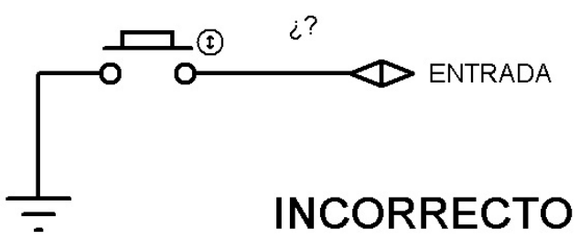
Resistencias Pull-Up y Pull-Down

Cuando en nuestros proyectos tenemos la necesidad de utilizar entradas digitales debemos tener presentes términos como resistencias pullup o pulldown. Los pulsadores o push buttons son un tipo de entrada digital muy utilizados, su función es cambiar de estado (abierto/cerrado) cuando es presionado. Con este cambio de estado podemos hacer que en la entrada del microcontrolador o Arduino tengamos 0V o 5V, pero para eso es necesario agregar además una resistencia en configuración pull-up o pull-down de acuerdo a nuestro criterio. Estas configuraciones son necesarias pues de otra manera el Arduino no sería capaz de distinguir correctamente el voltaje en la entrada, esto porque cuando el pulsador está abierto, el voltaje en la entrada no está determinado y puede ser leído como un 0 o un 1. Las resistencias pull-up y pull-down nos permiten establecer voltajes de reposo para cuando el pulsador no está presionado y asegurar una correcta lectura.

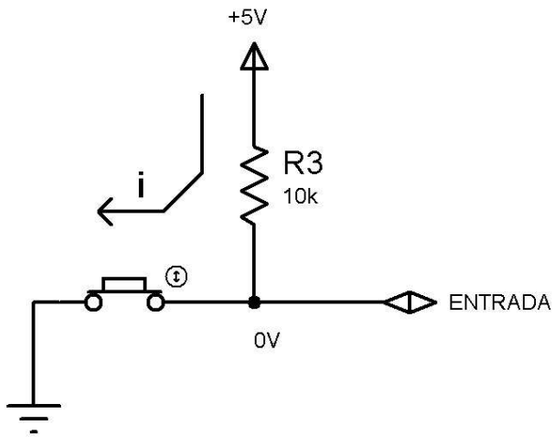


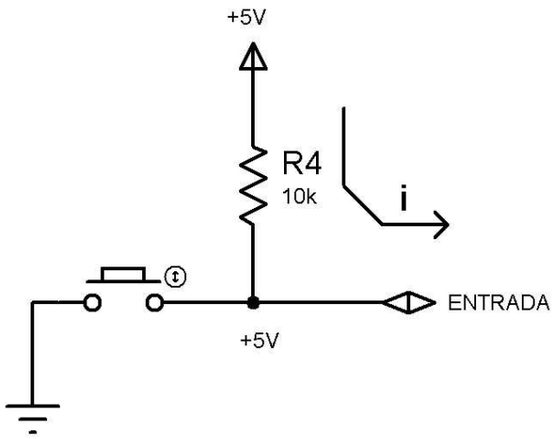
Resistencia Pull-Up

Como su nombre indica esta resistencia tiene la función de “jalar” hacia “arriba”, lo que significa que polariza el voltaje hacia el voltaje de fuente (VDD) que puede ser +5V o +3.3V. De esta forma cuando el pulsador está abierto o en reposo, el voltaje en la entrada del Arduino siempre será de +5V. Las entradas del Arduino son de alta impedancia lo que significa que la corriente que circulará por esa línea sea mínima en el orden de los micro-amperios, por lo que el voltaje que “cae” en la resistencia pull-up es mínimo y tenemos casi el mismo voltaje de fuente en la entrada del Arduino.

Cuando el pulsador es presionado, la corriente circula por la resistencia y luego por el pulsador, de esta forma tenemos que el voltaje en la entrada del Arduino es Tierra o 0V.

Entonces en la configuración pull-up cuando el pulsador está en reposo el Arduino lee 1 y cuando presionamos leerá 0.

