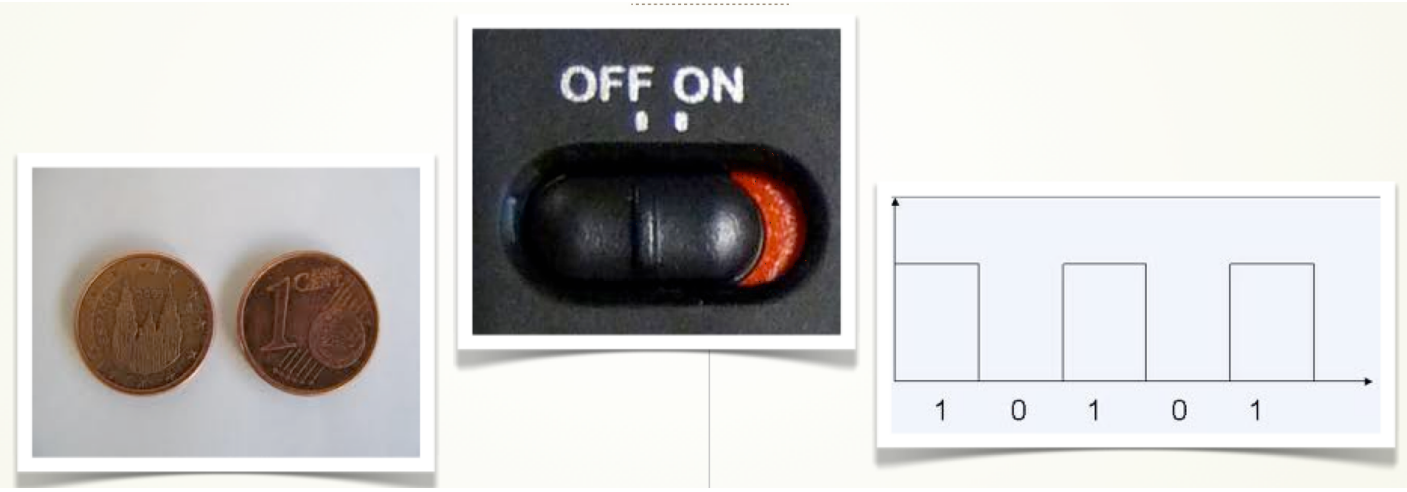
# Conexiones digitales




En este apartado aprenderemos el funcionamiento básico de las entradas y salidas digitales de la placa Arduino. Si observamos bien la placa, vemos que hay 13 pines digitales.

En este caso la señal digital no es más que un valor discreto de entre dos posibles, que si en rigor se asocian a tensiones, nosotros por sencillez los asociaremos a dos valores que serán ,apagado o encendido o lo que es lo mismo  LOW ó HIGH.

Así si asignamos un 0 al pin digital 4, es lo mismo que decir que ese pin, o mejor dicho, lo que esté conectado a ese pin estará apagado si le asignamos un 1, estamos diciendo que estará encendido.

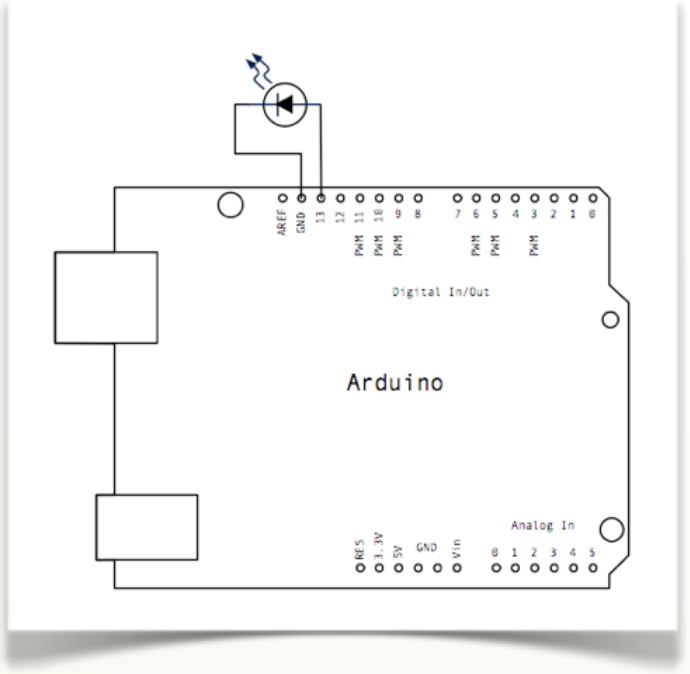
Entonces, ¿Con los 13 pines digitales de Arduino , podríamos actuar para controlar 13 bombillas? . Si, aunque Arduino es aún más potente ya que aún podemos usar los 5 pines analógicos también como salidas digitales. Veamos como.

