

# Tutorial\_01.- ARDUINO: Entradas y salidas digitales

## Contenido

<b>1. Salidas digitales.....</b>	<b>3</b>
Contenido teórico: Cálculo de R limitadora para un LED .....	3
Proyecto01_01: Encendido/Apagado de dos LEDs alternativamente .....	7
<b>2. Entradas digitales.....</b>	<b>11</b>
Contenido teórico: Resistencias de Pull-up y Pull-down.....	11
Contenido teórico: Pulsador .....	13
Proyecto01_02: Lectura de un pulsador con R de Pull-down externa.....	14
Contenido teórico: Resistencia de Pull-up interna .....	17
Proyecto01_03: Encendido de un Led encendido mediante un pulsador usando la resistencia interna de Pull-up.....	18
Proyecto01_04: Encendido/Apagado de un Led mediante un SOLO pulsador. Con resistencia interna de Pull-up.....	20
Contenido teórico: Filtrado de rebotes en pulsadores .....	21
Proyecto01_05: Lectura de un pulsador, evitando rebotes, para encender y apagar un LED con un mismo pulsador.....	23
Contenido teórico: Definiendo “arrays” .....	25
Contenido teórico: Declarando funciones .....	26
Proyecto01_06: Seleccionando entre dos animaciones de LED mediante un solo pulsador. 27	
<b>3. Ejercicios .....</b>	<b>28</b>

### Material necesario

- Placa Arduino
- Cable micro USB
- Placa Protoboard
- Cables Jumper-Wire
- Componentes electrónicos:
  - LEDs
  - Resistencias
  - Pulsadores

### Conceptos teóricos

- Diodo LED: Corriente directa ( $I_F$ ), tensión directa ( $V_F$ ) , longitud de onda y color
- Cálculo del valor de una resistencia limitadora (Ley de Ohm)
- Resistencias de Pull-up y Pull-Down
- Filtrado de rebotes de un pulsador

### Estructuras de programación en Arduino (C/C++)

- Funciones setup() y loop() de Arduino
- Condicional *If*
- Bucle *For*
- Arrays
- Declaración de funciones

### Funciones propias de Arduino

- pinMode()
- digitalWrite()
- digitalRead()

## **IMPORTANTE:**

Comprobaremos que nuestra placa **Arduino** está **desconectada** y sin energía, puesto que de no ser así podría dañarse tanto la placa, como el equipo. Una vez hemos realizado esta comprobación, pasaremos a realizar el montaje.

## 1. Salidas digitales

### Contenido teórico: Cálculo de R limitadora para un LED

Los pines digitales de Arduino pueden ser configurados para actuar como entradas o salidas. Su modo **por defecto** es el de **entrada**, y están en estado de alta impedancia, esto es, como si tuviesen conectada una resistencia muy grande, de unos 100 Megaohms.

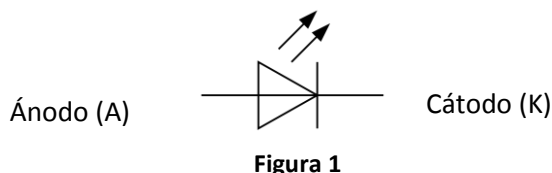
Para utilizar la E/S digitales **como salidas** debe configurarse por software, función **pinMode**, y entonces el pin correspondiente pasará a un estado de baja impedancia, siendo capaz de **proporcionar hasta 40 mA** de corriente. Para limitar esta corriente será recomendable conectar siempre la carga a través de una **resistencia limitadora**.

#### Pines configurados como salidas:

Cuando un pin es configurado como salida pasa a un estado de baja impedancia. De este modo, el pin será capaz de proporcionar **hasta 40mA** a una carga a él conectada. Esa corriente es suficiente, aunque **a veces excesiva**, para hacer lucir LEDs y activar algunos sensores, pero no para otros elementos como motores, relés o solenoides. En general, es recomendable no drenar más de 35mA de los pines de salida.

Si se intenta demandar más corriente de un pin de salida, el pin se destruirá aunque el resto del chip microcontrolador seguirá funcionando.

La carga a conectar en este primer ejemplo es un diodo LED, **Light-Emitting Diode**. Este tipo de diodo presenta la característica de emitir luz cuando es polarizado adecuadamente, esto es, de forma directa. Su símbolo es



Un **diodo LED**, al igual que cualquier otro tipo de diodo, es capaz de conducir la corriente eléctrica cuando está **polarizado directamente**. Esto es, tal y como vemos en el circuito de abajo

Si la polaridad de la fuente de tensión se invirtiera, el diodo quedaría **polarizado inversamente**

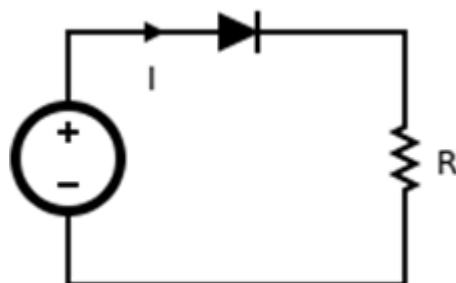


Figura 2

y bloquearía el paso de la corriente eléctrica I.

Tal y como se comentó anteriormente, un pin de Arduino configurado como salida, es capaz de suministrar hasta 40mA de corriente. Será necesario saber si esa corriente es suficiente para encender el LED, o bien, si es excesiva pudiendo quemar el LED.

Para saberlo, debe leerse la hoja de características del LED y averiguar cual es el intervalo de corriente adecuado para que luzca sin estropearse.

**Absolute Maximum Ratings: (Ta=25°C) .**

ITEMS	Symbol	Absolute Maximum Rating	Unit
Forward Current	I <sub>F</sub>	20	mA
Peak Forward Current	I <sub>FP</sub>	30	mA
Suggestion Using Current	I <sub>SU</sub>	16-18	mA
Reverse Voltage (V <sub>R</sub> =5V)	I <sub>R</sub>	10	uA
Power Dissipation	P <sub>D</sub>	105	mW
Operation Temperature	T <sub>OPR</sub>	-40 ~ 85	°C
Storage Temperature	T <sub>STG</sub>	-40 ~ 100	°C
Lead Soldering Temperature	T <sub>SOL</sub>	Max. 260°C for 3 Sec. Max. (3mm from the base of the epoxy bulb)	

Figura 3. Fuente: <https://www.sparkfun.com/datasheets/Components/LED/COM-09590-YSL-R531R3D-D2.pdf>

La tabla de arriba, establece un valor de 20mA como la **corriente directa, I<sub>F</sub>**, máxima que debería atravesar el diodo, también recomienda limitar dicha corriente a un valor entre 16 y 18 mA. Para ello, y tal como aparece en el circuito de la figura 3, se incluirá una **resistencia limitadora en serie** con el diodo.

