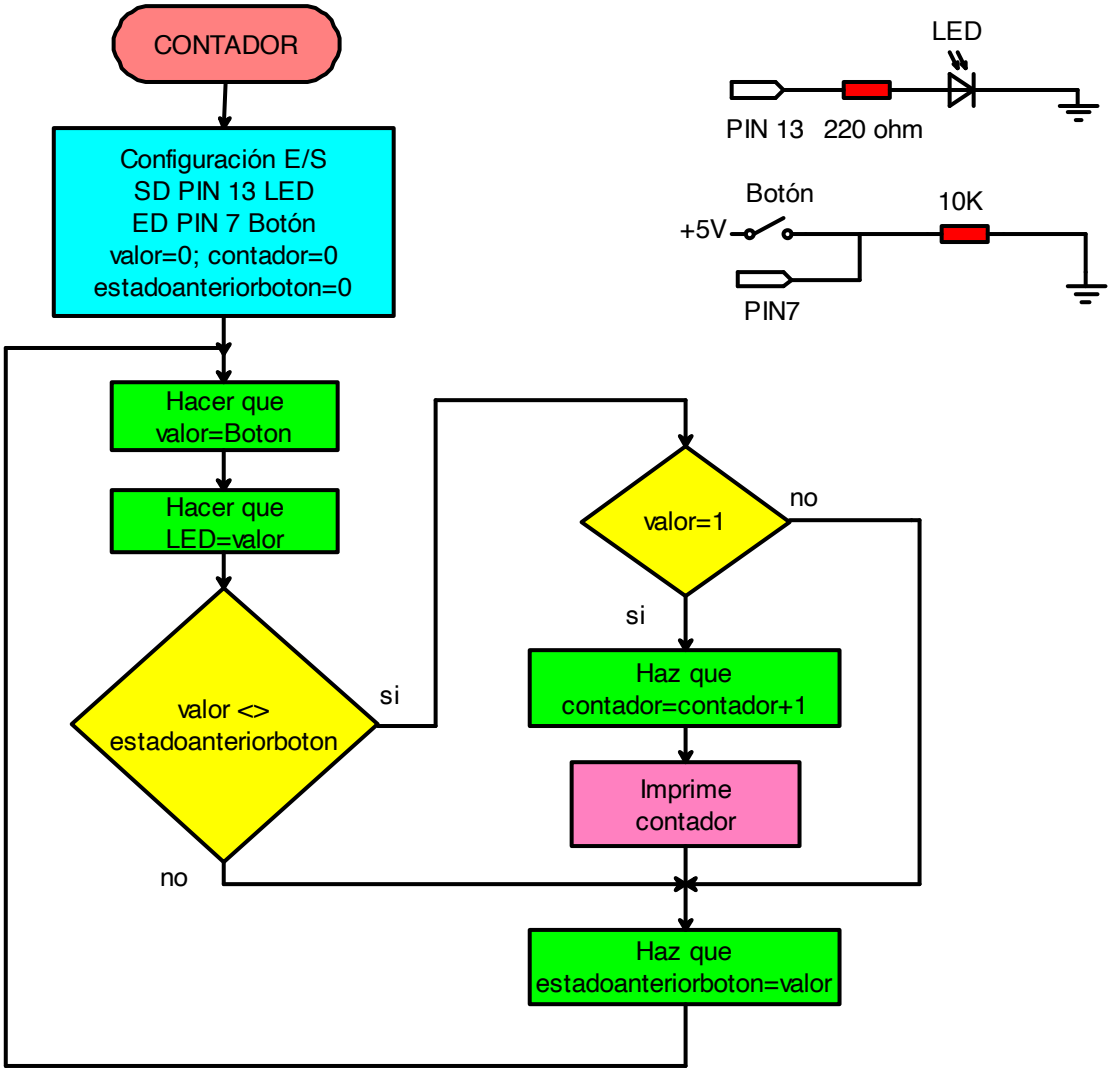
Código fuente

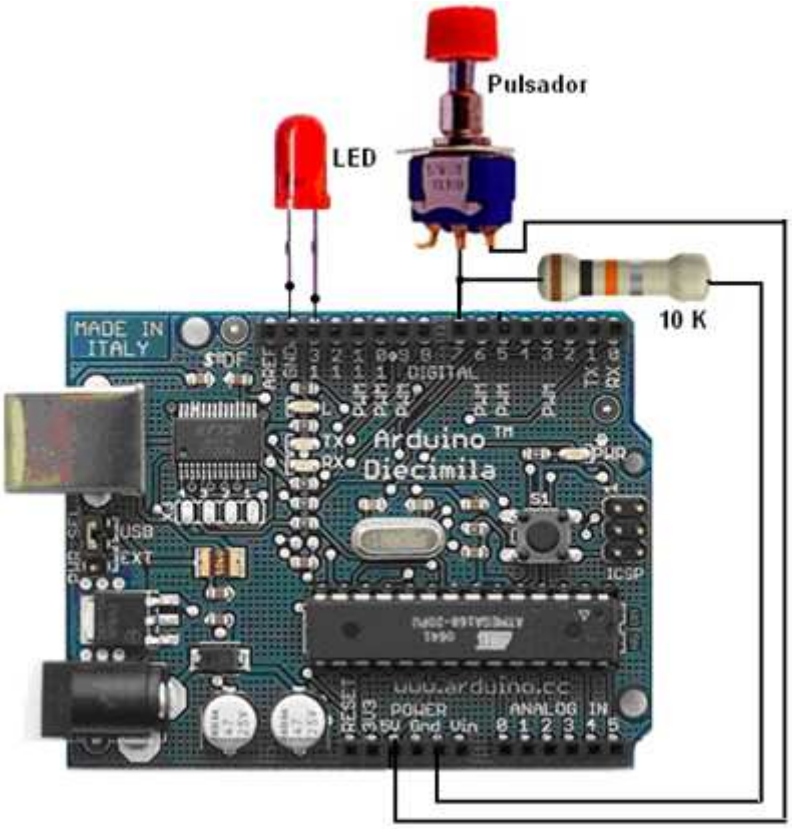
```
/* Estrella fugaz
 *-------------
 * Este programa es una variante del ejemplo del coche  fantástico. Muestra mediante
 * un loop una estrella fugaz que es dibujada en una línea de LED-s
 * directamente conectados a la placa Arduino. Puedes controlar la velocidad a la que
 * la estrella se mueve gracias a una variable llamada "waitNextLed".  También
 * puedes controlar la longitud de la cola de la estrella a través de la variable "tail
 * length"
 * @author: Cristina Hoffmann
 * @hardware: Cristina Hofmann
 */
 // Variable declaración
 // Declaración de los PIN-es mediante un array
 int pinArray [] = { 2,3,4,5,6,7,8,9,10,11,12 };
 int controlLed = 13;          // LED de control
 int waitNextLed = 100;      // Tiempo antes de encender el siguiente LED
 // Número de LED-s que permanecen encendidos antes de empezar a apagarlos para
 //formar la cola
 int tailLength = 4;
  // Número de LED-s conectados (que es también el tamaño del array)
 int lineSize = 11;

 void setup()          // Configuración de los PIN-es como salida digital
 {
   int i;
   pinMode (controlLed, OUTPUT);
   for (i=0; i< lineSize; i++)
   {
     pinMode(pinArray[i], OUTPUT);
   }
 }
```

```
 void loop()
 {
  int i;
 // Se establece la longitud de la cola en un contador
  int tailCounter = tailLength;
 // Se enciende el LED de control para indicar el inicio del loop
  digitalWrite(controlLed, HIGH);
   for (i=0; i<lineSize; i++)
 {
  digitalWrite(pinArray[i],HIGH);    // Se encienden consecutivamente los LED
 // Esta variable de tiempo controla la velocidad a la que se mueve la estrella
  delay(waitNextLed);
  if (tailCounter == 0)
   {
 // Se apagan los LED-s en función de la longitud de la cola.
 digitalWrite(pinArray[i-tailLength],LOW);
   }
  else
    if (tailCounter > 0)
      tailCounter--;
 }
  for (i=(lineSize-tailLength); i<lineSize; i++)
 {
  digitalWrite(pinArray[i],LOW);              // Se apagan los LED
// Esta variable de tiempo controla la velocidad a la que se mueve la estrella
  delay(waitNextLed);
 }
}
```

9. Contador

Se trata de contar las veces que se pulsa un botón conectado en la entrada 7 de Arduino a la vez que cada vez que contamos encendemos el led conectado en la salida 13. El valor de la variable que almacena el número de impulsos generados se envía al PC para que se visualice en la pantalla.



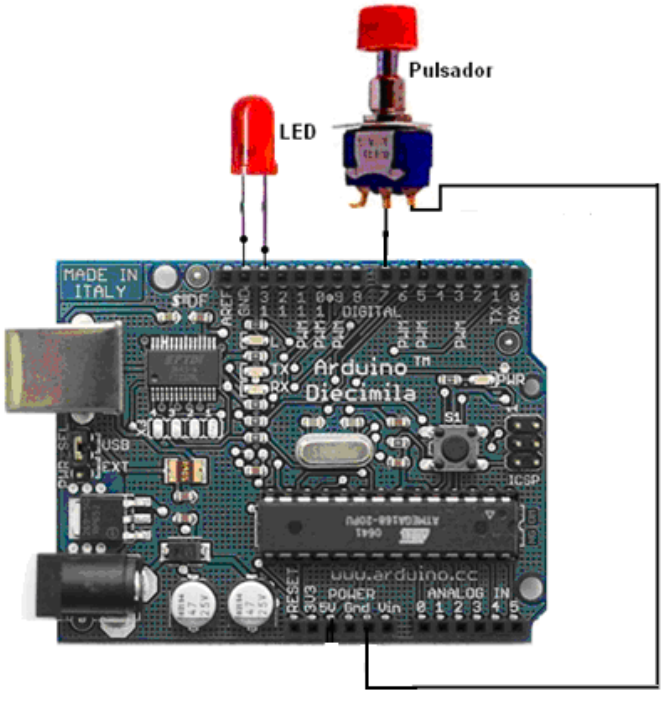


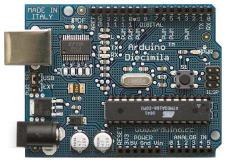
Esquema

```
/* Programa  Contador
 * -----------
 *
 * Detecta si el botón conectado a la entrada 7 ha sido presionado y enciende el LED
 * Envía al PC el valor de la variable de cuenta ""Contador" vía puerto serie.
 *
 * Christian Nold & Erica Calogero
 *
 */

int LED = 13;
int Boton = 7;
int valor = 0;
int contador = 0;
int estadoanteriorboton = 0;

void setup()
{
  beginSerial(9600);              // Configura velocidad de transmisión a 9600
```

```
  pinMode(LED, OUTPUT);   // inicializa como salida digital el pin 13
  pinMode(Boton, INPUT);     // inicializa como entrada digital el 7
}

void loop()
{
  valor = digitalRead(Boton);   // lee el valor de la entrada digital pin 7
  digitalWrite(LED, valor);
   if(valor != estadoanteriorboton){
   if(valor == 1){
   contador++;
   printInteger(contador);
   serialWrite(10);
   serialWrite(13);
 }
}
   estadoanteriorboton = valor;
}
```

Podríamos prescindir de la resistencia colocada con el pulsador si habilitásemos la resitencia interna Pull-up de la entrada PIN7 en ese caso el circuito quedaría como el siguiente:



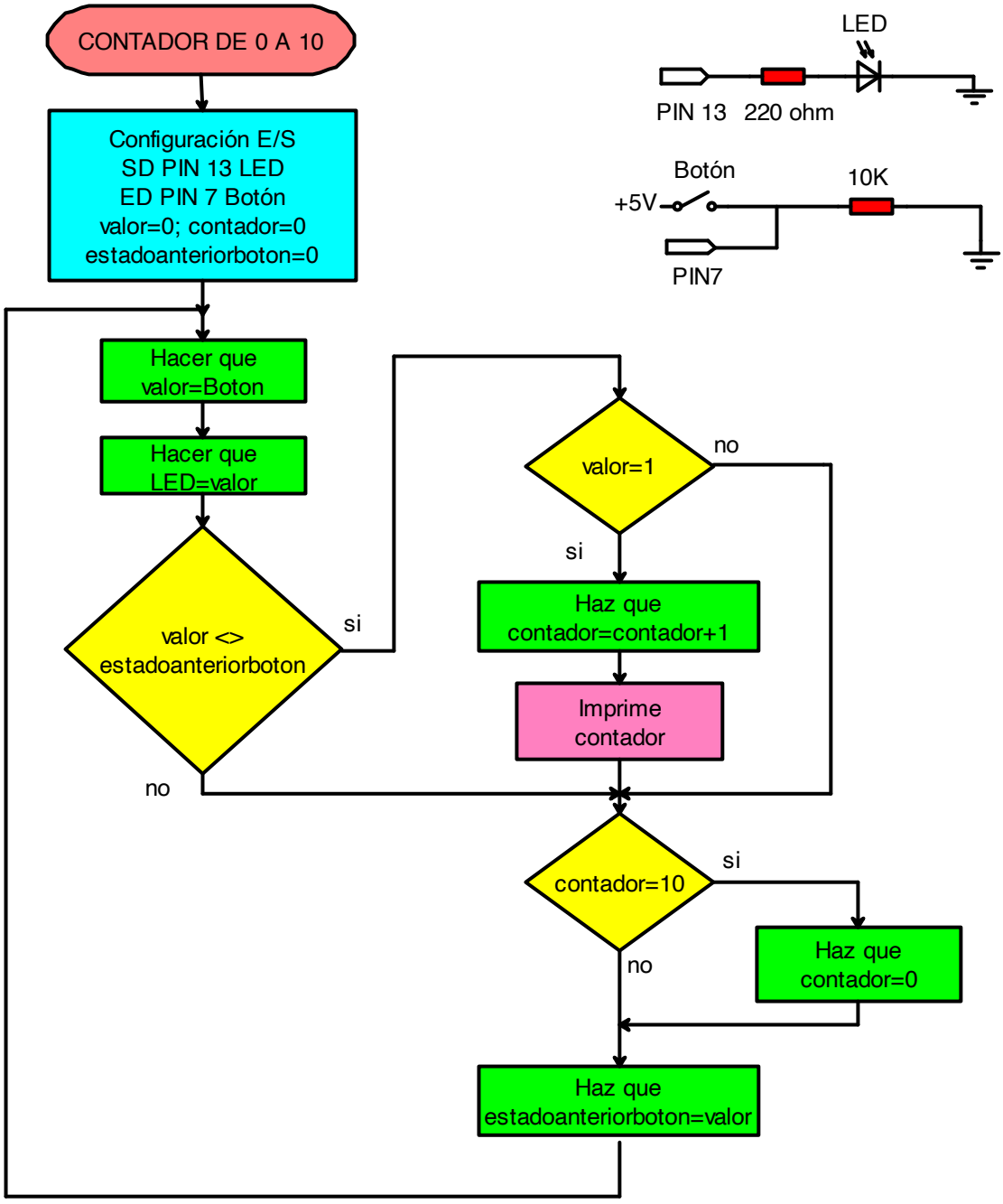


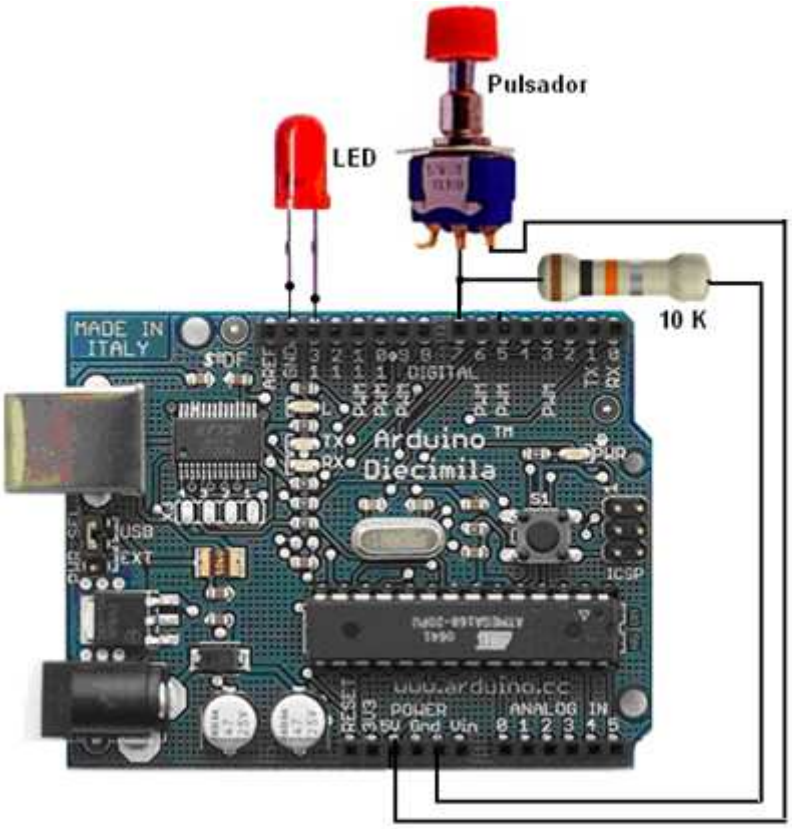
El programa en este caso seria muy parecido al anterior. Observese que ahora al pulsar el boton introducimos un “=” en el PIN7 por lo tanto si quiero que se encienda la salida PIN13 debo escribir en ella el valor leido del pulsador negado es decir “!valor”

```
/* Programa  Contador Modificado
 * -----------
 *
 * Detecta si el botón conectado a la entrada 7 ha sido presionado y enciende el LED
 * Envía al PC el valor de la variable de cuenta ""Contador" vía puerto serie.
 *
 * Christian Nold & Erica Calogero J.M. Ruiz
 *
 */

int LED = 13;
int Boton = 7;
int valor = 0;
int contador = 0;
int estadoanteriorboton = 0;

void setup()
{
  beginSerial(9600);                 // Configura velocidad de transmisión a 9600
  pinMode(LED, OUTPUT);   // inicializa como salida digital el pin 13
  pinMode(Boton, INPUT);   // inicializa como entrada digital el 7
  digitalWrite(Boton,HIGH);  // Habilitamos la resitencia interna Pull-up del PIN7
}

void loop()
{
  valor = digitalRead(Boton);   // lee el valor de la entrada digital pin 7
  digitalWrite(LED, !valor);   // Escribimos en la salida el bvlaor leido negado
   if(valor != estadoanteriorboton){
   if(valor == 1){
   contador++;
   printInteger(contador);
   serialWrite(10);
   serialWrite(13);
 }
}
   estadoanteriorboton = valor;
}
```

Contador de 0 a 10

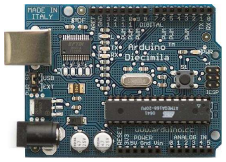
Se trata de realizar una variación sobre el ejemplo anterior de tal manera que cuando el valor del contador llegue a 10 se ponga a cero y comience de nuevo.





Esquema

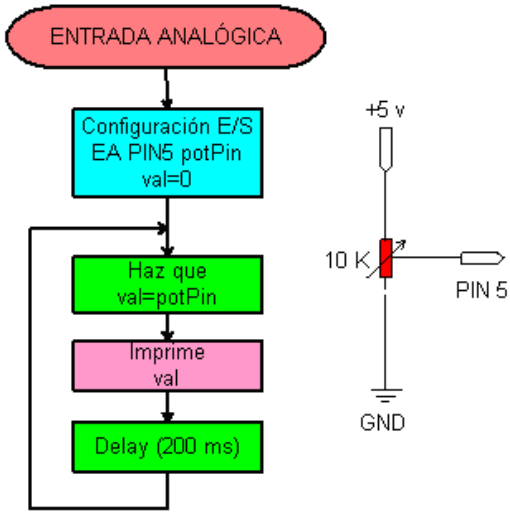
```
/* Programa Contador de 0 a 10
 * ----------
 *
 * Detecta si el botón conectado a la entrada 7 ha sido presionado y enciende el LED
 * Envía al PC el valor de la variable de cuenta ""Contador" vía puerto serie.
 *
 * Christian Nold & Erica Calogero
 *
 */

int LED = 13;
int Boton = 7;
int valor = 0;
int contador = 0;
int estadoanteriorboton = 0;

void setup()
{
  beginSerial(9600);              // Configura velocidad de transmisión a 9600
```

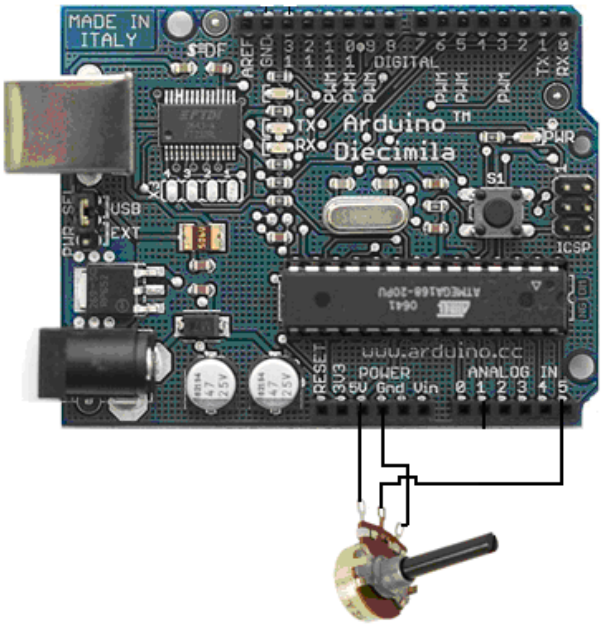



```
  pinMode(LED, OUTPUT);   // inicializa como salida digital el pin 13
  pinMode(Boton, INPUT);     // inicializa como entrada digital el 7
}

void loop()
{
  valor = digitalRead(Boton);   // lee el valor de la entrad digital pin 7
  digitalWrite(LED, valor);

 if(valor != estadoanteriorboton){
   if(valor == 1){

    contador++;
    printInteger(contador);
    serialWrite(10);
    serialWrite(13);
    if (contador==10) {            // Limita la cuenta al valor 10
       contador=0;
       }
  }
}
estadoanteriorboton = valor;
}
```

10. Entrada Analógica

Se trata de configurar un canal de entrad analógico pin 5 y enviar el valor leído al PC para visualizarlo





Organigrama Esquema

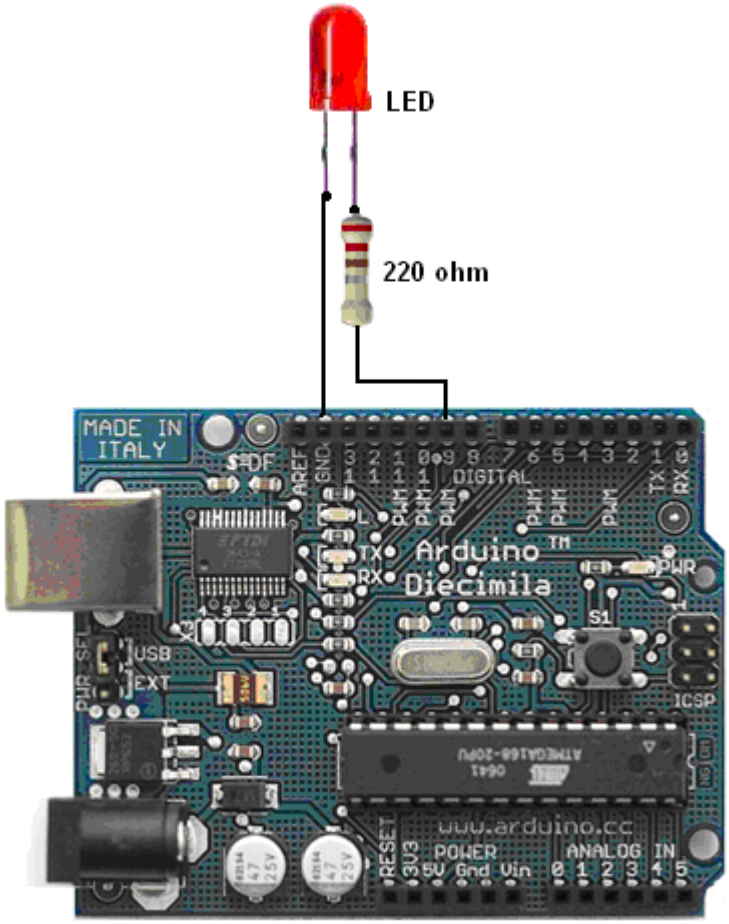
Programa

```
/* Entrada Analógica  */

int potPin = 5;    // selecciona el pin de entrada para colocar el potenciómetro
int val = 0;       // variable para almacenar el valor leído por la entrada analógica

void setup() {
  beginSerial(9600);
}

void loop() {

  val = analogRead(potPin);    // lee el valor del canal de ENTRADA analógica
  printInteger(val); // Envía al PC el valor analógico leído y lo muestra en pantalla
  serialWrite(10);
  delay(100);
}
```

11. Simulación de la luz de una vela

De trata de simular el movimiento de la llama de una vela. Hacemos uso de la instrucción para generar un numero aleatorio que lo asignamos a una salida analógica PWM y otro numero que lo asociamos a la variable de temporización (tiempo que esperamos para cambiar el valor de la salida).