## Montaje 1: LED parpadeante sin EDUBASICA

Vamos a controlar el encendido y apagado de un led conectado al pin13 de Arduino.

**¿Por qué el pin13 y no otro?** Podríamos hacerlo con otro, pero el pin13 tiene asociado un led en la placa justo debajo de el y así nos evitamos tener que montar. Si pusiéramos un pin polarizado correctamente entre el pin13 y GND también funcionaría. El pin13 tiene también una resistencia que hace posible conectarle un led directamente, si hacemos el montaje con otro pin debemos añadir esta resistencia de 10Kohm entre el led y el pin.

> Acuérdate: La pata más larga del LED es el (+) por lo tanto en el D13 y el corto (-) en GND.




https://create.arduino.cc/editor/javierquintana/27aca596-db6d-4a41-a100-faf60cefa56e/preview

https://create.arduino.cc/editor/javierquintana/27aca596-db6d-4a41-a100-faf60cefa56e/preview?embed

Todo lo que está entre las llaves de **loop()**, se ejecuta indefinidamente. Así vemos un efecto de led parpadeante ya que si analizamos las líneas del código vemos que el proceso es:

- Encendemos.
- Esperamos un segundo.
- Apagamos.
- Esperamos un segundo.

¡Atrevámonos y cambiemos los tiempos de parada!

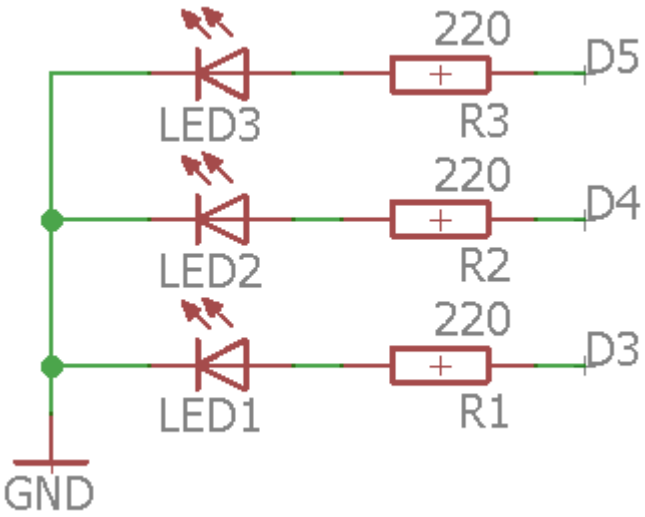
https://www.youtube.com/embed/EFFSLvIF9rY

## Montaje 2: LED EDUBASICA parpadeante

Igual que en el caso anterior, pero vamos a utilizar un LED de la shield de Edubásica, tenemos tres opciones:

- LED VERDE pin 3
- LED AMARILLO pin 4

- LED ROJO pin 5




El programa es igual que el anterior, pero cambiando el número del pin:

https://create.arduino.cc/editor/javierquintana/2c75c843-72c9-4dc4-b01a-410e318f1080/preview