Será necesario calcular el valor óhmico de la resistencia para conseguir limitar la corriente que atravesará el diodo. Este cálculo se efectuará mediante **la ley de Ohm**.

$$\text{Voltaje (V)} = \text{Resistencia(R)} * \text{Intensidad (I)}$$

Para el circuito de la figura 3, es importante recordar, que cualquier elemento electrónico produce una caída de tensión en él. De modo que no todo el voltaje V aportado por la fuente de tensión llegará a la resistencia R, según la **ley de tensión de Kirchhoff**, para la única malla del circuito de la figura 3, se cumple:

$$V_{CC} = V_{\text{diodo}} + R * I$$

Para esta ecuación nos falta averiguar el dato de la **caída de tensión en el diodo, V<sub>D</sub>**.

Ese dato aparece también reflejado en la hoja de características del diodo, y varía según el color elegido para la luz del **LED**. El color del LED elegido en la hoja de características se identifica por la **longitud de onda** que aparece, aproximadamente 620 nm. Esa longitud de onda corresponde a la luz roja.

De la hoja de características se extrae, que la caída de tensión que presentará el LED rojo será de entre 1.8 y 2.2 voltios.

Absolute Maximum Ratings: (Ta=25°C)

ITEMS	Symbol	Test condition	Min.	Typ.	Max.	Unit
Forward Voltage	$V_F$	$I_F=20\text{mA}$	1.8	---	2.2	V
Wavelength (nm) or TC(k)	$\Delta \lambda$	$I_F=20\text{mA}$	620	---	625	nm
*Luminous intensity	$I_v$	$I_F=20\text{mA}$	150	---	200	mcd
50% Viewing Angle	$2\theta_{1/2}$	$I_F=20\text{mA}$	40	---	60	deg

Figura 4

Se puede elegir como valor de la caída de tensión en el diodo el valor promedio del rango reflejado en la hoja de características. Esto es, 2 voltios.

Aplicando la Ley de tensión de Kirchoff y la Ley de Ohm

$$V_{CC} = V_F + R * I_F$$

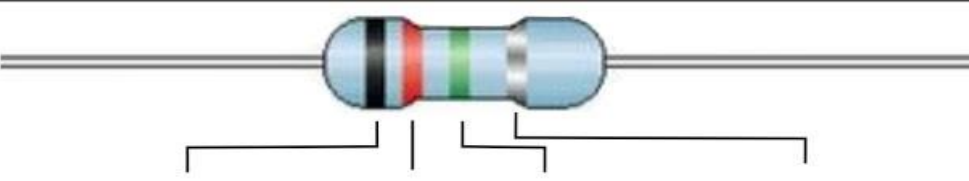
$$5V = 2V + R * 18\text{mA}$$

$$R = 3V / 18\text{mA} = 166 \text{ ohmios}$$

A continuación, habrá que elegir una resistencia con valor estándar cercano al valor calculado. Si se elige una resistencia de valor **220 ohmios**, gracias a la ley de ohm, se comprueba que la corriente quedará limitada a unos **14 mA**. Bastará comprobar que la luz emitida es suficiente.

Si por el contrario se eligiera un valor estándar inferior al calculado, la corriente sería superior a los 18 mA empleados en el cálculo, quedando por encima del valor recomendado.

Para identificar las resistencias por su valor se emplea un código de colores reflejado en la figura siguiente.



Color	1ra. Banda	2da. Banda	3ra. Banda Multiplicador	Tolerancia %
Negro	0	0	x1	
Cafe	1	1	x10	
Rojo	2	2	x100	2%
Naranja	3	3	x1000	
Amarillo	4	4	x10000	
Verde	5	5	x100000	
Azul	6	6	x1000000	
Violeta	7	7	x10000000	
Gris	8	8	x100000000	
Blanco	9	9	x1000000000	
Circuitos Básicos				Dorado 5%
				Plata 10%

Figura 5. Código de colores para resistencias

Otro parámetro importante a la hora de elegir una R es la **potencia** que es capaz de disipar en vatios. Este valor se calcula con la fórmula:

$$P = R \cdot I^2$$

Si la resistencia se expresa en ohmios y la intensidad o corriente en amperios, la unidad resultante será el **watio (W)**.

Para el ejemplo que nos ocupa, se calcula que la potencia que deberá ser capaz de disipar la R elegida es de:

$$P = 220 \text{ ohm} * (0.014)^2 \text{ A} = 0.043 \text{ W} = 43\text{mW}$$

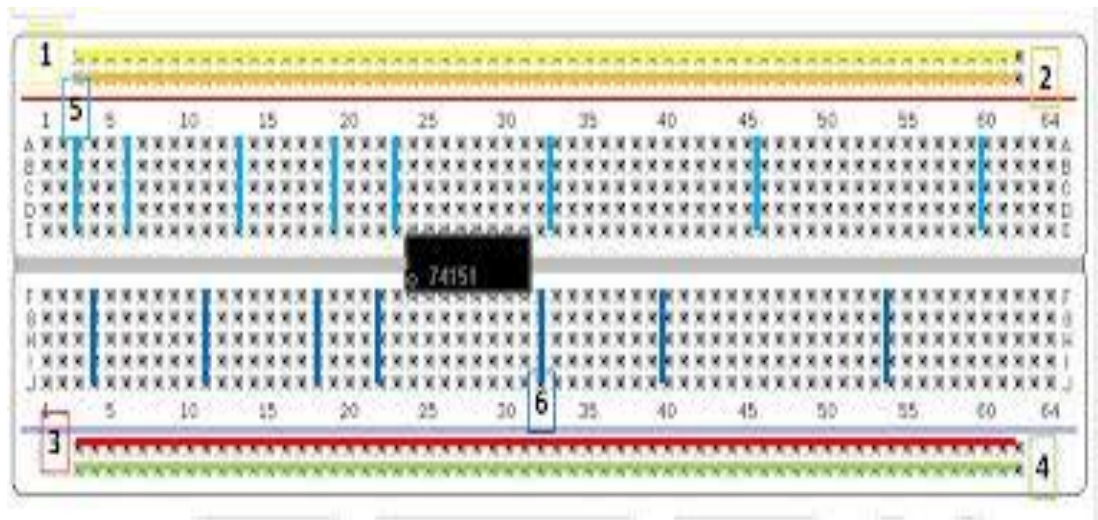
Las resistencias comerciales suelen disipar 0.125W, 0.250W, 0.600W, 1W, etc. En el laboratorio trabajarás principalmente con resistencias de 0.250W.

## Proyecto01\_01: Encendido/Apagado de dos LEDs alternativamente

*El siguiente proyecto enciende y apaga dos leds de forma alterna, esto es, mientras uno luce, el otro se mantiene apagado.*

### Material

- 1 Led rojo, 1 Led verde, disponibles en 5mm
- 2 Resistencias limitadoras a **calcular** y elegir.
- Placa Protoboard: Ver figura para recordar el interconexionado de la placa.