Esquema



Programa

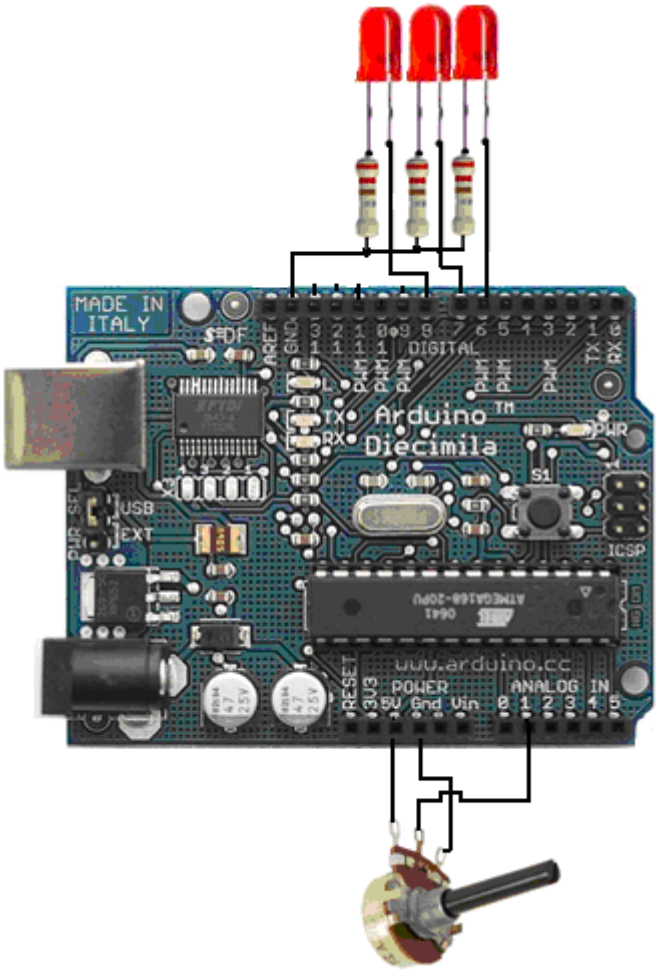
```
/*
 * Simula luz de vela
 * Saca por una de las salidas del puerto PWM un valor aleatorio que activa un LED
 *
 * 2007 Tod E. Kurt <tod@todbot.com>
 * http://todbot.com/
```

```
 *
 */

int ledPin = 9;    // selecciona el puerto PWM
int val = 0;       // define y pone a cero la variable "brillo"
int delayval = 0;  // define el intervalo de cambio de valor de salida

void setup() {
  randomSeed(0);           // inicializa el generador de números aleatorios
  pinMode(ledPin, OUTPUT);  // declara el pin de SALIDA pin 9
}

void loop() {
  val = random(100,255);       // genera un número aleatorio entre 100 y 255 que asigna
a la variable val
  analogWrite(ledPin, val);    // envía ese valor a la salida pin 9

  delayval = random(50,150);   // genera un numero aleatorio entre 30 y 100 y lo
asigna a la variable de temporización
  delay(delayval);             // espera un tiempo delayval medido en milisegundos
}
```

12. Construcción de un indicador de nivel (vúmetro con diodos led)

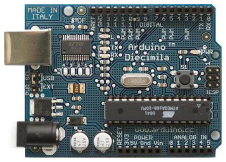
Se trata de construir u indicador de nivel que sea capaz de medir el valor de una señal de entrada generada por un potenciómetro desde una entrada analógica.

Se establecerán 3 diodos Led conectados a las salidas PIN6, PIN7 y PIN8. La entrada la conectaremos en la entrada analógica PIN 1 (analog IN1)



Programa para Arduino.

```
int ledPin1 = 8; // Selección de PIN para cada LED
int ledPin2 = 7;
int ledPin3 = 6;
int inPin= 1; // selecciona la entrada analógica 1 (potenciómetro)
```

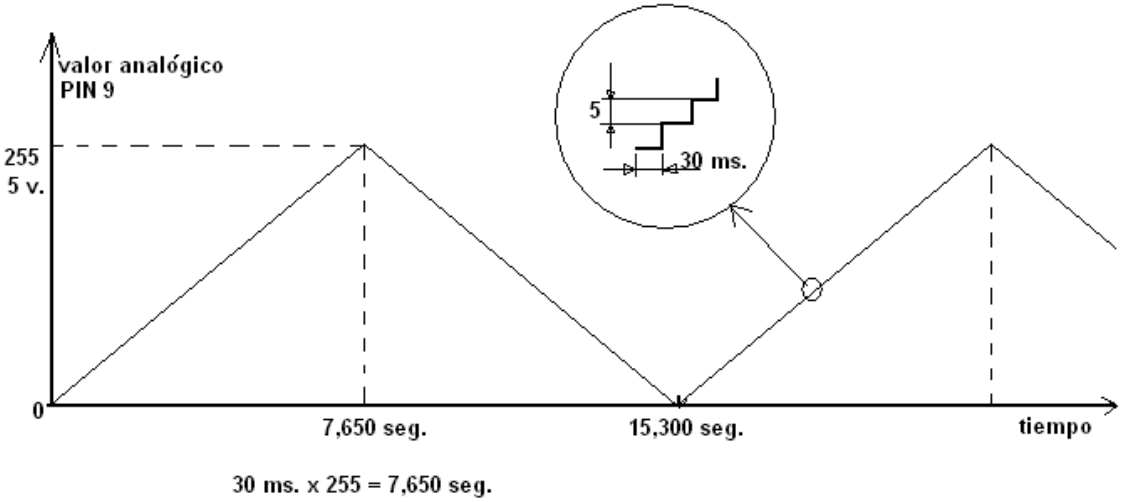



```
void turn_off() { //Apaga los 3 LEDS
digitalWrite(ledPin1, LOW);
digitalWrite(ledPin2, LOW);
digitalWrite(ledPin3, LOW);
}
void setup() {
pinMode(ledPin1, OUTPUT); // declara LEDs como salidas
pinMode(ledPin2, OUTPUT);
pinMode(ledPin3, OUTPUT);
turn_off();  //
}
void loop(){
int val;
val= analogRead(inPin); // lee el valor de la señal analógica
turn_off();apaga los tres LED
// Si el valor de la señal medida es > 256 enciende LED del PIN8
if (val>= 256) digitalWrite(ledPin1, HIGH);
// Si el valor de la señal medida es > 512 enciende LED del PIN7
if (val>= 512) digitalWrite(ledPin2, HIGH);
// Si el valor de la señal medida es > 758 enciende LED del PIN6
if (val>= 768) digitalWrite(ledPin3, HIGH);
}
```

13. Encendido y apagado de una luz de manera analógica

Se trata de que enviemos hacia la salida 9 un valor analógico ascendente y descendente cíclicamente comprendido entre 0 y 255 en incrementos de 5.

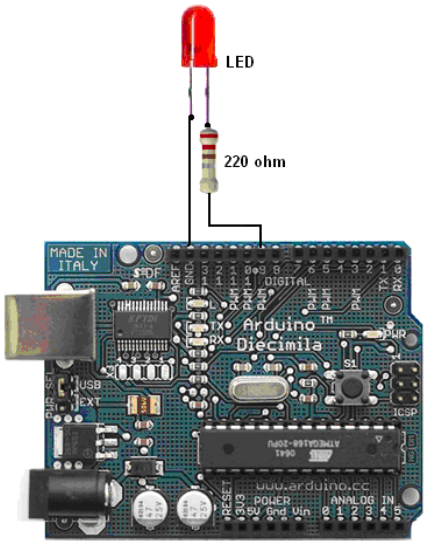
Para la realización de este ejercicio se debe empleará una estructura de programación tipo **for** que realice el incremento o decremento de una variable entre 0-255 y 255-0 con un valor te retardo entre cambio de valor de 30 mseg.



30 ms. x 255 = 7,650 seg.

En la figura vemos una repr4sentacion e l valor de la señal de salida en el PIN 9. Téngase en cuenta que el valor 255 equivale a 5 voltios y el valor 0 a 0 voltios.

Esquema



Programa

```
// Salida Analógica Cíclica
// by BARRAGAN <http://people.interaction-ivrea.it/h.barragan>

int value = 0;                  // Valor a sacar por la salida analógica PIN 9
int ledpin = 9;                 // Salida analógicas PIN 9

void setup()
{
  // nothing for setup
}

void loop()
{
        for(value = 0 ; value <= 255; value+=5) // Variación de la variable se salida
ente el MIN yMAX
  {
        analogWrite(ledpin, value);          // Enviar valor a la salida (entre 0 y 255)
        delay(30);                      // Esperar 30 ms para ver el efecto de variación
  }
        for(value = 255; value >=0; value-=5)   // Variación de la variable de salida
entre MAX y MIN
  {
    analogWrite(ledpin, value);
    delay(30);
  }
}
```

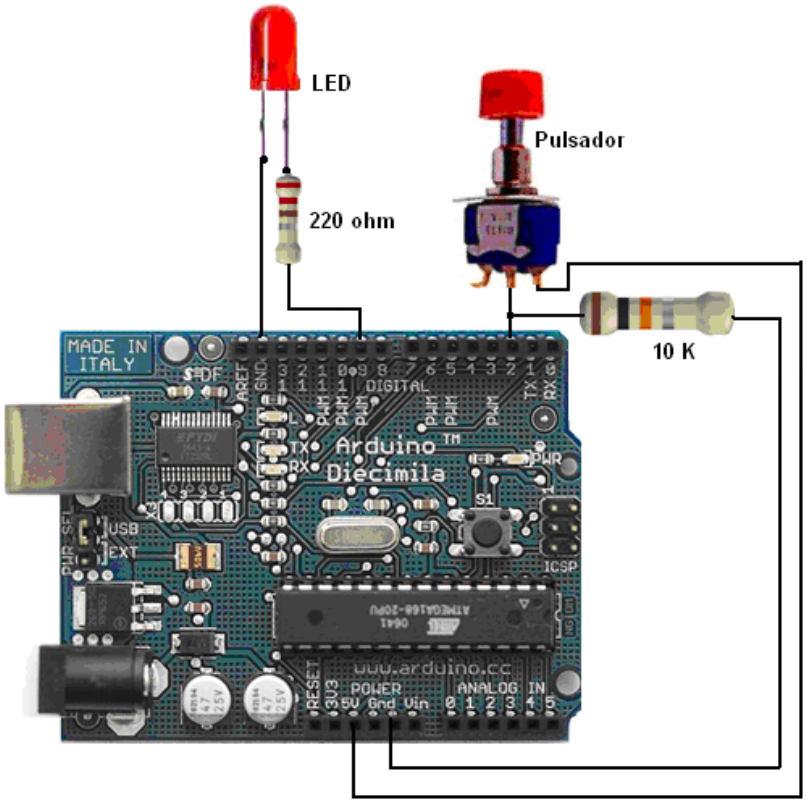
14. Control de la iluminación de una lámpara.