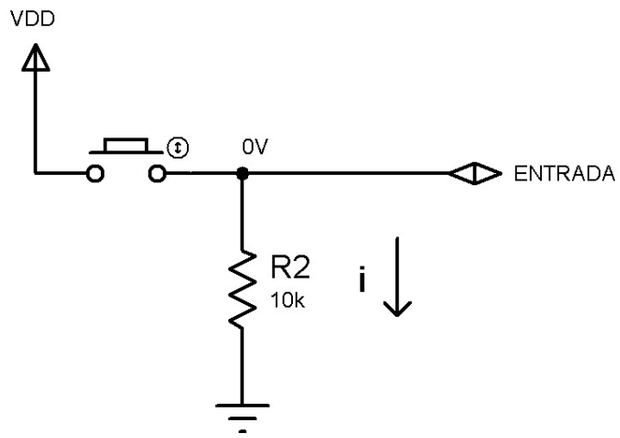


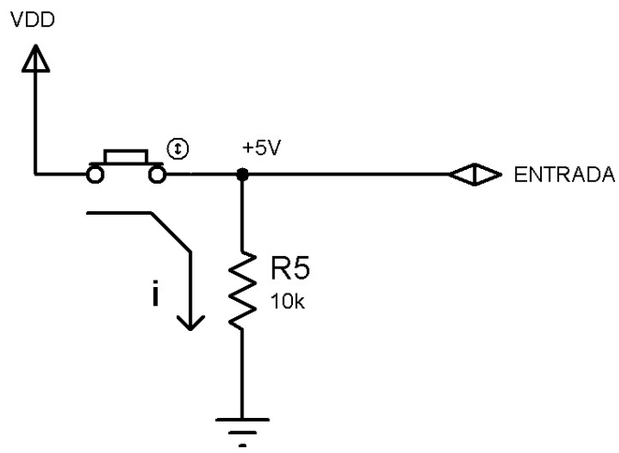


Resistencia Pull-Down

De forma similar la resistencia pull-down “jala” el voltaje hacia “abajo” o “0V”. Cuando el pulsador está en reposo, el voltaje en la entrada del Arduino será 0V. Cuando presionamos el pulsador la corriente fluye de +5V por el pulsador hacia la resistencia y termina en 0V, de esa forma tenemos +5V en la entrada del Arduino.

Entonces en la configuración pull-down cuando el pulsador está en reposo el Arduino lee 0 y cuando presionamos leerá 1.





¿Cuál configuración debo usar?

Esto depende del criterio del diseñador, no existen mayores diferencias técnicas. La configuración más popular es la de tipo pull-up, debido a que muchos microcontroladores incluido Arduino poseen resistencias pull-up internas, las cuales pueden ser activadas utilizando un comando en el programa del microcontrolador y así ahorrar el uso de resistencias. Se debe tener en cuenta que para la configuración pull-up se lee “0” cuando el pulsador es presionado.

¿Qué valor de resistencia debo utilizar?

Podemos utilizar resistencias con valores entre 1K y 10K, esto dependerá de algunos factores como la frecuencia de variación, longitud del cable. Sin embargo hay que destacar que cuanto mayor sea la resistencia para el pull-up, más lento es el pin en responder a los cambios de voltaje. Esto es debido a que el sistema que alimenta el pin de entrada es esencialmente un condensador junto con la resistencia pull-up, formando de esta manera un filtro RC, y filtros RC tardan tiempo para cargarse y descargarse. Si tienes una señal de cambio muy rápido (como USB), un alto valor de resistencia de pull-up puede limitar la velocidad a la que el pin puede cambiar de estado. Por lo que es más común encontrar valores de pull-up entre 1KΩ y 4.7KΩ.

Activar Pull-up interno en Arduino

Muchos microcontroladores incluyen resistencias pull-up internas, las cuales se puede activar mediante instrucciones en el programa.Para el caso de Arduino solo se tiene declarar al momento del setup.

Esta técnica es muy utilizada tanto para conectar pulsadores como para circuitos I2C.

Definición de los modos de los pines digitales: INPUT, INPUT_PULLUP, y OUTPUT

Los pines digitales de Arduino se pueden usar como una de las tres formas siguientes:

INPUT **INPUT_PULLUP** **OUTPUT**

Cuando se cambia un pin **con pinMode ()**, cambia el comportamiento eléctrico del pin.

Pines configurados como INPUT

Los pines de Arduino (Atmega) configurados como INPUT con pinMode () se dice que están en estado de alta impedancia. Los pines configurados como INPUT demandan una corriente extremadamente baja del circuito que esta muestreando, equivalente a una resistencia en serie de 100 Megohms. Esto los hace útiles para la lectura de un sensor.

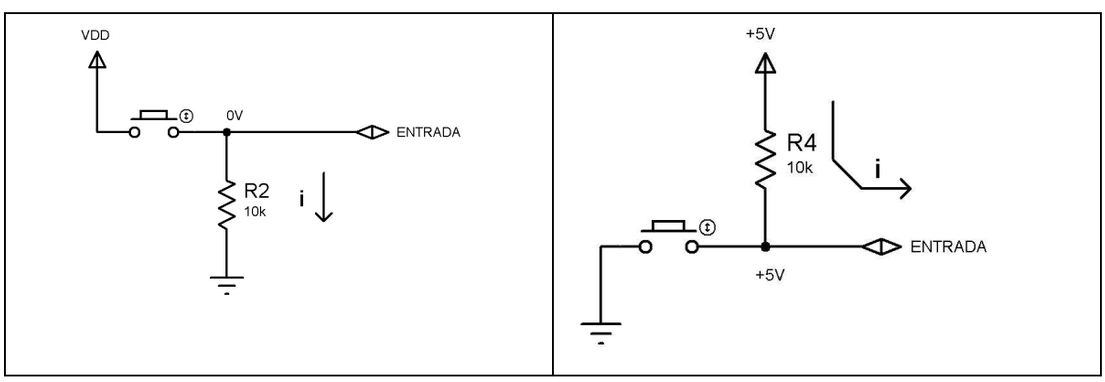
Si tenemos un pin configurado como INPUT y estamos leyendo un interruptor, cuando el interruptor está en estado abierto la entrada del pin está **"flotante"**, dando como resultado una lectura impredecible.

