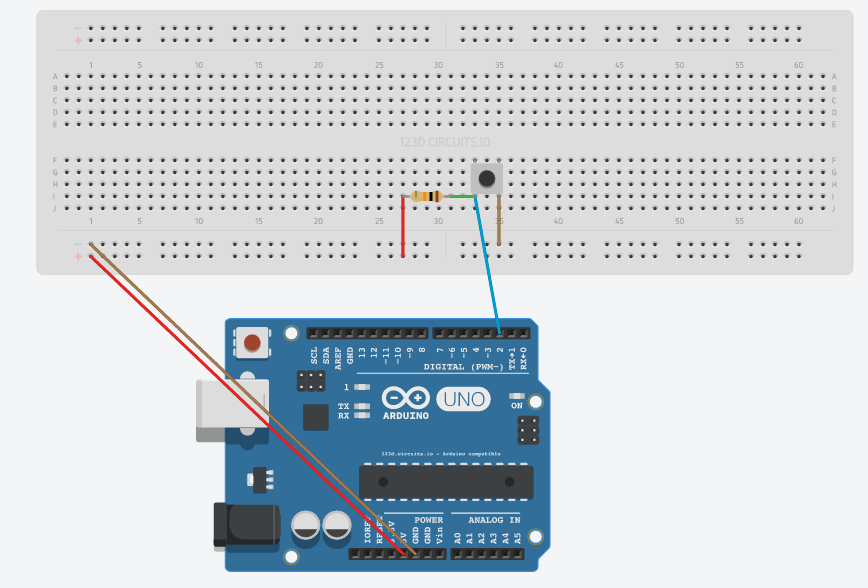
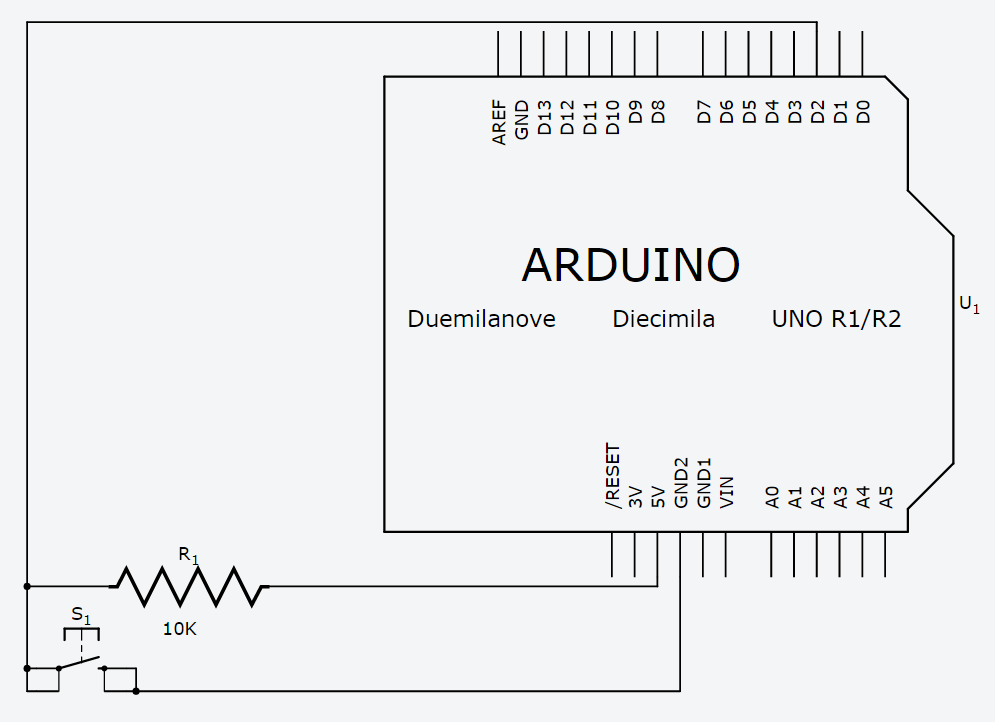
MATERIAL REQUERIDO.

|  |  |
| --- | --- |
| [Imagen de Arduino UNO](http://www.prometec.net/producto/arduino-uno/) | [**Arduino UNO o equivalente**](http://www.prometec.net/categoria-producto/arduinos/). |
| [Breadboard](http://www.prometec.net/wp-content/uploads/2014/09/Img_3_4.png)[conexiones](http://www.prometec.net/wp-content/uploads/2014/09/Img_3_6.png) | Una [Protoboard](http://www.prometec.net/producto/protoboard-830/)más [cables](http://www.prometec.net/producto/cables-dupont-macho-macho/). |
| [componente](http://www.prometec.net/wp-content/uploads/2014/09/Img_3_5.png) | Una [resistencia](http://www.prometec.net/producto/kit-de-resistencias-variadas/)de 330Ω. |
| [boton contactor](http://www.prometec.net/wp-content/uploads/2014/09/Img_5_1.png) | Un pulsador. |





/\*

Button

Turns on and off a light emitting diode(LED) connected to digital

pin 13, when pressing a pushbutton attached to pin 2.