Con esta aplicación se pretende controlar el grado de iluminación de una lámpara (simulada con un LED) mediante un pulsador.

Funcionamiento:

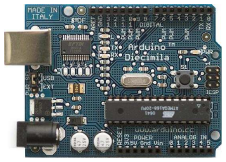
Si no pulsamos el pulsador (entrada 0) la lámpara incrementar y decrementar su brillo o nivel de iluminación.
Si pulsamos (entrada 1) la lámpara se encenderá y apagará con una cadencia de 50 mseg.

Esquema



Programa

```
/* Control de iluminación mediante Pulsador
 */

int ledPin = 9;              // Selección del PIN de salida Analógica
int inputPin = 2;            // Selección del PIN para la entrada de pulsador
int val = 0;                 // variable para leer el estado del pulsador
```

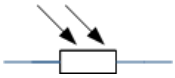



```
int fadeval = 0;

void setup() {
  pinMode(ledPin, OUTPUT);      // designación de salida Analógica
  pinMode(inputPin, INPUT);     // designación de pulsador de entrada
}

void loop(){
  val = digitalRead(inputPin);  // leer valor de entrada
  if (val == HIGH) {            // Botón pulsado
    digitalWrite(ledPin, LOW);  // puesta a "0" de la salida
    delay(50);
    digitalWrite(ledPin, HIGH); // puesta a "1" de la salida
    delay(50);
  }
  else {  // Si se presiona el boton
    for(fadeval = 0 ; fadeval <= 255; fadeval+=5) { // valor de salida analógica
asciende de min a max)
      analogWrite(ledPin, fadeval);          // fija el valor en la salida ( desde 0-255)
      delay(100);
    }
    for(fadeval = 255; fadeval >=0; fadeval-=5) { // valor de salida analógica desciende
(desde max to min)
      analogWrite(ledPin, fadeval);
      delay(100);

    }

  }
 }
```

15. Sensor de Luz o LDR (Light Dependent Resistor):

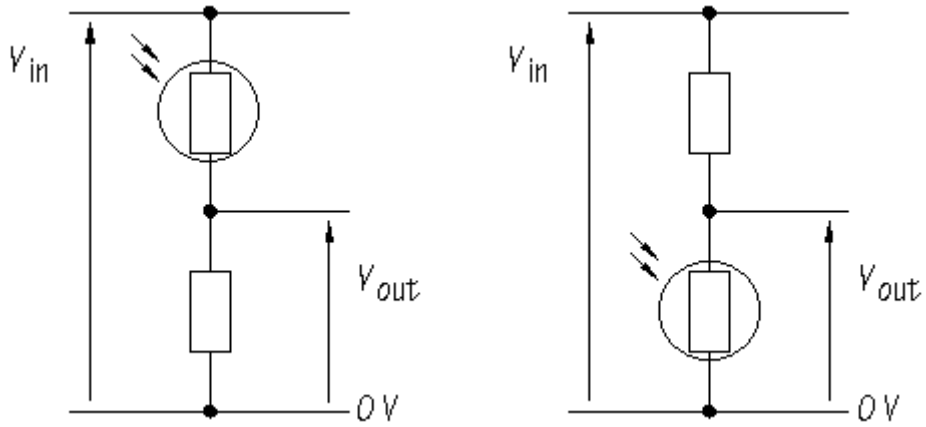
Descripción:

Un LDR es una resistencia variable, que varia su valor dependiendo de la cantidad de luz que incide sobre su superficie. Cuanta mas intensidad de luz incide en la superficie de la LDR menor será su resistencia y cuanto menos luz incide mayor será la resistencia. Suelen ser utilizados como sensores de luz ambiental o como una fotocélula que activa un determinado proceso en ausencia o presencia de luz.



Un sensor de luz se compone de una LDR como parte de un divisor de tensión resistivo.

Ejemplo:



Vout=((Rbotton/(Rbottom+Rtop))*Vin

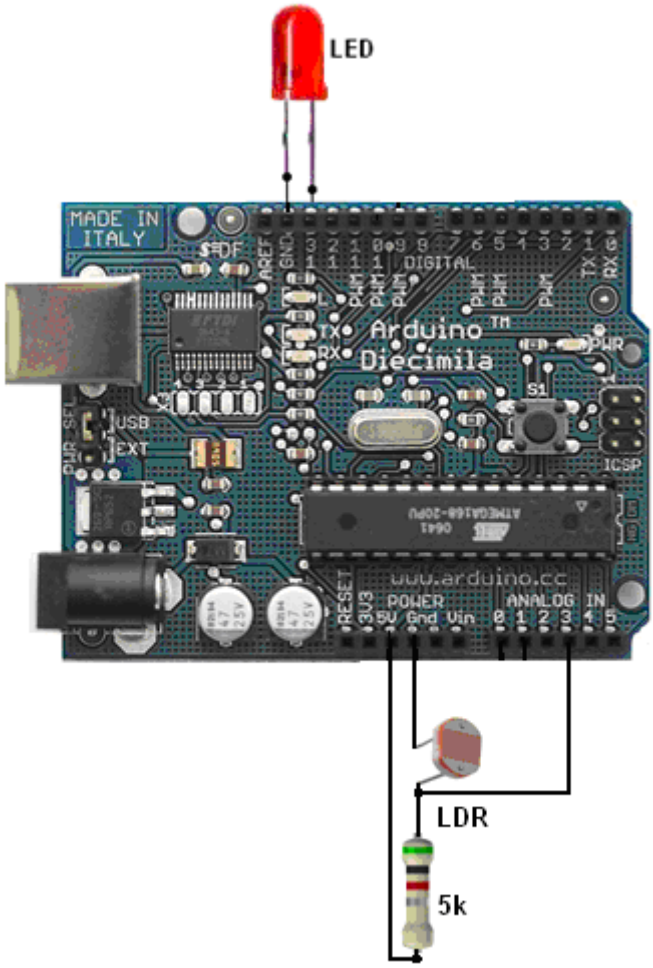
Si la LDR es usada como R_{top}, como en el primer circuito, da tensión alta (HIGH) en la salida cuando la LDR está en la luz, y una tensión baja (LOW) en la salida cuando la LDR está en la sombra.

La acción del divisor de tensión es inversa cuando la LDR es usada como R_{bottom} en lugar de R_{top}, como en el segundo circuito. El circuito da tensión Baja (LOW) en la salida cuando la LDR está en la luz, y una tensión alta (HIGH) en la salida cuando la LDR está en la sombra. El circuito divisor de tensión dará una tensión de la salida que cambia con la iluminación, de forma inversamente proporcional a la cantidad de luz que reciba (sensor de oscuridad).

Listado de componentes:

- 1 LDR
- 1 Resistencia de 5kΩ
- Un par de cables
- 1 Diodo LED

El esquema del circuito puede quedar de la siguiente forma:



El circuito consta de un divisor de tensión formado por la LDR y la resistencia de 5kΩ.

Código:

```
/* Sensor de Luz LDR
 * ------------------
 *
 * Enciende y apaga un LED (light emitting diode) conectado al pin digital #13.
```

```
 * La cantidad de tiempo que el LED estará encendido y apagado depende del
 * valor obtenido de analogRead().
 * La salida del sensor de luz o divisor de tensión, está conectado
 * a la entrada del pin 3, por lo que
 * a más luz, el parpadeo del LED será menor y a menos luz el parpadeo del LED
mayor.
 * (sensor de oscuridad)
 *
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
 *
 */
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
}
void loop() {
  val = analogRead(LightPin);    //lee el valor del 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
}
```

16. Sensor de temperatura o NTC

Descripción

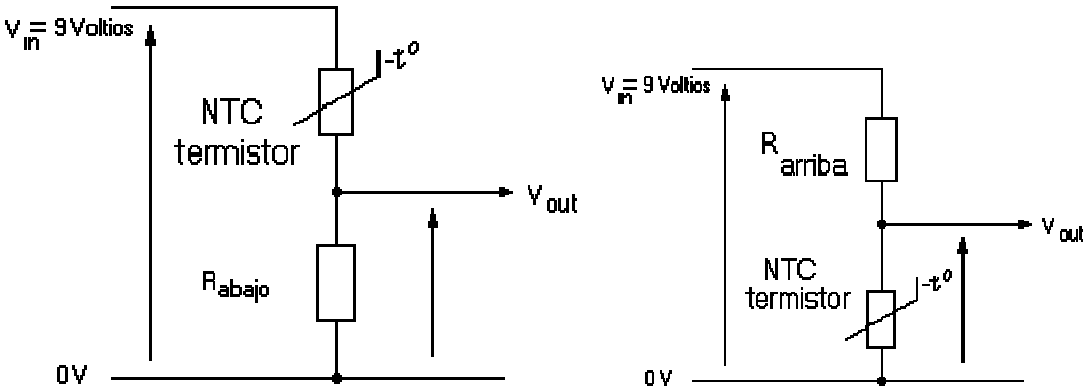
En este ejemplo se trata de medir la temperatura desde el PIN3 de entrada analógica y ver si este valor supera un valor dado de 500 (medida absoluta) si supera este valor activará la salida digital PIN13 y si no la apagara. Además queremos que se muestre en el monitor de salida del IDE Arduino el valor leído. D sensor utilizaremos un sensor del tipo NTC.

Un NTC o termistor NTC es una resistencia variable, que varia su valor dependiendo de la temperatura ambiente. Cuanta mas temperatura menor será su resistencia y cuanto menos temperatura mayor será la resistencia. Suelen ser utilizados en alarmas.

Un sensor de temperatura se compone de un NTC como parte de un divisor de tensión resistivo.

Ejemplo

Como alarma de incendio o sensor de calor, utilizaremos un circuito que entregue una tensión alta cuando se detecten las condiciones de temperatura caliente. Necesitamos poner un divisor de tensión con un termistor NTC en la posición que ocupa R_{arriba}:



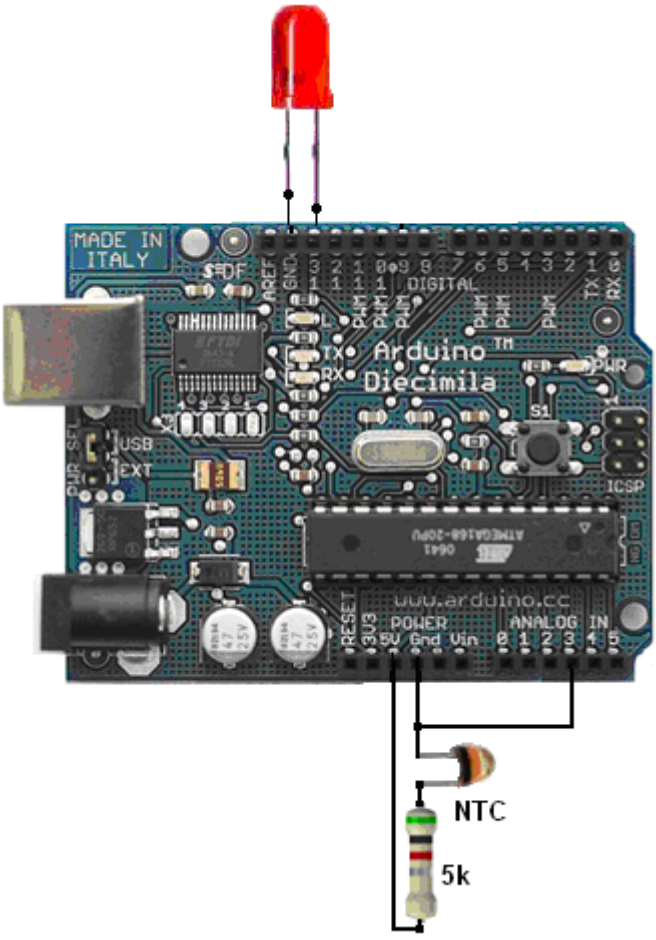
Como alarma de frío o sensor de frío, usaremos un circuito que dé una tensión alta en condiciones frías. Necesitamos un divisor de voltaje con el termistor NTC en lugar de R_{bajo}:

Listado de componentes:

- 1 NTC sensor de temperatura
- 1 Resistencia 1kΩ
- 1 Diodo LED
- Un par de cables

Esquema del Circuito:

El circuito consta de un divisor de tensión formado por el NTC y la resistencia de 1kΩ.



