Botones y llaves

Dentro de la electrónica la principal forma que tenemos de comunicarnos con nuestros dispositivos es atreves de botones y llaves ya sea digitales o físicos, de hecho, de ahí viene todos nuestros sistemas digitales siendo los dos estados que se pueden conseguir con estos, prendido y apagado. Obviamente con el tiempo esto ha ido evolucionando, pero sigue compartiendo la misma base de siempre, actualmente lo que antes hacían las llaves para complejos sistemas ahora lo hacen transistores automáticamente en el caso de electrónica aplicada.

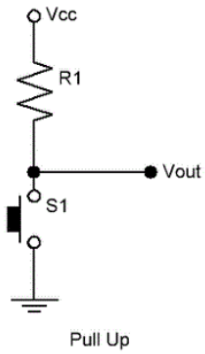
Pero como podemos manipularnos con nuestros microcontroladores, bueno pese a ser algo sencillo tiene una pequeña ciencia por detrás:

Modos de conexión

Sabemos que al presionar un botón pasara de un estado a otro, pero que estado enviara a la hora de hacer la conexión, ¿un valor positivo (I) o un valor negativo (0)?

Bueno todo eso dependerá si lo conectamos como pullup o pulldown, ¿y esto en qué consiste?, bueno:

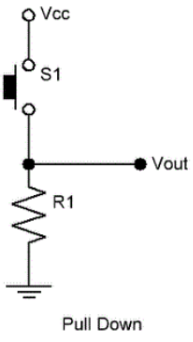
PullUp



al hacer una conexión con resistencia PullUp la conexión con nuestro pin (Vout) siempre estará conectada a el valor **positivo** que tengamos, en caso de que se presione el botón la corriente circulara por la resistencia hacia el GND y tendrá una caída de tensión total, por lo tanto, en este caso el pin recibirá un valor **negativo**

Se podría decir que esta normalmente positivo

PullDown

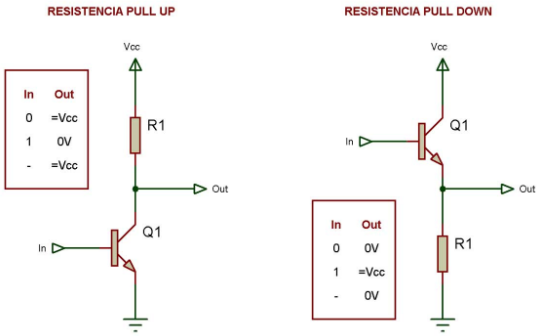


En este caso ocurre todo lo contrario, nuestro pin siempre estará conectado al GND, en el caso de que se presione el botón la corriente circulara por la resistencia y se producirá su debía caída de tensión, pero al estar el pin conectado antes de la resistencia este se quedara con el VCC (recordar circuitos en paralelo)

Se podría decir que esta normalmente negativo

Cabe aclarar que en ambos casos esta resistencia debe ser bastante grande, con valores mayores a 1kΩ estaríamos listos.

Esto es aplicado para cualquier sistema que cambie de estados, como se dijo anteriormente los transistores

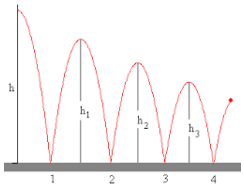


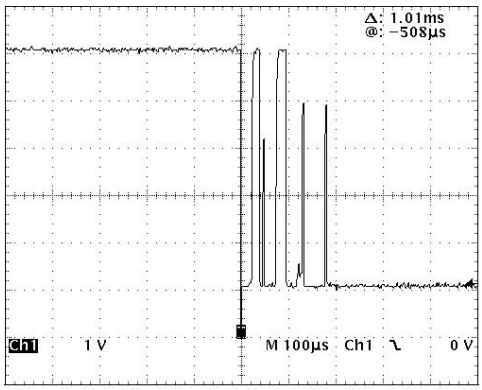
Pero en este caso hay que tener en cuenta otras cuestiones propias de cada transistor que no nos incumbe en este momento.

Física interna

Cuando digo lo de física interna no hago referencia a que vamos a ver sus mecanismos si no una parte interna de estos sistemas que nos incumbe, el cual es el rebote

Este se puede a deber por imperfecciones propias de los botones o el coeficiente de restitución el cual genera que cada vez que presionamos el botón pueda haber micro rebotes del orden de microsegundos que hagan que nuestro pulso no sea un solo movimiento si no varios:

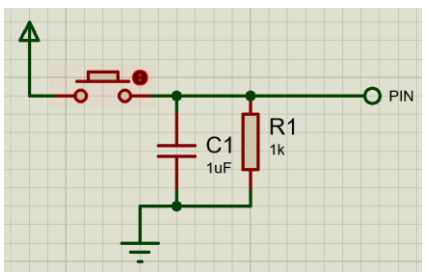




Esta sería la señal típica a la hora de presionar un botón, aunque puede haber botones de mejores calidades o peores en un marco general se vera así, como pueden ver tiene una conexión pulldown y a la hora de presionar el botón tiene pequeños rebotes que si los leyéramos en vez de parecer que presiono el botón 1 vez, lo hiciera 5 veces distintas (buena calidad del boton), hay 2 formas de evitar esto, físicamente o por código:

Físicamente (al pedo)

Es en verdad la forma más fácil, aunque menos precisa de hacerlo, y es conectar un pequeño capacitor cerámico el cual al conectarlo de esta forma amortiguara las señales de ruido producidas por el rebote, no la recomiendo mucho pero siempre es bienvenida.