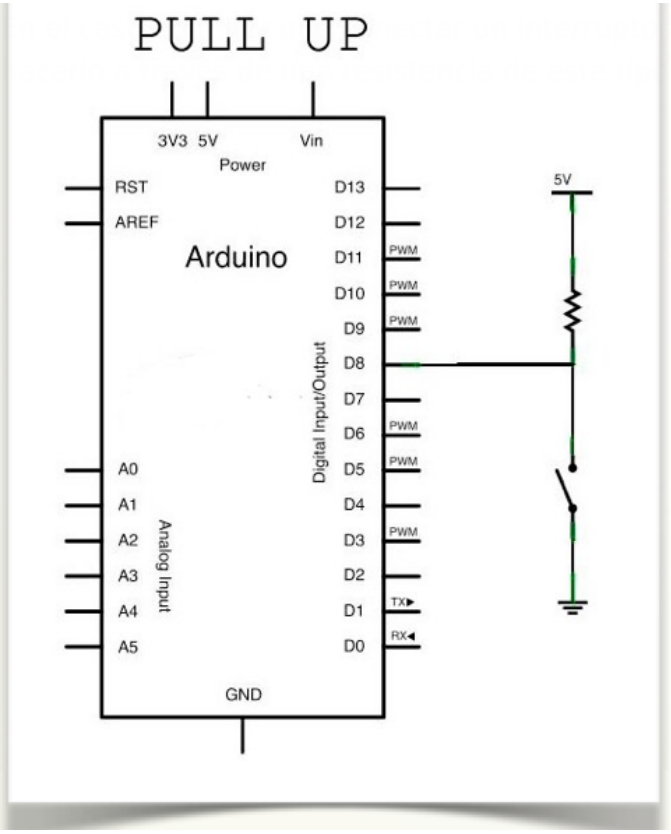
https://create.arduino.cc/editor/javierquintana/2c75c843-72c9-4dc4-b01a-410e318f1080/preview?embed

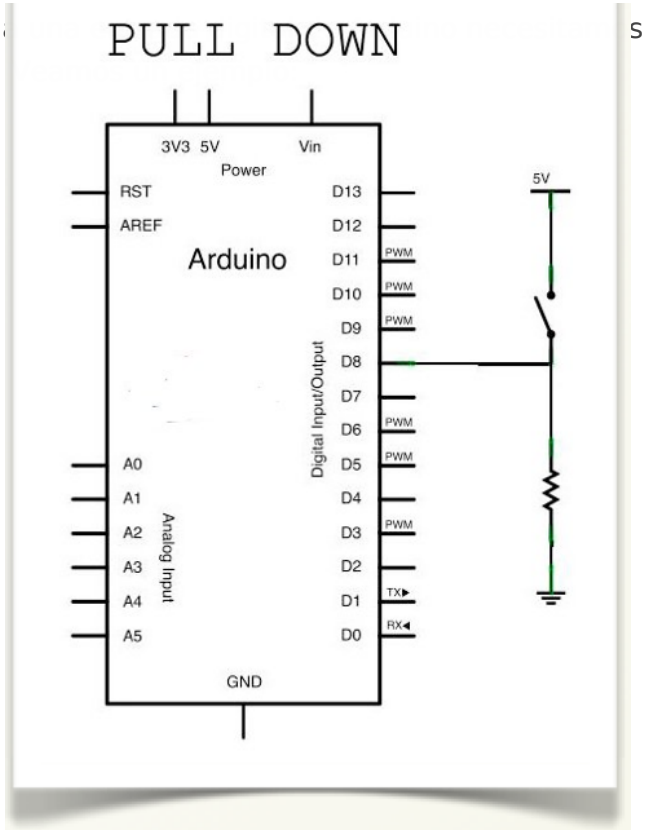
https://www.youtube.com/embed/UHttCda49Vo?rel=0

# Resistencias pull-up y pull-down

En los proyectos con dispositivos digitales, caso de la placa Arduino, reciben señales de entradas digitales del exterior. Estas señales externas sirven para activar o desactivar un circuito, recibir información del estado de un sensor,...