### Funciones en Arduino

#### digitalRead(Pin)

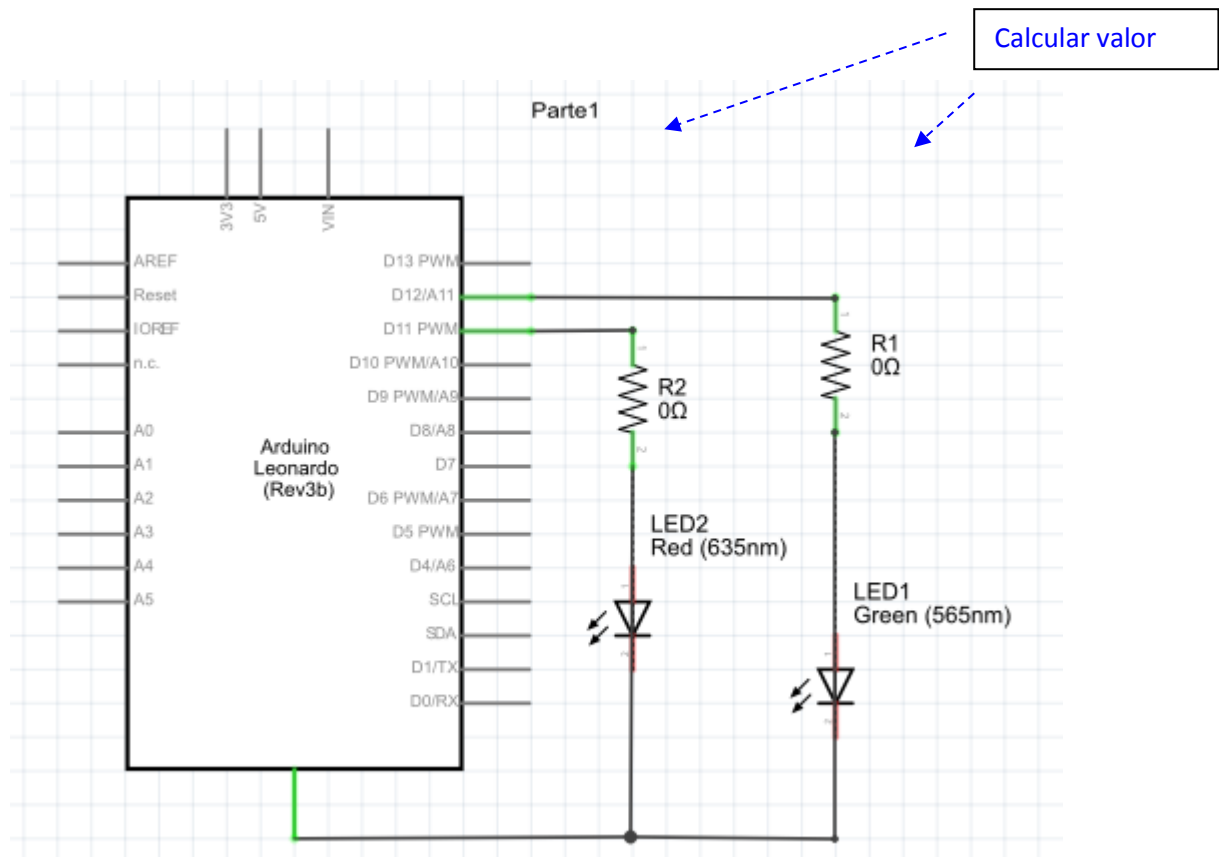
Este código incluye una nueva función para leer el valor introducido por una entrada digital.

<http://arduino.cc/en/Reference/DigitalRead>

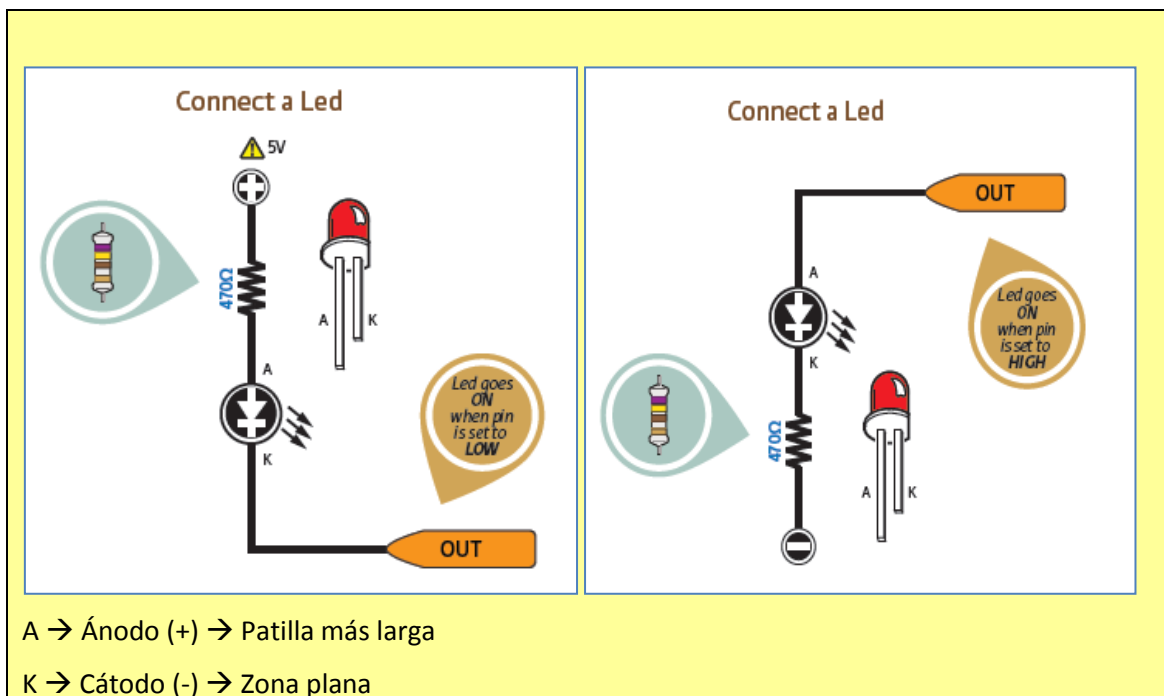
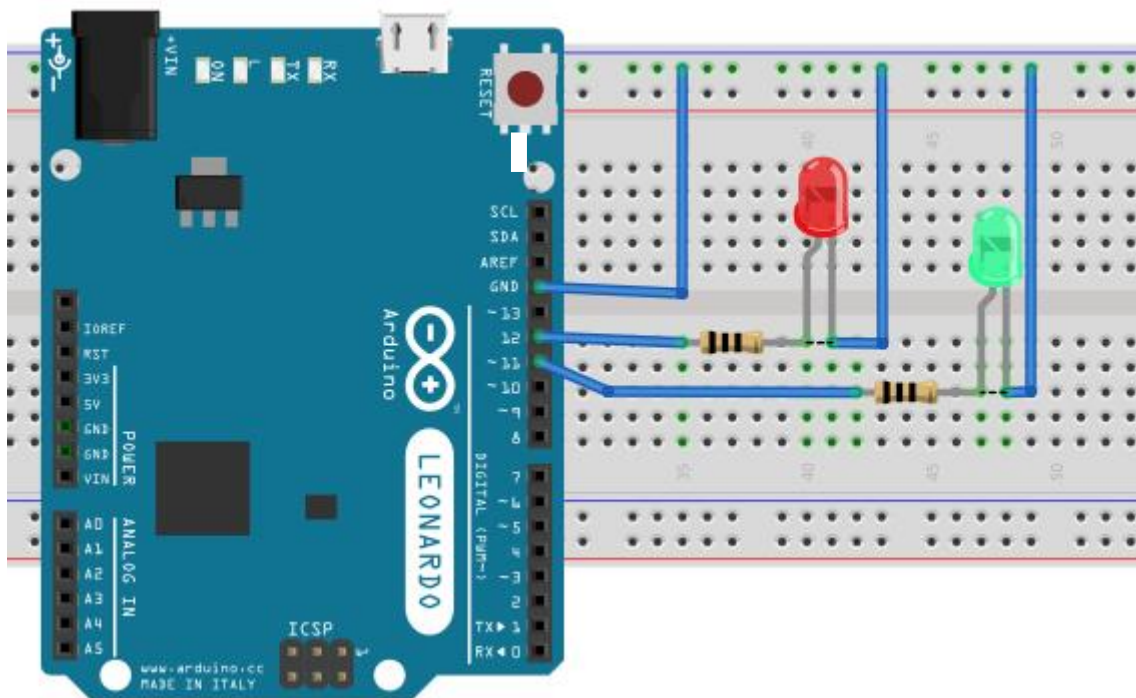
## Esquema de montaje:

Recuerde determinar el valor de las resistencias limitadoras.

**!!! Recuerda la POLARIDAD DE LOS DIODOS LED!!!**







### Código:

```
const int GreenLED = 11; // Alias for pin number 11
const int RedLED   = 12; // Alias for pin number 12

void setup() {

  // initialize the LED pin as an output:
  pinMode(GreenLED, OUTPUT);
  pinMode(RedLED, OUTPUT);

}

void loop(){

  //turn the Green Led on:
  digitalWrite(GreenLED, HIGH);
  digitalWrite(RedLED, LOW);

  // Wait for 1 second
  delay(1000);

  // turn the Red Led on
  digitalWrite(RedLED, HIGH);
  digitalWrite(GreenLED, LOW);

  // Wait for 1 second
  delay(1000);
}
```

## 2. Entradas digitales

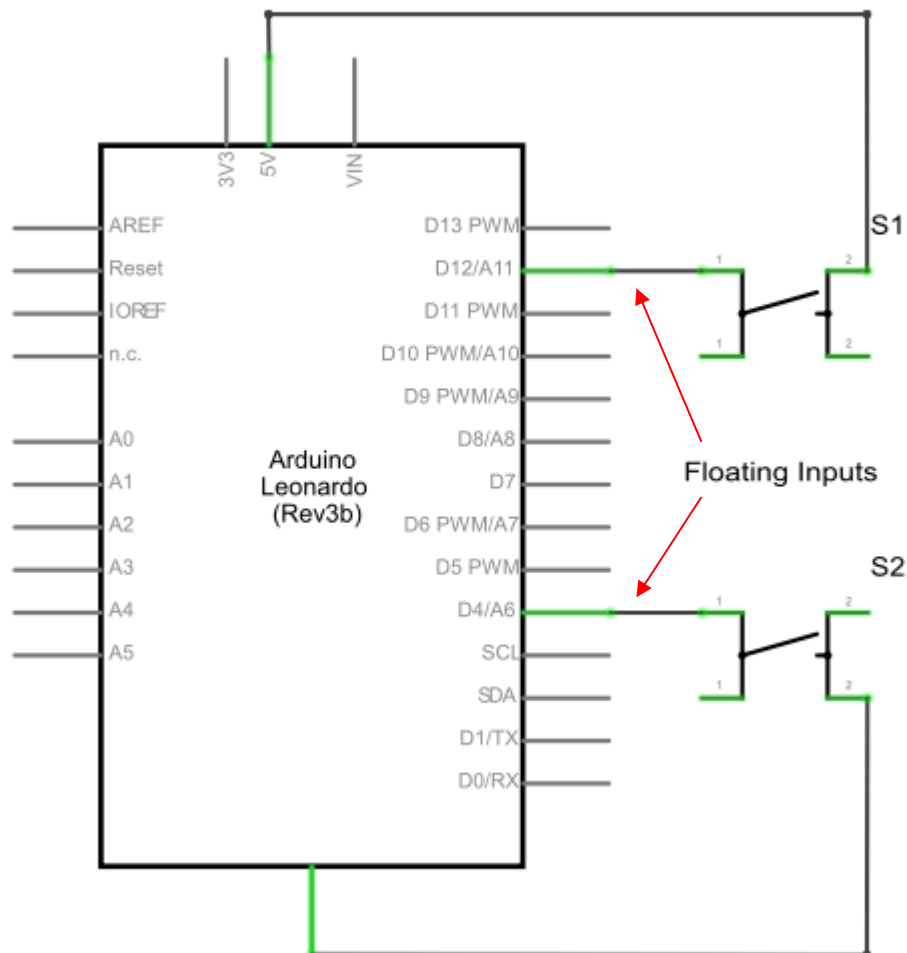
### Contenido teórico: Resistencias de Pull-up y Pull-down

En Arduino, las entradas digitales se interpretan de la siguiente manera:

- 1 (lógico) = High = +5v
- 0 (lógico) = Low = 0v

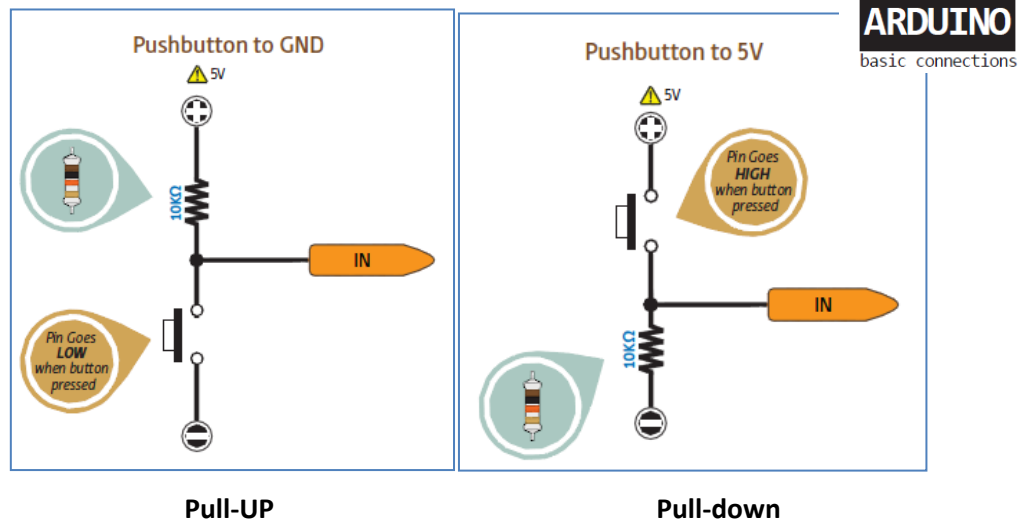
Esto quiere decir que si tenemos una tensión de +5v aplicada a una entrada, ésta la interpretará como "1" (HIGH) y si está conectada a 0V o GND, se interpretará como "0" (LOW).

Si un pin de E/S configurado como entrada no está conectado, entonces se dice que el **pin está "flotando"**. Esta situación da lugar a que la patilla reciba ruido del entorno y que Arduino lo pueda interpretar como "0" o como "1" indistintamente.

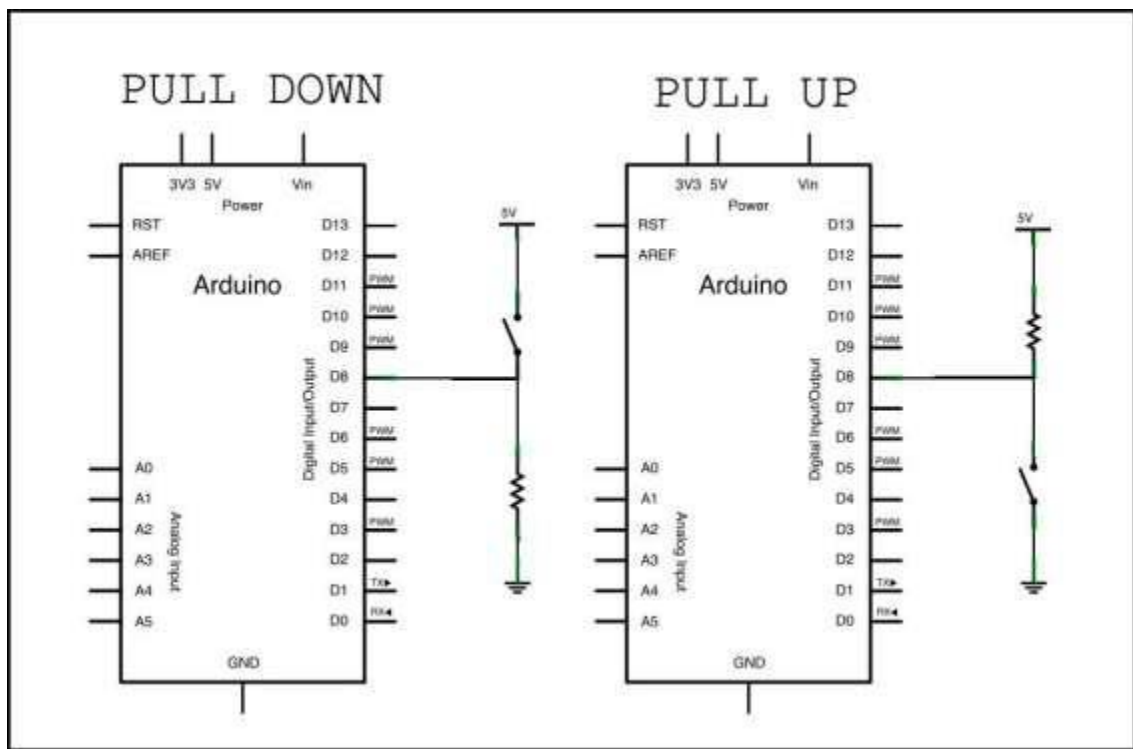


Cuando los pulsadores no están conectados, tanto la entrada 12 como la 4 quedan en modo "flotante".

Para evitarlo es necesario dar un valor estable al pin de entrada cuando el pulsador esté en estado de reposo, para ello se utilizan las resistencias de **Pull-up** y de **Pull-Down** (En la figura ambas resistencias se resuelven con un valor de 10K).

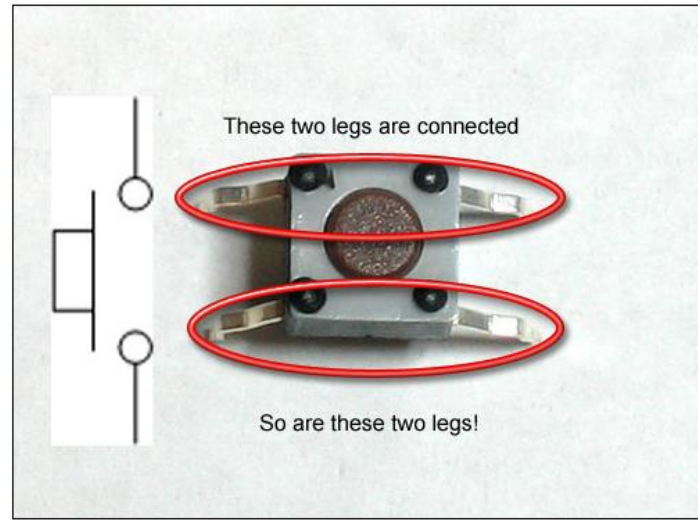


El siguiente esquema muestra cómo se conectarían ambas resistencias en un circuito básico de la entrada 8 de Arduino.



## Contenido teórico: Pulsador

Como se puede observar, los pulsadores electrónicos parecen llevar cuatro patillas, pero en realidad sólo son dos muy largas que salen por los laterales del pulsador para facilitar su conexión.



## Proyecto01\_02: Lectura de un pulsador con R de Pull-down externa

*En este proyecto, nuestro programa leerá el valor que introduce un pulsador en una entrada digital. En base a dicho valor, activaremos, o no, una salida digital a la que conectaremos un diodo LED. Al pulsar deberá encenderse el LED y apagarse al soltar el pulsador.*

Es un ejemplo sencillo, pero que nos permite aclarar un concepto muy importante sobre las entradas digitales, las resistencias de Pull-Up o Pull-Down.