Código

Tenemos dos modos al hacer esto, modo berreta, y modo piola.

En el primer caso lo podemos hacer usando un simple delay de 1 a 5 ms dependiendo de nuestro botón, esto garantizara que el tiempo en que se produce nuestro rebote el código este parado. No me gusta mucho esta opción dado que nos afecta a la libre continuación de nuestro programa, a menos de que sea algo muy simple lo que hacemos siempre es preferible evitar usar el uso de delay en todos los casos.

```
if(digitalRead(boton)){
  //algo
}
delay(2);
```

El otro caso es usar mediante un millis()

```
if(digitalRead(boton) && millis() - time < resolucion){
  time = millis();
  //algo
}
```

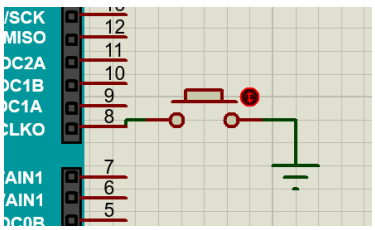
Lo que hará que a la hora de presionar el botón no vuelva a repetir una acción hasta que pase el tiempo que le pongamos (resolución), lo bueno es que hacer esto nos abre un abanico de ajustes que queramos hacerle cambiante el valor resolución y agregando otros parámetros al if **(importante recordar lo que vimos antes con el manejo de millis())**

Arduino

Todo esto que vimos es aplicable a un caso general, pero que opciones tenemos con Arduino.

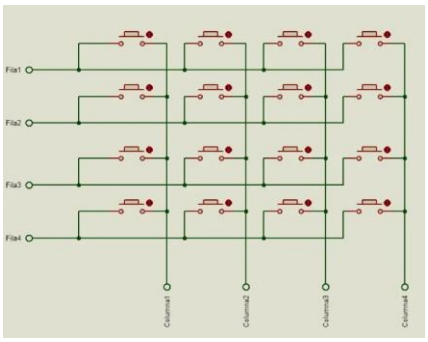
Primero que el sistema de conexión se nos es mucho más sencilla con Arduino al usar el modo **INPUT_PULLUP** dentro de la configuración de pinMode(), esto nos permite que el Arduino active la entrada de tipo pullUp facilitando la conexión ya que solo debemos conectar los botes por un lado al GND y por el otro a la entrada evitando tener que soldar resistencias, eso si demos tener en cuenta que normalmente esta siempre en estado alto.

```
void setup() {
  pinMode(boton, INPUT_PULLUP)
}

void loop() {

  if(!digitalRead(boton) && millis() - time < resolucion){
    time = millis();
    //si se presiona el boton se hara la accion que este aqui
  }
```



Ver como coloco el ! antes del digitalRead() para que si se presione el boton se active el if

Aparte de esto también tenemos matrices las cuales son un conjunto de botones los cuales al ser demasiados se usa un sistema de matrices que reduce la cantidad de pines usados a $2\sqrt{n}$ pines



Como todo sistema de matrices este se maneja por iteraciones en filas y lectura de sus columnas o viceversa, actualmente existen varias librerías para el manejo de estas según nuestras necesidades.

También para estos casos se suele usar mucho el PCF8574 dado que amplía la cantidad de pines que podemos utilizar y solo los tenemos que manejar por dos cables del I2C, recomiendo dos librerías:

- Keypad
- Keypad_I2C

Ambos hechos por Alexander Brevig

Interrupciones

Sabemos que el uso más común de interrupciones en Arduino es para el uso de pulsadores, en caso de necesitar algo que funcione inmediatamente debemos saber que siempre podemos contar con el uso de interrupciones las cuales se encuentras en el pin 2 y 3 del Arduino basado en él ATMEGA328P, en otros modelos esto puede variar

Librerías

Para el simple manejo de pulsadores como todo en Arduino sin necesidad de tener que usar matrices igual se puede usar librerías que nos expanden las posibilidades a la hora de manejarlos, aunque personalmente recomiendo hacer en este caso todo por nuestra cuenta ya que es algo simple que nos aporta un montón de experiencia, pero igual acá dejo esta librería:

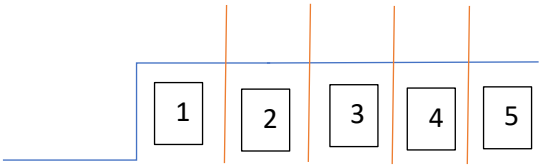
- GFButton

Como usara puede verlo aqui y algunos ejemplos aqui

Modos de uso:

Ya para ir cerrando debemos tener en cuenta que para un pulsador se puede manejar de varias formas entre ellas, por ejemplo, para un pulso seguido

En este caso hay dos opciones, una que cuente como un solo pulso hasta que se deje de presionar el botón, o que según una cantidad de tiempo x cuente como varios pulsos:



Esto dependerá de nuestras necesidades y como lo programemos.