Código:

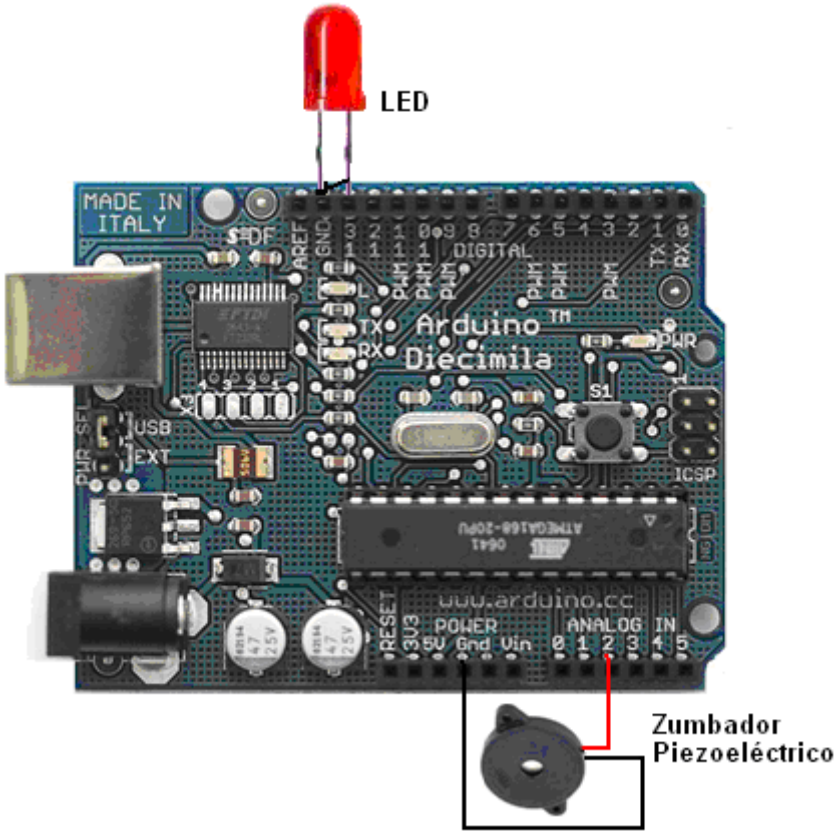
```
//Detector de valor de temperatura

 int led=13;
 int ntc=3;
int medida=0;
```

```
//variable que guarda el límite de temperatura al que se activa el ventilador
 int nivel=500;
 void setup(){
   pinMode(led,OUTPUT);
   pinMode(motor,OUTPUT);
   beginSerial(9600);
 }
//procedimiento que envía al puerto serie, para ser leído en el monitor,
 void monitoriza(){
   printInteger(medida);    //el valor de la señal de la NTC en la entrada analógica
   printString("     ");
   delay(100);              //para evitar saturar el puerto serie
 }
 void loop(){
   medida=analogRead(ntc);
   monitoriza();
   if(medida>nivel){          //si la señal del sensor supera el nivel marcado:
     digitalWrite(led,HIGH);    //se enciende un aviso luminoso

   }
   else{           // si la señal está por debajo del nivel marcado
     digitalWrite(led,LOW);
   }
 }
```

17. Sensor de Fuerza.

Se trata de convertir un zumbador piezoeléctrico en un sensor de presión o fuerza utilizando este como sensor de entrada en uno de los pines de entrada analógica de Arduino (PIN 2)

Esquema



Programa

```
/* Sensor piezoelectrico
 * ----------
 * Convertir un zumbador piezoelectrico en un sensor de de fuerza
 *
 * Created 24 October 2006
 * copyleft 2006 Tod E. Kurt <tod@todbot.com
 * http://todbot.com/
 */
int ledPin = 13;
```

```
int piezoPin = 2;

int THRESHOLD = 1;  // Configura valor mínimo para que se encienda la salida
PIN13

int val = 0;       // variable que almacena el valor leído por el sensor
int t = 0;         // valor del intervalo de medida

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(19200);
  Serial.println("ready");      // indicador de espera
}

void loop() {
  digitalWrite(ledPin,LOW);     // indicador de reposo (esperando)

  val = analogRead(piezoPin);   // lectura de valor del piezoeléctrico
  if( val > THRESHOLD ) {      // si el valor leído es superior al mínimo establecido
    digitalWrite(ledPin, HIGH); // activa salida 13
    t = 0;
    while(analogRead(piezoPin) > THRESHOLD) {
      t++;
    } // wait for it to go LOW  (espera con una pequeña histéresis)
    if(t>100) {  // escribe en el puerto
      Serial.print("knock! ");
      //Serial.println(t);
    }
  }
}
```

18. Generador de notas musicales

Se trata de generar hasta 8 notas musicales por una de las salidas analógicas de Arduino –PIN10-

Se debe crear un array (vector) de datos compuesto por los valores correspondientes a las 8 notas que se pretende sacar:

int notas[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};

Se deben definir también el tiempo de pausa entre nota y nota y el tiempo de pausa de fin de secuencia de notas:

int tnota=100;
int pausa=100;

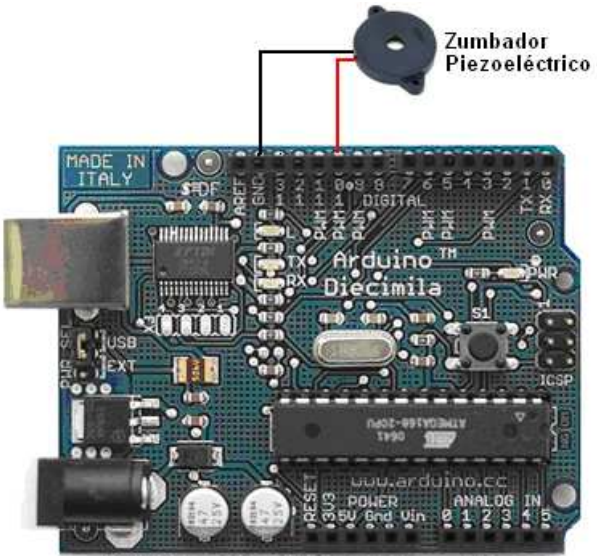
Las iteraciones para el recorrido de las 8 notas se realizan con una instrucción de tipo for:

for(n=0;n<8;n++)

El tiempo de activado y desactivado de la salida del zumbador también se resuelve con un bucle for:

for(m=0;m<=tnota;m++){

Esquema/Organigrama



Programa

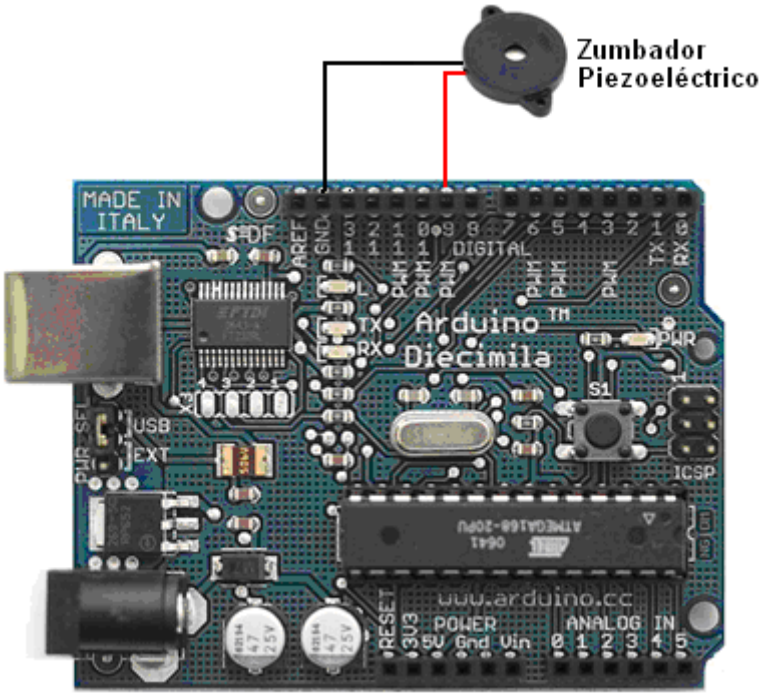
```
// Generador de Notas Musicales

int piezo=10;
int notas[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};  //cadena con los tiempos que corresponden a las distintas notas
int n=0;
int m= 0;
int tnota=100;                //nº de repeticiones del pulso. Nos da la duración de la nota
int pausa=100;
void setup() {
 pinMode(piezo,OUTPUT);
}
void loop(){
  for(n=0;n<8;n++){           //iteración que recorre la lista con las duraciones de los pulsos de cada nota
    for(m=0;m<=tnota;m++){
     digitalWrite(piezo,HIGH);
     delayMicroseconds(notas[n]);    //Tiempo en microsegundos que está a 5V la salida del piezoeléctrico
     digitalWrite(piezo,LOW);
     delayMicroseconds(notas[n]);    //Tiempo en microsegundos que está a 0V la salida del piezoeléctrico
    }
    delay(pausa);             //tiempo en silencio entre escalas
  }
}
```

19. Toca tonos desde el puerto serial

En este ejercicio usaremos un Piezo eléctrico para reproducir tonos, aprovechando la capacidad que tiene el procesador para producir señales PWM y así reproducir música.

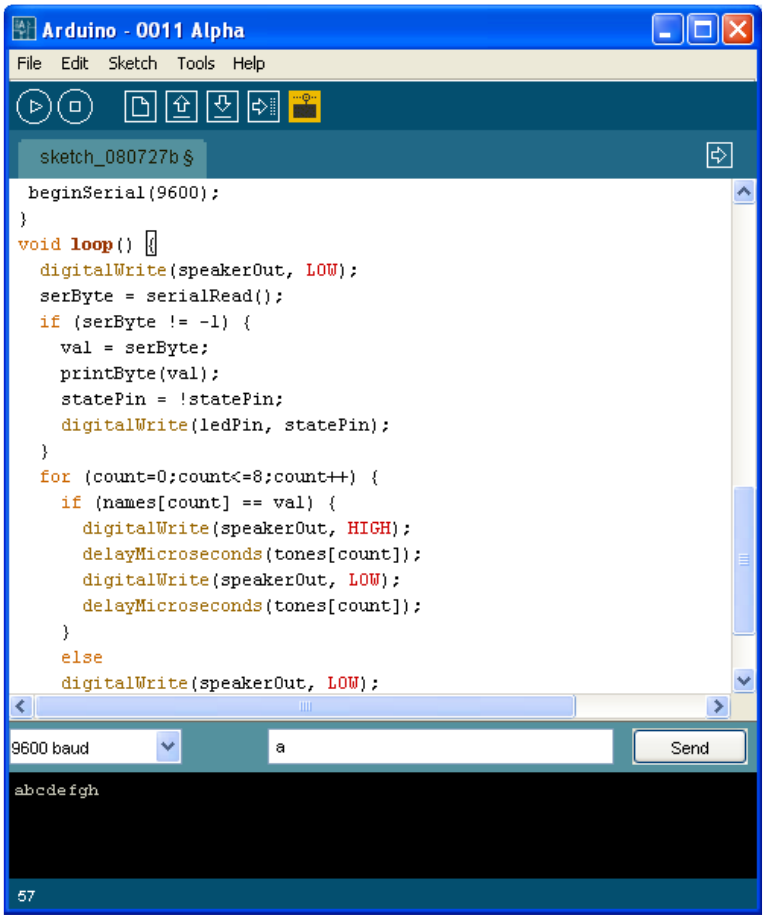
Un piezo eléctrico no es más que un dispositivo que puede ser usado tanto para reproducir o detectar tonos. En nuestro ejemplo estamos conectando el piezo en el pin número 9, que permite enviar una señal PWN .

Los tonos pueden ser generados a partir de cualquier programa capaz de enviar valores ASCII a través del puerto serial. Programas de Terminal, Processing, Pure Data o Director, pueden ser usados para generar los tonos. En nuestro caso los caracteres ASCII los enviaremos por la ventana de dialogo del IDE Arduino.