Deberemos colocar el LED correctamente polarizado con su resistencia, y el pulsador con una **resistencia de PULL-DOWN de 10KΩ a masa** para cuando esté en reposo. De este modo, cuando pulsemos el botón, llegará una tensión 5v a la entrada donde esté conectado el pulsador y Arduino lo interpretará como un "1" lógico. A partir de aquí el programa hará el resto, proporcionándonos la tensión de salida que activará el diodo LED.

### Material:

- 1 Led verde o rojo
- 1 Resistencia de 220 Ω.
- 1 Resistencia de 10 KΩ (Pull-down).
- 1 Pulsador

### Estructuras de programación en Arduino

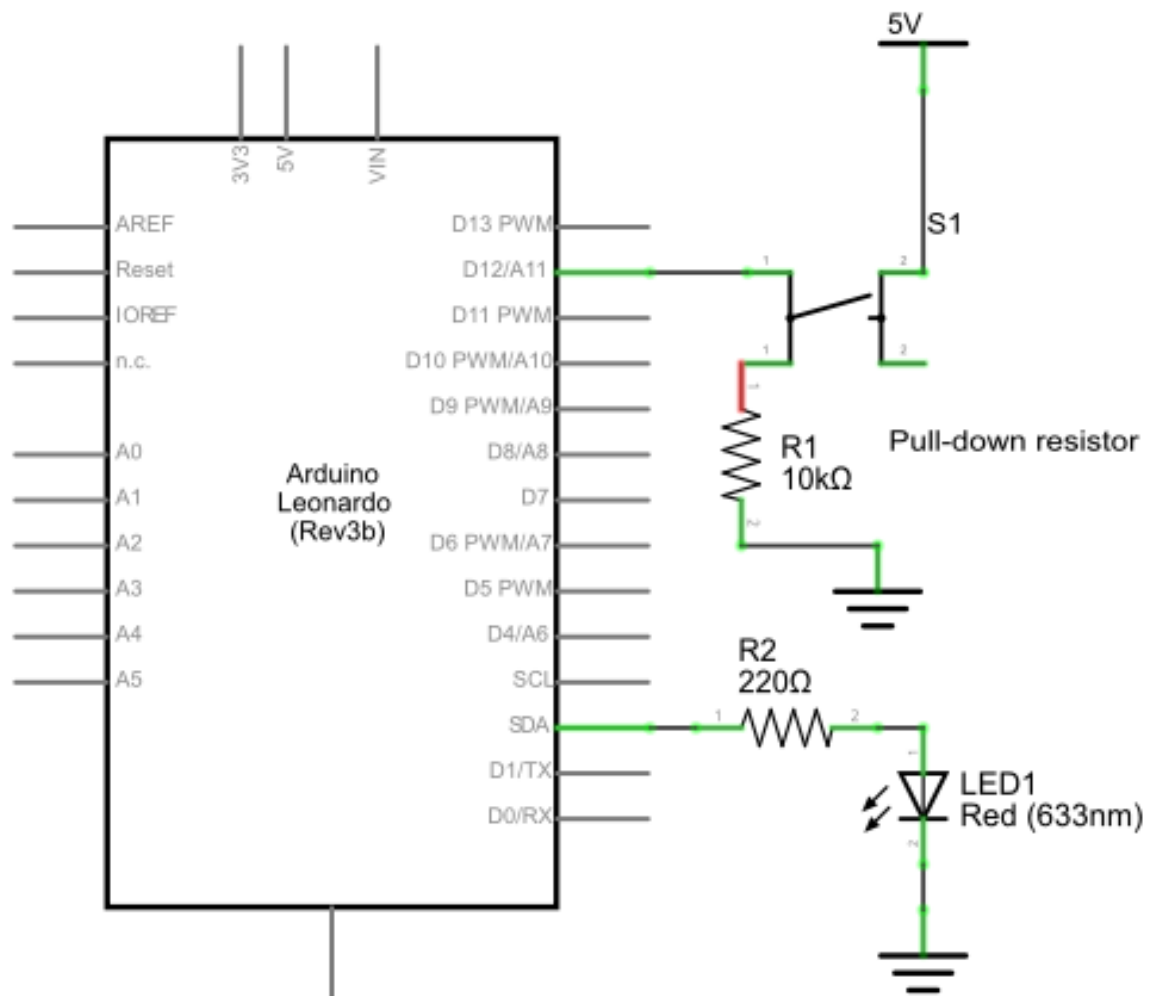
#### **If-else**

Este código incluye una estructura condicional.

<http://arduino.cc/en/Reference/If>

<http://arduino.cc/en/Reference/Else>

## Esquema



D3  
D2

## Código

```
/*Switch on the LED while the pushbutton is pressed*/

//Ports definition
const int GreenLED = 2;
const int PushButton = 12; //Alias for the pin with the pushbutton

// Variable to store the read value in the pushbutton
// It is declared as integer 16-bit
// The variable is initialized to 0
int Value_PushButton =0;

void setup() {
  //Ports setup
  pinMode(GreenLED,OUTPUT);
  // Line does not required pin default value is INPUT
  pinMode(PushButton,INPUT);

}

void loop(){
  //For reading the pushbutton
  Value_PushButton = digitalRead(PushButton);

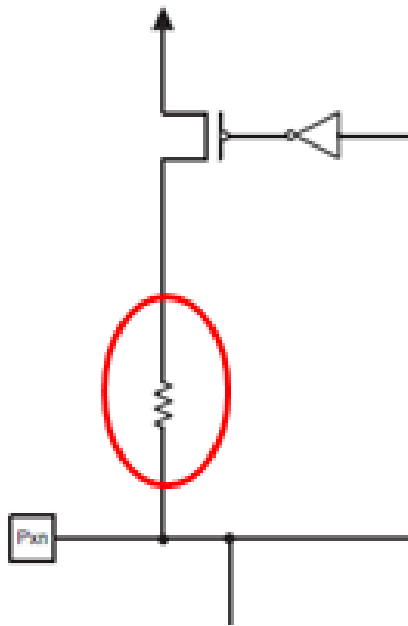
  if (Value_PushButton ==HIGH)
  {
    digitalWrite (GreenLED,HIGH); //LED on while Pushbutton is pressed
  }
  else
  {
    digitalWrite (GreenLED,LOW);
  }
}
```



## Contenido teórico: Resistencia de Pull-up interna

Los microcontroladores incluidos en las plataformas Arduino cuentan con una resistencia de pull-up interna en sus pines de entrada/salida digital. Su valor es de 20 KOhm y puede ser [activada](#) o desactivada por [software](#).

Si optamos por usar la [R de pull-up interna](#), no será necesario añadir una resistencia de pull-up en el montaje externo.