The circuit:

\* LED attached from pin 13 to ground

\* pushbutton attached to pin 2 from +5V

\* 10K resistor attached to pin 2 from ground

\* Note: on most Arduinos there is already an LED on the board

attached to pin 13.

created 2005

by DojoDave <http://www.0j0.org>

modified 30 Aug 2011

by Tom Igoe

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/Button

\*/

// constants won't change. They're used here to

// set pin numbers:

const int buttonPin = 2; // the number of the pushbutton pin

const int ledPin = 13; // the number of the LED pin

// variables will change:

int buttonState = 0; // variable for reading the pushbutton status

void setup() {

// initialize the LED pin as an output:

pinMode(ledPin, OUTPUT);

// initialize the pushbutton pin as an input:

pinMode(buttonPin, INPUT);

}

void loop() {

// read the state of the pushbutton value:

buttonState = digitalRead(buttonPin);

// check if the pushbutton is pressed.

// if it is, the buttonState is HIGH:

if (buttonState == HIGH) {

// turn LED on:

digitalWrite(ledPin, HIGH);

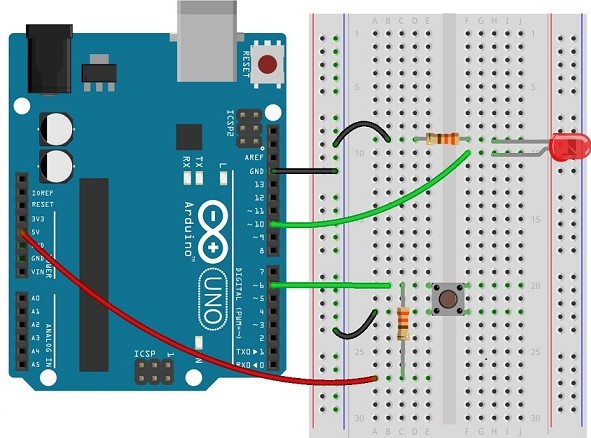
} else {

// turn LED off:

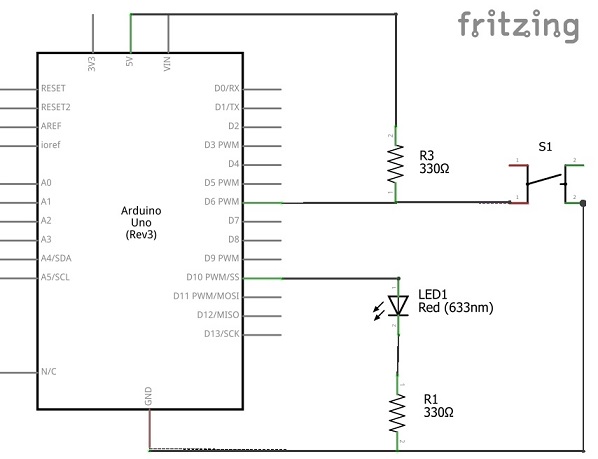
digitalWrite(ledPin, LOW);

}

}



* *En este esquema hemos seguido la práctica habitual de usar cables negros para conectar a masa y cables rojos para conectar a tensión (5V).*
* *Obsérvese que el pulsador S1 tiene cuatro pines (el que está sobre la resistencia horizontal). Esto es porque cada entrada del interruptor tiene dos pines conectados. En nuestro circuito simplemente ignoramos los pines secundarios.*



Obsérvese que mientras no pulsemos S1 el pin 6 de Arduino está conectado a 5V a través de la resistencia R3 forzando una lectura de tensión alta (HIGH). En cambio cuando pulsemos S1 cerraremos el circuito del pin 6 a Ground con lo que leerá tensión baja, LOW. En ambos casos tenemos un valor de tensión definido.

Si no pusiéramos la resistencia R3, al pulsar S1 leeríamos correctamente LOW en el pin 6. Pero al dejar de pulsar S1 el pin 6 estaría en un estado flotante, que es ni HIGH ni LOW  sino indeterminado. Como esto es inaceptable en circuitos digitales forzamos una lectura alta con R3.

* *A esta resistencia que fuerza el valor alto en vacio se le conoce como****pullup****Si la conectáramos a masa para forzar una lectura a Ground se le llamaría****pulldown****resistor.*
* *Esta resistencia es clave para que las lecturas del pulsador sean consistentes. El circuito, simplemente, no funcionará bien si se omite (volveremos sobre esto)..*

la resistencia no sólo sirve para definir el estado lógico, sino también porque si no estuviera, al cerrar el pulsador estaríamos llevando 5V a GND y haríamos cortocircuito en la placa.int LED = 10 ;

int LED\_INTERNO = 13 ;

int boton = 2;

void setup()

{

pinMode( LED, OUTPUT) ; // LED como salida

pinMode( LED\_INTERNO, OUTPUT) ; // LED como salida

pinMode( boton , INPUT) ; //botón como entrada

}

void loop()

{

int valor = digitalRead(boton) ; // leemos el valor de boton en valor

digitalWrite( LED, valor) ;

digitalWrite( LED\_INTERNO, valor) ;

}