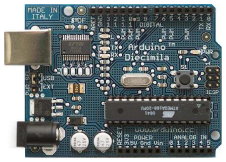
Ejemplo de la conexión de un piezo eléctrico al pin 9, y utilizaremos 8 tonos tal como se indica en el código del programa. Si pulsamos cualquier otra letra que no sea de las que tienen asignados tonos el zumbador deja de emitir tonos.

Aspecto del IDE Arduino enviando caracteres ASCII a la tarjeta para que estos se conviertan en tonos



Código:

```
/* Teclado Serial
 * ---------------
 *
 * Programa para reproducir tonos dependiendo de los datos que vengan del Puerto
 * serie. El cálculo de los tonos se realiza de acuerdo a la siguiente operación:
 *
 *      pulsoAlto = 1/(2 * frecuenciaTono) = periodo / 2
 *
 * Donde los tonos son descritos como en la siguiente tabla:
 *
 * tecla        frecuencia      periodo         PW (pulsoAlto)
 * c             261 Hz         3830            1915
 * d             294 Hz         3400            1700
 * e             329 Hz         3038            1519
```




```
 * f                 349 Hz          2864            1432
 * g                 392 Hz          2550            1275
 * a                 440 Hz          2272            1136
 * b                 493 Hz          2028            1014
 * C                 523 Hz          1912            956
 * Cualquier otra letra provoca el silencio del zumbador
 * (cleft) 2005 D. Cuartielles para K3
 * Trad. Juan C. Carvajal. Modificado J.M.Ruiz
 */
int ledPin = 13;
int speakerOut = 9;
byte names[] ={'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'};
int tones[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
byte val = 0;
int serByte = -1;
int statePin = LOW;
int count = 0;

void setup() {
 pinMode(ledPin, OUTPUT);
 pinMode(speakerOut, OUTPUT);
 beginSerial(9600);
}
void loop() {
  digitalWrite(speakerOut, LOW);
  serByte = serialRead();
  if (serByte != -1) {
    val = serByte;
    printByte(val);
    statePin = !statePin;
    digitalWrite(ledPin, statePin);
  }
  for (count=0;count<=8;count++) {
    if (names[count] == val) {
      digitalWrite(speakerOut, HIGH);
      delayMicroseconds(tones[count]);
      digitalWrite(speakerOut, LOW);
      delayMicroseconds(tones[count]);
    }
     else
       digitalWrite(speakerOut, LOW);
  }
}
```

20. Timbre de llamada

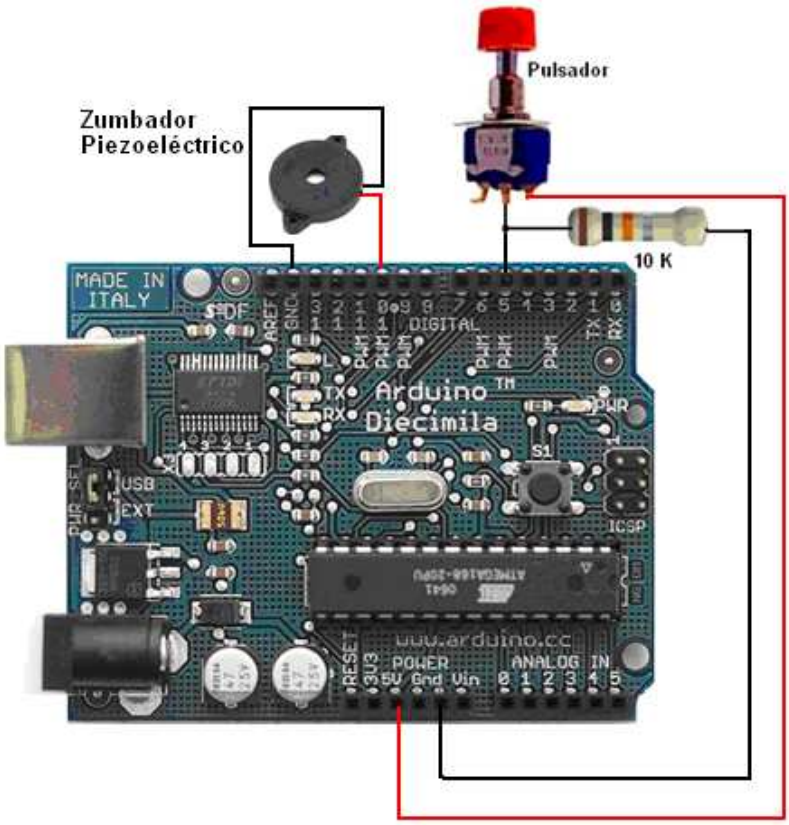
Se trata de realizar un timbre a través de un zumbador (salida 10) que emita dos tonos recogidos de una colección de ocho tonos, por ejemplo el *tono 0* y el **tono 6**. El timbre se activa mediante un pulsador conectado en el PIN5 (entrada digital).

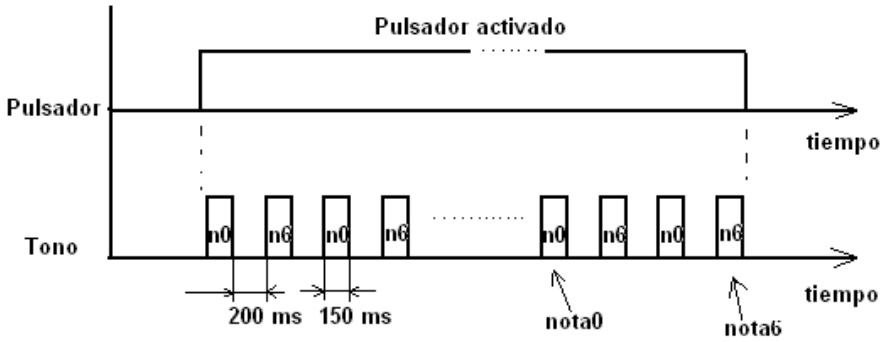
int notas[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};

tono0=1915 tono6=1014

Para la resolución de este ejemplo se sugiere crear un procedimiento llamado nota al que se incoará cuando se pulse el pulsador conectado en el PIN 5

Esquema/organigrama





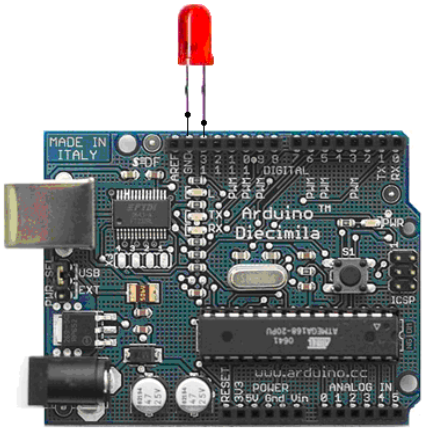
Programa

```
//Timbre de llamada

int notas[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956}; //definición de matriz de 8 notas
int puls=5;  // designación del pulsador de llamada
int zumb=10; // designación de la salida hacia el zumbador
int tnota=150;
int n=0;
int m=0;
void setup (){
 for(n=0;n<4;n++){
  pinMode(zumb,OUTPUT);
  pinMode(puls,INPUT);
 }
}
void nota(){   // rutina que genera los tonos de llamada
   for(m=0;m<=tnota;m++){
    digitalWrite(zumb,HIGH);
    delayMicroseconds(notas[n]);
    digitalWrite(zumb,LOW);
    delayMicroseconds(notas[n]);
   }
}

void loop(){
 if(digitalRead(puls)==HIGH){
  n=0;                    //elegimos la primera nota del timbre
  nota();                  //que aquí es la primera de la cadena
  delay(200);
```

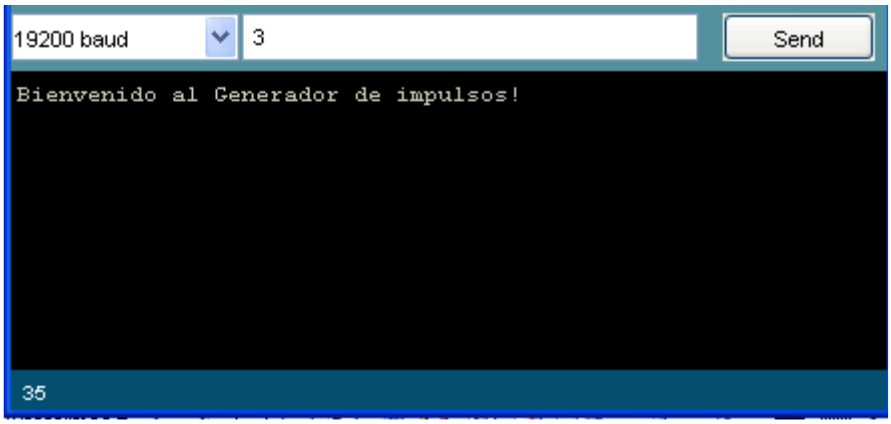
```
  n=6;                  //elegimos la segunda nota del timbre
  nota();               //que aquí es la sexta de la cadena
  delay(200);
 }
}
```

21. Enciende y apaga un número de veces un LED

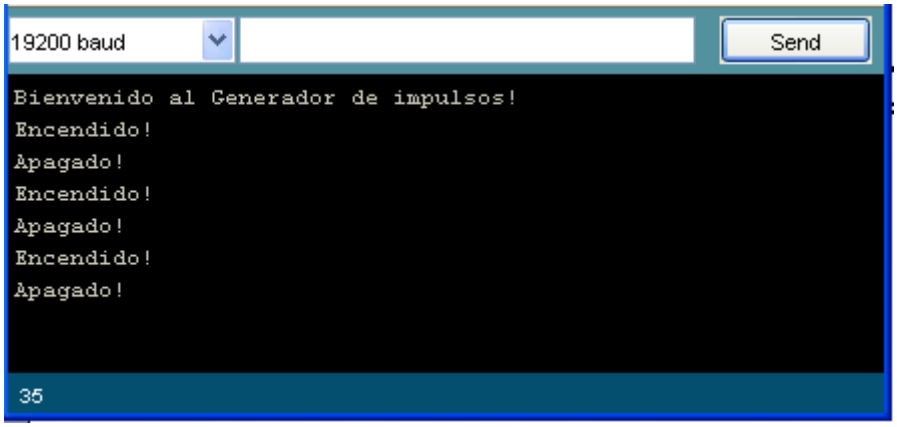
Se trata de realizar un ejemplo que active y desactive una salida digital (PIN13) un número de veces que indicaremos mediante un número a través del terminal del IDE Arduino.