## Proyecto01\_03: Encendido de un Led encendido mediante un pulsador usando la resistencia interna de Pull-up

*Repetir el proyecto anterior de modo que el LED se encienda al pulsar el botón. Utilizar la resistencia de pull-up interna.*

### Material

- 1 Led verde
- 1 Resistencia de 220  $\Omega$
- 1 Pulsador

### Función Arduino

`pinMode(<nombre_Pin>,INPUT_PULLUP)`

El pin se configura como entrada y se habilita una resistencia de Pull-up interna de 20KOhm incluida en el microcontrolador de ATmega328.

Si se conecta un pulsador, Arduino leerá HIGH cuando el botón esté en reposo y LOW cuando se pulse. Es un comportamiento opuesto al del proyecto anterior.

<http://arduino.cc/en/Tutorial/DigitalPins#.Uxig7D95P1A>

## Código

```
const int GreenLED = 2;           // Alias for LED pin
const int PushButton = 12;        // Alias for Pushbutton pin

int Value_PushButton=0;           // Variable to store the
push value

void setup() {
  pinMode(GreenLED, OUTPUT);
  pinMode(PushButton, INPUT_PULLUP); // Set PushPin as INPUT
  and
                                   // enable the internal Pull-
                                   up //resistor
}

void loop() {

  COMPLETE THE CODE HERE!!!

}
```

## Proyecto01\_04: Encendido/Apagado de un Led mediante un SOLO pulsador. Con resistencia interna de Pull-up.

*Conserva el montaje del proyecto anterior de modo que el LED se encienda cuando pulse el botón y se apague si lo volvemos a pulsar. Utilizar la resistencia de pull-up interna.*

A continuación se introducirá como mejora que no será necesario mantener el dedo sobre el pulsador para conseguir que el LED se quede encendido. Debemos por tanto dotar al circuito de algún tipo de “memoria” que recuerde en qué estado (ON/OFF) quedó el LED la última vez que actuamos sobre el pulsador.

### Código

```
const int GreenLED = 12;    // Alias for LED pin
const int PushButton = 2;   // Number for Pushbutton pin

int Value_PushButton=0;     // Variable to store the push value

boolean StateLED=false;    // Variable to store the state of the LED

void setup() {
  pinMode(GreenLED, OUTPUT);
  pinMode(PushButton, INPUT_PULLUP);    // Set PushPin as INPUT and
                                         // enable Pull-up resistor
}

void loop(){

  Value_PushButton = digitalRead(PushButton); //Read the button

  /* Check if the input is LOW (button is pressed) and change the
  "state"*/

  if (Value_PushButton == LOW) {
    StateLED = ! StateLED;    //Toggle the value of state
  }

  if (StateLED == true)      //If Pushbutton is pressed
  {
    digitalWrite (GreenLED,HIGH); //Switch on the LED
  }
  Else
  {
    digitalWrite (GreenLED,LOW);  //Switch off the LED
  }
}
```

## Comentarios

Al probar el funcionamiento del circuito verás que su comportamiento es incierto, la luz se enciende y apaga tan rápidamente que no nos da tiempo a observarlo. A continuación se explica la causa de este problema y cómo podemos remediarlo.

## Contenido teórico: Filtrado de rebotes en pulsadores

El pulsador, de naturaleza mecánica, contiene un resorte que al ser pulsado o liberado oscila varias veces entre los estados *high* y *low* como puede verse en la figura de abajo.

El Microcontrolador funciona a una velocidad muy alta, (16 millones de instrucciones por segundo) comparado con nuestra velocidad pulsando el botón. Esto significa que mientras nuestro dedo pulsa una sola vez, Arduino puede haber realizado miles de lecturas en el pulsador, y por tanto haber cambiado el valor de la variable “state”. Como se aprecia en la figura 6, para el micro es como si el botón fuese pulsado/soltado varias veces seguidas. Luego es impredecible en qué estado quedará el LED cuando terminemos nuestra pulsación.

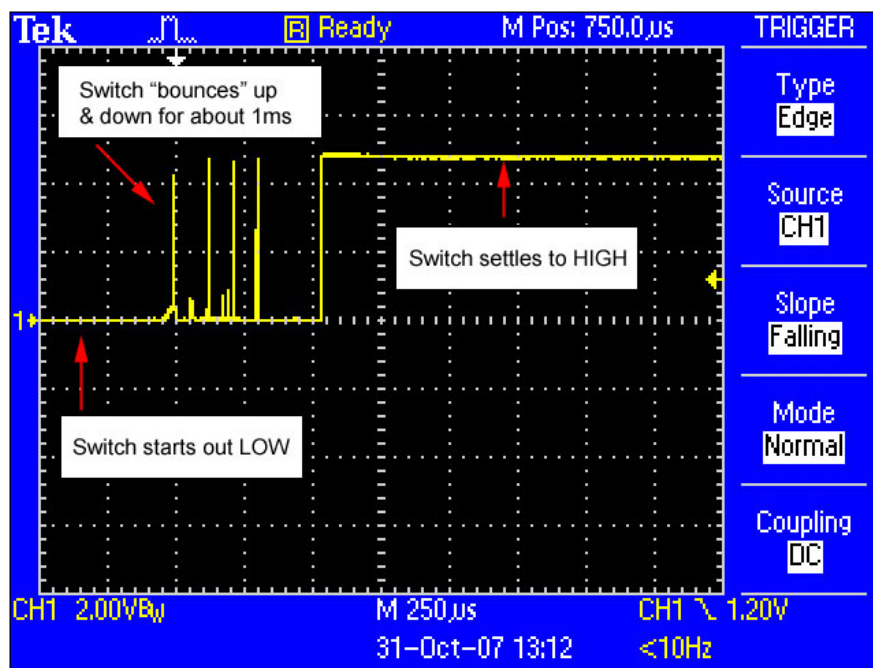


Figura 6

Para **evitar estos rebotes por software**, haremos dos lecturas seguidas del pulsador y las compararemos. Si el valor ha cambiado de HIGH (valor en reposo del pulsador) a LOW (valor al apretar) significará que se ha pulsado, diremos que se **ha detectado una transición o flanco** del pulsador. En este caso, el código ejecutará el encendido/apagado del LED. El resto de combinaciones no producirán cambios en el funcionamiento del circuito.

También existe una solución electrónica a este problema, y es montar un circuito como el de la figura 6. Vemos que se ha añadido un condensador que absorberá estos picos.