Para asegurar una lectura apropiada del pin cuando el interruptor está abierto, debe usarse una resistencia **pullup o pulldow**. La función de estas resistencias es llevar el pin a un estado conocido cuando el interruptor está abierto. Usualmente se elige una resistencia de 10 K ohm, ya que es un valor suficientemente bajo como para evitar que una entrada quede "flotante", y al mismo tiempo un valor suficientemente alto como para no drenar demasiada corriente cuando el interruptor está cerrado.

Si se usa una resistencia pulldown, el pin de entrada estará a nivel LOW cuando el interruptor esté abierto y en estado HOGH cuando el interruptor esté abierto.

Si se usa una resistencia pullup, el pin estará a nivel HIGH cuando el interruptor esté abierto y LOW cuando el interruptor esté cerrado.

Lo dicho anterior es para resistencias agregadas externamente.



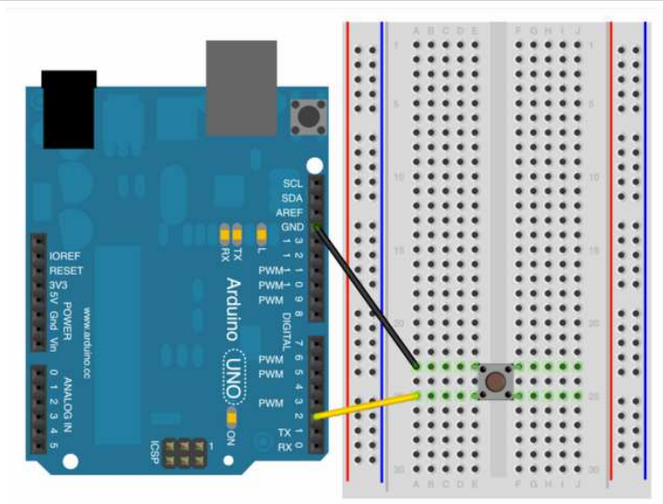
Pines configurados como INPUT_PULLUP

El microcontrolador Atmega del Arduino tiene resistencias pull-up internas (resistencias conectadas internamente a la alimentación) a las que podemos acceder. Si prefieres usar estas en lugar de resistencias pull-up externas, puedes usar el argumento INPUT_PULLUP en pinMode ()

pinMode(3, INPUT_PULLUP);

Los pines configurados como INPUT o INPUT_PULLUP se pueden dañar o destruir si se conectan a tensiones más bajas que la masa (tensiones negativas) o por encima de la tensión de alimentación (5 V o 3,3 V).

Esto nos permitirá hacer:



Donde se ha colocado un pulsador sin resistencia externa de PULLUP, ya que se usa la interna de Arduino.

MI RECOMENDACIÓN ES USAR RESISTENCIAS EXTERNAS SIEMPRE

Pines configurados como Outputs

Los pines configurados como OUTPUT con pinMode () se dice que están en estado de baja impedancia. Esto significa que pueden proporcionar una cantidad sustancial de corriente a otros circuitos. Los pines del Atmega pueden ser source (fuente, proporcionar corriente) o sink (drenador, absorver corriente) hasta 40 mA (milliamperios) a otros dispositivos o circuitos. Esto hace que sean útiles para encender LEDs porque los LEDs típicamente consumen menos de 40 mA. Carga mayores de 40 mA (por ejemplo, motores) requieren un transistor u otra circuitería de interface.

Los pines configurados como salidas se pueden dañar o destruir se se conectan directamente a masa o al positivo de la alimentación.

Definición de los LED_BUILTIN (LED incorporados)

Muchas placas Arduino tienen un pin conectado a un LED incluido en la placa conectado con una resistencia en serie. La constante **LED_BUILTIN** es el número de pin al que el LED incluido está conectado en la placa. La mayoría de las placas tiene este LED conectado al pin digital 13.

Ejemplo de programa donde se utiliza lo visto:

```
/*
 * Prueba de KY-003 Módulo de
 * Sensor Magnético por efecto Hall
*/
int sensor = 2 ; // define como entrada del sensor KY-003 el pin digital 2
int estado ; // define una variable para ingresar el estado

void setup () {

// define el pin sensor como entrada con resistor pull-up

  pinMode(sensor, INPUT_PULLUP);

// define el pin del LED del Arduino como salida

  pinMode(LED_BUILTIN, OUTPUT);

}

void loop () {
  estado = digitalRead(sensor) ; // lee el estado del sensor
// cuando el sensor Hall detecta un campo magnetico
// el LED del Arduino se activa
  if (estado == LOW)
  {
    digitalWrite(LED_BUILTIN, HIGH);
  }
  {
    digitalWrite(LED_BUILTIN, LOW);
  }
}
```