Montaje en la tarjeta



Aspecto del terminal del IDE Arduino al iniciar el programa



Aspecto del terminal del IDE Arduino una vez generados los 3 impulsos por el terminal PIN13

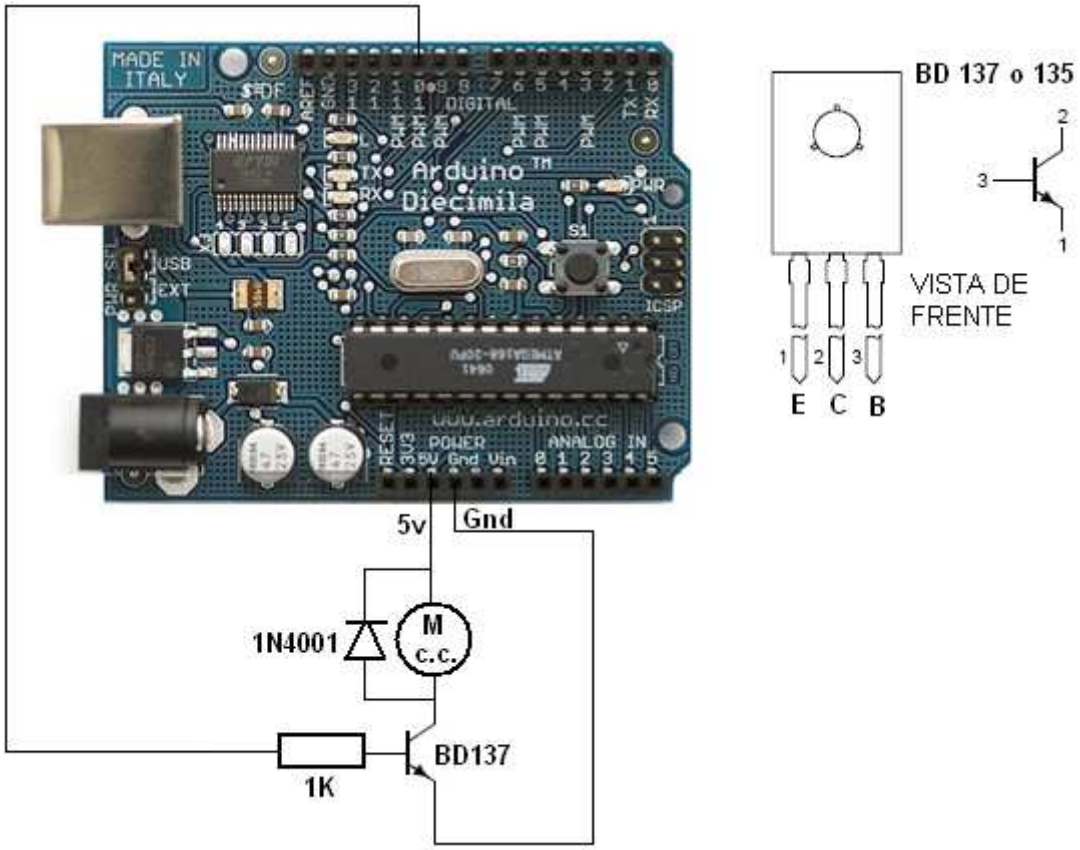
Solución:

```
/*
 * Impulsos programados
 * ----------------
 * Cambia de estado ON Off un lED conectado en el una salida digital
 * pin 13. El se encenderá y apagará tantas veces como se indique mediante un digito
* ASCII leído desde el puerto serie.
 *
 * Created 18 October 2006
 * copyleft 2006 Tod E. Kurt <tod@todbot.com>
 * http://todbot.com/
 *
 * based on "serial_read_advanced" example
 */

int ledPin = 13;   // selecciona el pin para el LED
int val = 0;       // variable que almacena el valor leído del puerto

void setup() {
  pinMode(ledPin,OUTPUT);    // declara el PIN del LED como salida
  Serial.begin(19200);       // conecta con el puerto serie a la velocidad de 19200
  Serial.println("Bienvenido al Generador de Impulsos Programados");
}

void loop () {
  val = Serial.read();      // lee el numero del puerto (una sola cifra)

  //si el valor leído es un solo digito se ejecuta el programa
```

```
  if (val > '0' && val <= '9' ) {
    val = val - '0';       // convierte el carácter leído en un numero
    for(int i=0; i<val; i++) {
      Serial.println("Encendido!");
      digitalWrite(ledPin,HIGH); // enciende el LED
      delay(150); // espera
      digitalWrite(ledPin, LOW); // apaga el LED
      delay(150); // espera
    }
  }
}
```

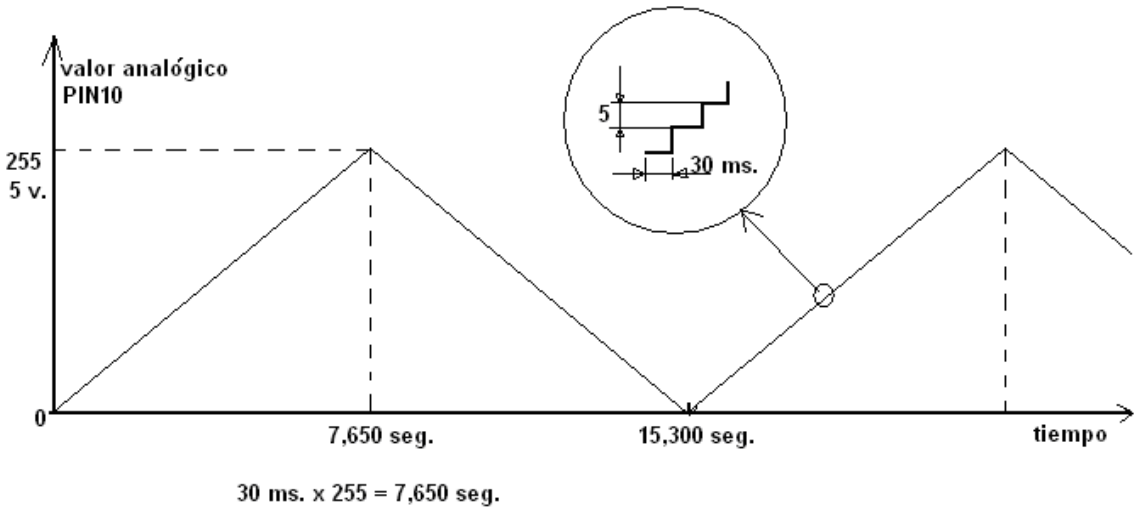
22. Control de un motor de cc con un transistor

Con este ejemplo vamos a controlar la velocidad de un motor de cc mediante la utilización de un transistor BD137. Se trata de utilizar la posibilidad de enviar una señal de PWM a una de las salidas configurables como salidas analógicas (para ATMEGA 168



Téngase en cuenta que el motor debe ser de bajo consumo por dos motivos: primero porque si alimentamos en las pruebas desde el conector USB no debemos sacar demasiada corriente del ordenador y segundo porque el transistor es de una corriente limitada.

El diodo 1N4001 se coloca como protección para evitar que las corrientes inversas creadas en el bobinado del motor puedan dañar el transistor.



La tensión que sacaremos a la salida 10 (analógica tipo PWM) variara en forma de rampa ascendente y descendente de manera cíclica tal como vemos en la figura. Este efecto lo conseguimos con una estructura del tipo for:

for(valor = 0 ; valor <= 255; valor +=5) (ascendente)
for(valor = 255; valor >=0; valor -=5) (descendente)

Obsérvese que los incrementos del valor de la tensión van de 5 en 5 y tenemos que considerar que 0v equivale a 0 y 5 v. equivale a 255.

Programa:

```
int valor = 0;    // variable que contiene el valor a sacar por el terminal analógico
int motor = 10;  // motor conectado al PIN 10
void setup() { } // No es necesario
void loop() {
  for(valor = 0 ; valor <= 255; valor +=5) {

// se genera una rampa de subida de tensión de 0 a 255 es decir de 0 a 5v
    analogWrite(motor, valor);
    delay(30); // espera 30 ms para que el efecto sea visible
  }
  for(valor = 255; valor >=0; valor -=5) {

// se genera una rampa de bajada de tensión de 255 a 0 es decir de 5 a 0v
   analogWrite(motor, valor);
   delay(30);
  }
}
```

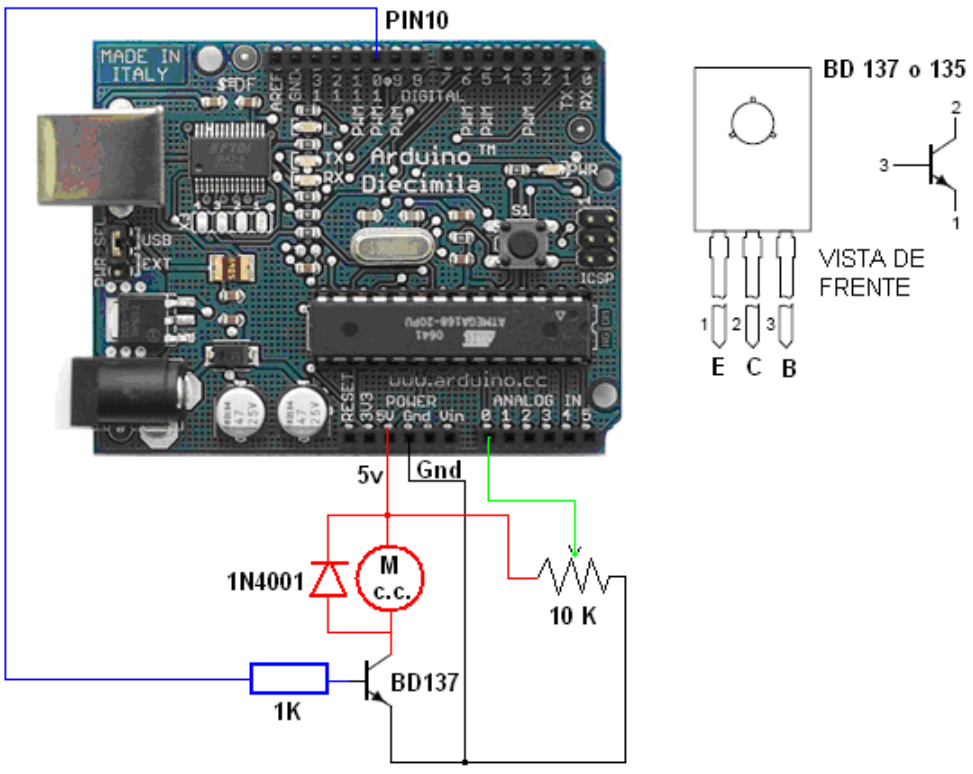
Variante del montaje: Control de la velocidad mediante un potenciómetro

Se trata de controlar la velocidad a nuestro gusto es decir mediante un potenciómetro que se coloca en una de las entradas analógicas y en función del valor que se lea en la entrada así girará mas o menos rápido el motor.

Programa:

```
int valor = 0;    // variable que contiene el valor a sacar por el terminal analógico
int motor = 10;  // motor conectado al PIN 10
int potenciometro=0; // Se define la entrada analógica

void setup() { } // No es necesario

void loop() {
        valor = analogRead(potenciometro); // se lee el valor de la entrada analógica y se asigna a val
        analogWrite(motor, valor); // Se manda a la salida analógica 0 el valor leído
        delay(30); // espera 30 ms para que el efecto sea visible
 }
```

Esquema



23. Control de un motor de cc con el driver L293D

Con esta aplicación vamos a mover un motor de cc haciendo uso de un CI de potencia que es especifico para estas aplicaciones. El circuito podrá mover hasta dos motores, nosotros solo lo haremos con uno.

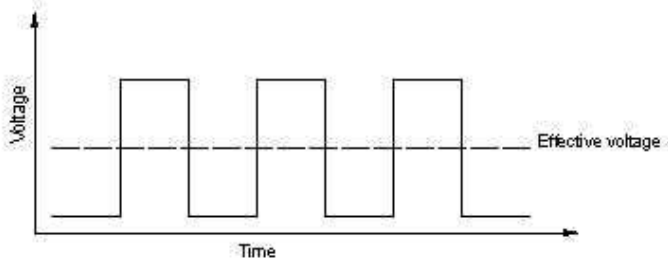
Como ventana en este montaje podremos mover el motor en los dos sentido de giro cosa que con el anterior montaje no podíamos.

El funcionamiento será como el primer montaje del motor anterior es decir vamos a crear una rampa de subida de tensión y una de bajada con el fin de que el motor modifique su velocidad de modo automático.

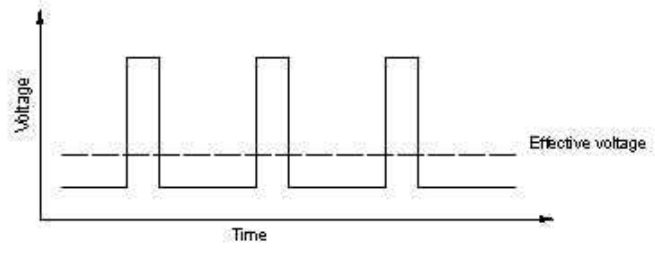
Control o Driver de un motor de continua:

Los dos parámetros que queremos controlar de un motor de continua, es su dirección de giro y su velocidad. La dirección se controla cambiando su polaridad. En cambio, para su velocidad, debemos utilizar la técnica de modulación por ancho de pulso-PWM.

Aquí hay algunos gráficos donde se muestra la relación entre la señal de pulsos (PWM) y el voltaje efectivo:



Cuando el tiempo que el pulso está activo es la mitad del periodo de la señal o el parámetro duty cycle está al 50%, el voltaje efectivo es la mitad del voltaje total de entrada.



Cuando el duty cycle es reducido al 25%, el voltaje efectivo es un cuarto del voltaje total de entrada. Entonces la velocidad del motor disminuye.

De esta forma controlando el duty cycle o el tiempo que el pulso está activo (frecuencia), podemos controlar la velocidad del motor de continua.