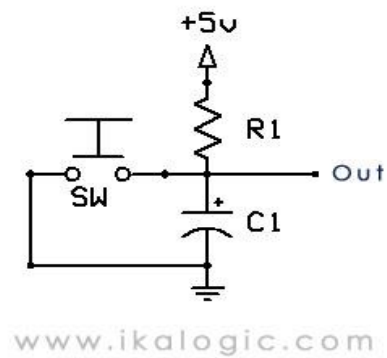


Figura 6

## Proyecto01\_05: Lectura de un pulsador, evitando rebotes, para encender y apagar un LED con un mismo pulsador.

*Realizar un montaje con un LED y un pulsador (resistencia dePull-up interna) de modo que alterne el estado (encendido/apagado) del LED cada vez que Arduino detecte que se ha pulsado el botón.*

### Código

El nuevo código es una variante del anterior.

```
const int GreenLED = 12;      // Number for LED pin
const int PushButton = 2;     // Number for Pushbutton pin

int First_Value_PushButton = 1; // It stores the FIRST reading
int Second_Value_PushButton = 0; // It stores the SECOND reading

boolean stateLED=false;       // Variable to store the push value

void setup() {
  pinMode(GreenLED, OUTPUT);
  pinMode(PushButton, INPUT_PULLUP); // Set PushPin as INPUT and
                                     // enable Pull-up resistor
}

void loop(){

  // First time, "First_ValPush" is the initialized value
  // Microcontroller reads a second value from the button
  Second_ValPush = digitalRead(PushButton);

  // Check if there was a transition (HIGH to LOW)-PRESS Button
  if ((First_Value_PushButton == HIGH) && (Second_Value_PushButton ==
LOW)) {
    stateLED = !stateLED;
  }

  // For future readings, "Second_ValPush" is now the "First_ValPush"
  First_Value_PushButton = Second_Value_PushButton;

  if (stateLED==true)           //If Pushbutton is pressed
  {
    digitalWrite (GreenLED,HIGH); //Switch on the LED
  }
  else
  {
    digitalWrite (GreenLED,LOW);  //Switch off the LED
  }
}
```

### Comentarios

Si probamos el funcionamiento del circuito se comprueba que ha mejorado. Sin embargo, aún iría mejor si hacemos que el microcontrolador “nos espere” unos milisegundos, tras detectar la transición, antes de proseguir con el resto del programa.

```
// check if there was a transition
if ((First_value_PushButton == LOW) && (Second_value_PushButton == HIGH)) {
    stateLED = !stateLED;
    delay (100);
}
```



## Contenido teórico: Definiendo “arrays”

<https://www.arduino.cc/en/Reference/Array>

Existen diferentes formas válidas de crear un vector o arrays de datos del mismo tipo.

```
//Estableciendo la longitud del array, sin inicializar
int myInts[6];

//Establece los elementos, aunque no define su longitud
int myPins[] = {2, 4, 8, 3, 6};

//Definiendo longitud y los elementos
int mySensVals[6] = {2, 4, -8, 3, 2};

//Array de caracteres, tipo char, para formar una cadena
char message[6] = "hola";
```

El primer elemento de la matriz tiene índice 0. Por lo tanto:

```
mySensVals[0] = 10;
```

Para leer el valor de un elemento del array:

```
x = mySensVals[4];
```

Las matrices se utilizan muchas veces en el interior de bucles FOR, donde el contador de bucle se utiliza como el índice de cada elemento de la matriz. Por ejemplo, podría usarse para configurar como OUTPUT varios pines de ARduino:

```
int myLeds[]={2,3,4,5,6};
int counter=0;
int timer=50;

void setup() {
  for (counter=0;counter<5;counter++){
    //En cada paso del bucle lee el elemento que marca el índice
    //“counter”

    pinMode(myLeds[counter],OUTPUT);
  }
}

void loop() {
  //En cada paso del bucle enciende y apaga un LED
  for(counter=0;counter<5;counter++){
    digitalWrite(myLeds[counter],HIGH);
    delay(timer);
    digitalWrite(myLeds[counter],LOW);
    delay(timer);
  }
}
```

## Contenido teórico: Declarando funciones

<https://www.arduino.cc/en/Reference/FunctionDeclaration>

El caso típico para la creación de una función es cuando se tiene que realizar la misma acción varias veces en un programa, por tanto, se reutiliza la misma porción de código que es lo que se llama FUNCIÓN. El uso de funciones ayuda además a dar más claridad a un código muy largo.

Hay dos funciones obligatorias en un “sketch” Arduino, `setup ()` y `loop ()`. Pueden crearse otras funciones fuera de los corchetes de esas dos funciones.

La sintaxis utilizada para crear una función puede verse en el ejemplo siguiente. Toda función tendrá un nombre y será del tipo del valor que devuelva la función (`int`, `void`...). La función puede requerir de algún valor que será usado en el cuerpo de ésta, a este valor se le llama ARGUMENTO ó PARÁMETRO. Más tarde, cuando la función sea llamada, establecerá en la misma llamada el valor del parámetro.

Tipo de función    Nombre de la función    Parámetros

```
void Animation01(int pin, int timer){  
  //Cuerpo de la función  
  digitalWrite(pin,HIGH);  
  delay(timer);  
  digitalWrite(pin,LOW);  
  delay(timer);  
}
```

Una vez creada la función, podrá ser llamada desde el código principal situado en la función LOOP.

## Proyecto01\_06: Seleccionando entre dos animaciones de LED mediante un solo pulsador.

*Monta un circuito con cuatro LEDs controlados por los pines 2, 3, 4 y 5. El siguiente código activará una secuencia de iluminación en los LEDs mientras se mantiene apretado el pulsador. En caso contrario, se mantendrán apagados.*

```
int myLeds[]={2,3,4,5};
int counter=0;
const int PushButton =8;
int value_PushButton =0;

void setup(){
  pinMode(Button1, INPUT_PULLUP);
  for (counter=0;counter<5;counter++){
    pinMode(myLeds[counter],OUTPUT);
  }
}

void loop(){
  //Read the pushbutton
  value_PushButton = digitalRead(PushButton);

  //
  if (value_PushButton ==LOW ){
    {
      Animation01(100);
    }
  }
  else
  {
    SwitchOFF();
  }
}

//FUNCTIONS//

int Animation01(int timer){
  //Switch ON or OFF a LED in each iteration
  //from the LED2 to LED5
  for(int counter=0;counter<4;counter++){
    digitalWrite(myLeds[counter],HIGH);
    delay(timer);
    digitalWrite(myLeds[counter],LOW);
    delay(timer);
  }
}

int SwitchOFF(){
  //Apaga todos los leds
  for(int counter=0;counter<4;counter++){
    digitalWrite(myLeds[counter],LOW);
  }
}
```

### 3. Ejercicios

- 1.1. Al iniciarse el programa un LED deberá estar encendido, al pulsar el pulsador, el LED se apagará y se mantendrá apagado durante un tiempo volviendo luego a encenderse.
- 1.2 Monta 4 LEDs para conseguir hacer con ellos una animación de modo que las luces parezcan ir de derecha a izquierda y luego al revés. Incluye un parámetro que permita modificar fácilmente el tiempo que permanece encendido cada LED.