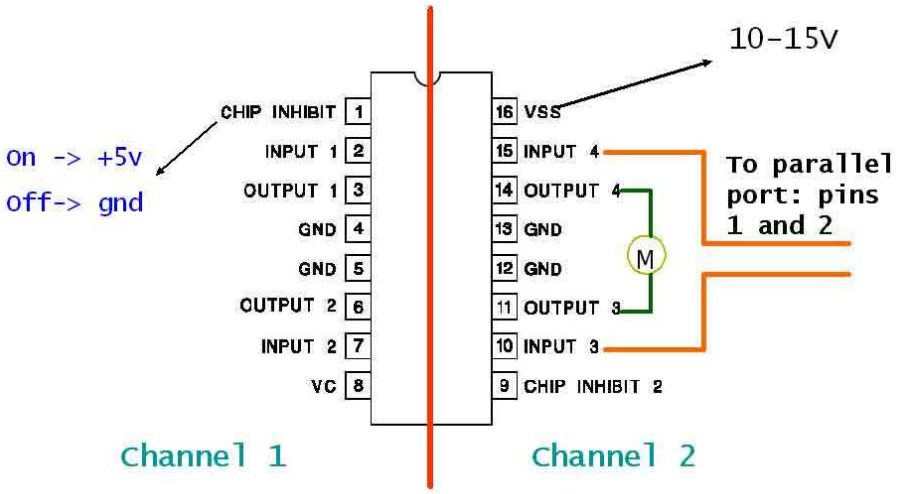
Una forma de realizar dicho control en Arduino, es utilizando la salida analógica PWM. Hay que recordar que la señal de salida PWM (pines 9,10) es una señal de frecuencia constante (30769 Hz) y que sólo nos permite cambiar el "duty cycle" o el tiempo que el pulso está activo (on) o inactivo (off), utilizando la función analogWrite().

La otra forma es generando señales PWM utilizando la capacidad del microprocesador a través de la función digitalWrite ().

Si queremos controlar simultáneamente la velocidad y dirección de un motor, necesitamos utilizar un circuito integrado o chip llamado de forma general como "puentes H", por ejemplo como el L293D.

Chip L293D/B(puente H):

Es un circuito integrado o chip, que puede ser utilizado para controlar simultáneamente la velocidad y dirección de dos motores de continua (contiene dos puentes H). La diferencia entre el modelo L393D y L293B, es que el primero viene con diodos de protección que evita los daños producidos por los picos de voltaje que puede producir el motor.



Contiene 4 pines digitales (2,7,10, 15) para controlar la dirección de los motores.

Los pines "enable" (1,9) admiten como entrada una señal PWM, y se utiliza para controlar la velocidad de los motores con la técnica de modulación de ancho de pulso.

Los motores van conectados entre uno de los pines 3, 6, 11, o 14.

La tensión Vss es la que alimentará o dará potencia al motor.

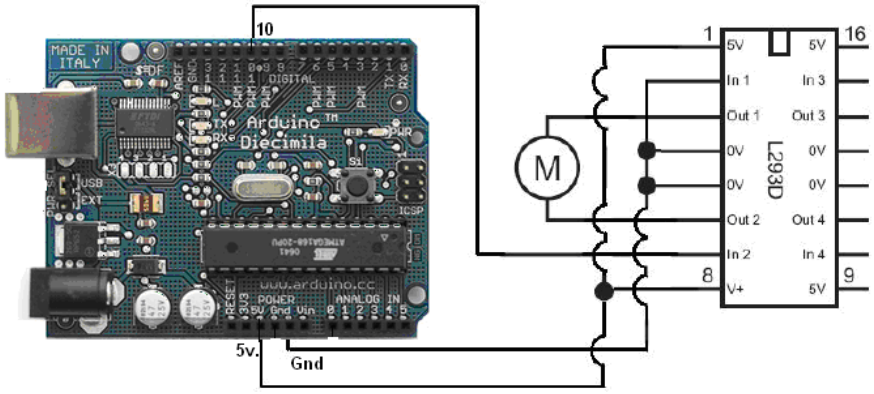
Montaje Básico: Control simple de un motor con el CI L293 a velocidad constante

En un primer lugar sólo vamos a demostrar el control de la velocidad de un motor de continua a través del integrado L293D. Para ello fijamos los pines de control de dirección a 5v y 0v,de forma que sólo girará en un sentido. Si queremos cambiar el sentido, sólo será necesario cambiar dicha polarización.

Programa

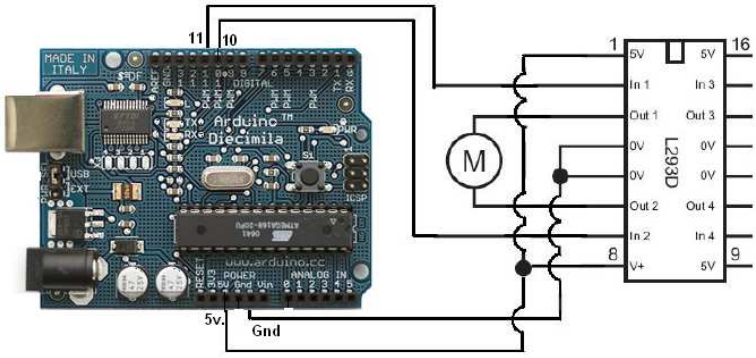
```
// Control simple de un motor con el CI L293 a velocidad constante

int motorpin =10; // PIN de salida analógica PWM
void setup() { }
void loop() {
  analogWrite(motorpin, 125);  // activa el motor a una velocidad constante
  delay(100);                  // espera 100 ms para la próxima lectura
}
```

Esquema



24. Control de un motor: velocidad variable y sentido de giro variable (1ª opción)

Esquema



Programa

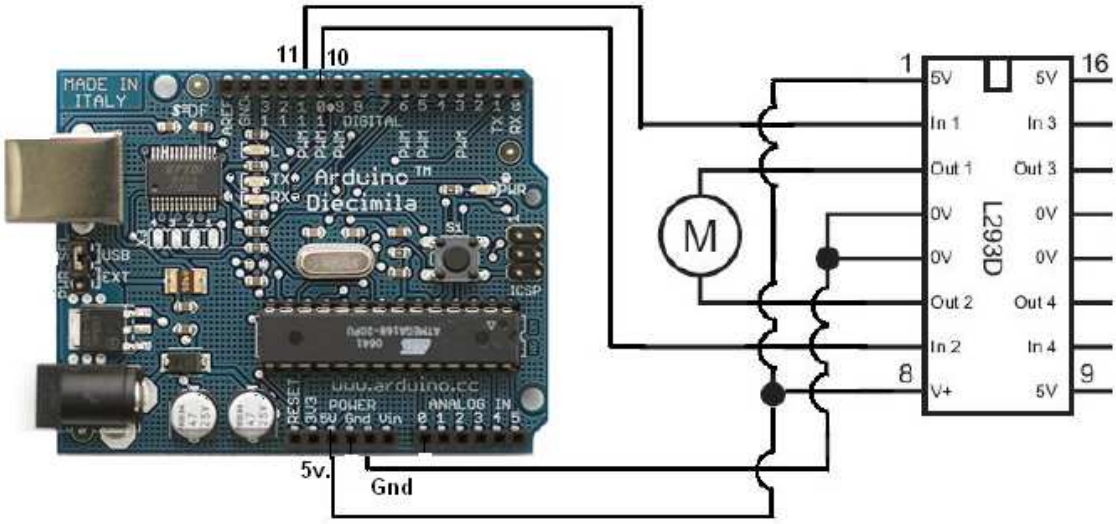
```
// Control de Motor con driver L293D
int valor = 0;                  // variable que contiene el valor
int motorAvance = 10;           // Avance motor --> PIN 10
int motorRetroceso = 11;        // Retroceso motor --> PIN 11
void setup() { }                // No es necesario

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);
  }
}
```

25. Control de un motor: velocidad variable y sentido de giro variable (2ª opción)

Vamos a demostrar el control simultáneo de la velocidad y del sentido de un motor de continua.

```
!OJO probar en Arduino con un sólo motor
```

Esquema



/* PWM Motor by DojoCorp <http://www.0j0.org> Demonstrates the use of analog output pins (PWM) controlling a motor. Description of the pinout used by the motor control board: - pwm pin 1 controls the speed on the A side of the H bridge - pwm pin 2 controls the speed on the B side of the H bridge - digital pin 8 controls the direction for motor B - digital pin 9 controls the direction for motor A Created 19 March 2005 */

int value = 0; // valor actual
int pwmpinA = 1; // motor A conectado a la entrada analógica pin1
int pwmpinB = 2; // motor B conectado a la entrada analógica pin 2
int dirpinA = 9; // motor A pin dirección
int dirpinB = 8; // motor B pin dirección
boolean directionA = true; // variable estado de giro del motor A
boolean directionB = true; // variable estado de giro del motor B
void setup() {
  // no es necesario setup

```
}

void loop() {
  if (directionB) {
    digitalWrite(dirpinB,HIGH);
  } else {
    digitalWrite(dirpinB,LOW);
  }
  directionB = !directionB;
  // varia velocidad de mínimo a máximo
  for(value = 0 ; value <= 255; value+=5) {
// envía al pin del salida del motor A el valor de velocidad de 0 a 255
    analogWrite(pwmpinA, value);
// envía al pin del salida del motor B el valor de velocidad de 0 a 255
    analogWrite(pwmpinB, value);
    delay(30);                  // espera  30 ms
  }
  // varia velocidad de máximo a mínimo
  for(value = 255; value >=0; value-=5) {
    analogWrite(pwmpinA, value);
    analogWrite(pwmpinB, value);
    delay(30);
  }
}
```

26. Utiliza un relé para encender dispositivos de 220V

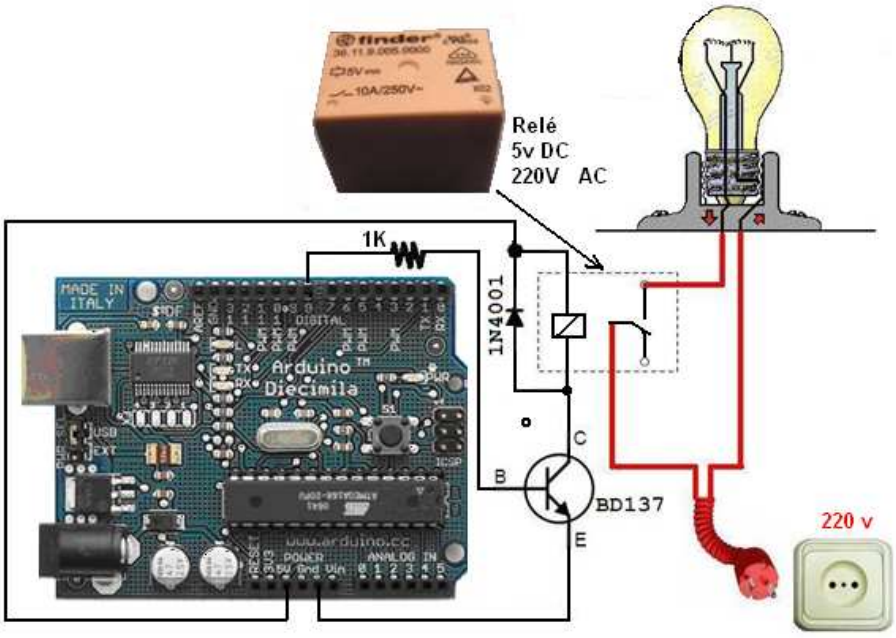
Este sencillo ejemplo enseña como encender una bombilla de 220V de corriente alterna (AC) mediante un circuito de 5V de corriente continua (DC) gobernado por Arduino. Se puede utilizar con cualquier otro circuito de 220V con un máximo de 10A (con el relé del ejemplo).

¿Qué es un relé?

El relé es un dispositivo electromecánico, 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. (fuente: Wikipedia)

De aquí extraemos una información muy importante: Podemos separar dos circuitos de forma que funcionen con voltajes diferentes. Uno a 5V (Arduino) y otro a 220V (la bombilla).

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.



Esquema de conexionado a una placa Arduino

Código fuente

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