Las resistencias “pull-up” y “pull-down” son resistencias que se ponen en las entradas digitales para fijar un valor por defecto, nivel alto (“1”) o nivel bajo (“0”), cuando no se detecta ningún valor. Esto ocurre cuando la entrada no está conectada a nada.

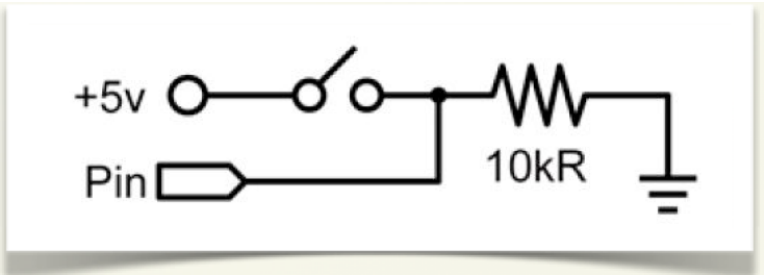



La resistencia **“pull-up”** establece un nivel alto (1) en la entrada digital en el estado de reposo del circuito. Un circuito con una entrada “pull-up” sería el siguiente.

La resistencia **“pull-down”** establece un nivel bajo (0) en la entrada digital en el estado de reposo del circuito. Este tipo de circuito es el más empleado en las entradas digitales para evitar lecturas erróneas debido a ruidos externos y consumo de energía. La resistencia suele ser de 10 kΩ y el circuito con esta configuración sería el siguiente.

Un ejemplo de circuito **“pull-down”** lo tenemos en la placa EduBásica en el pin digital D2, preparado para configurarlo como entrada, tiene conectado un pulsador y una resistencia pull-down. El esquema del circuito es el siguiente




El funcionamiento de este circuito que está conectado al pin digital D2 como entrada es detectar si el pulsador está pulsado o no.

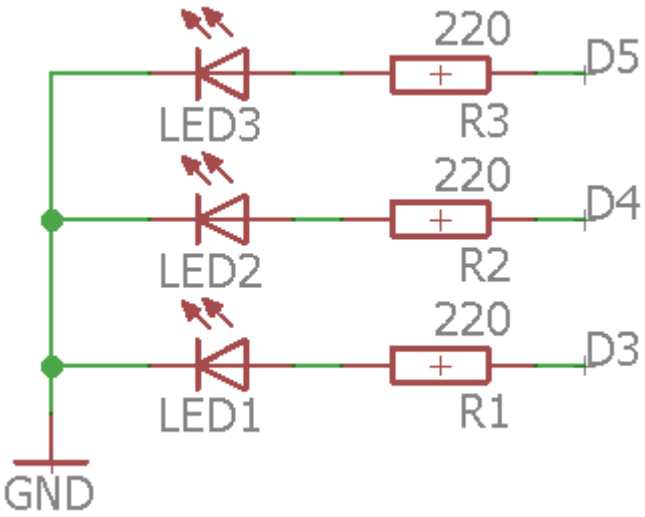
- Si el pulsador no está pulsado en Pin, que está conectado a D2, tenemos 0V por no pasar corriente entre el pin D2 y masa. Por tanto, corresponde a un nivel bajo o “0” lógico.
- Si el pulsador esta pulsado en el pin D2 tenemos 5V, que corresponde a un nivel alto o “1” lógico.
- Si en el anterior circuito no se pone la resistencia de 10KΩ y el pulsador está abierto con el propósito de tener un nivel bajo porque no hay tensión, puede ocurrir y de manera aleatoria que el pin D2 lea un nivel alto por alguna interferencia producida por motores eléctricos, bobinas de un relé u otro dispositivo de nuestro proyecto.

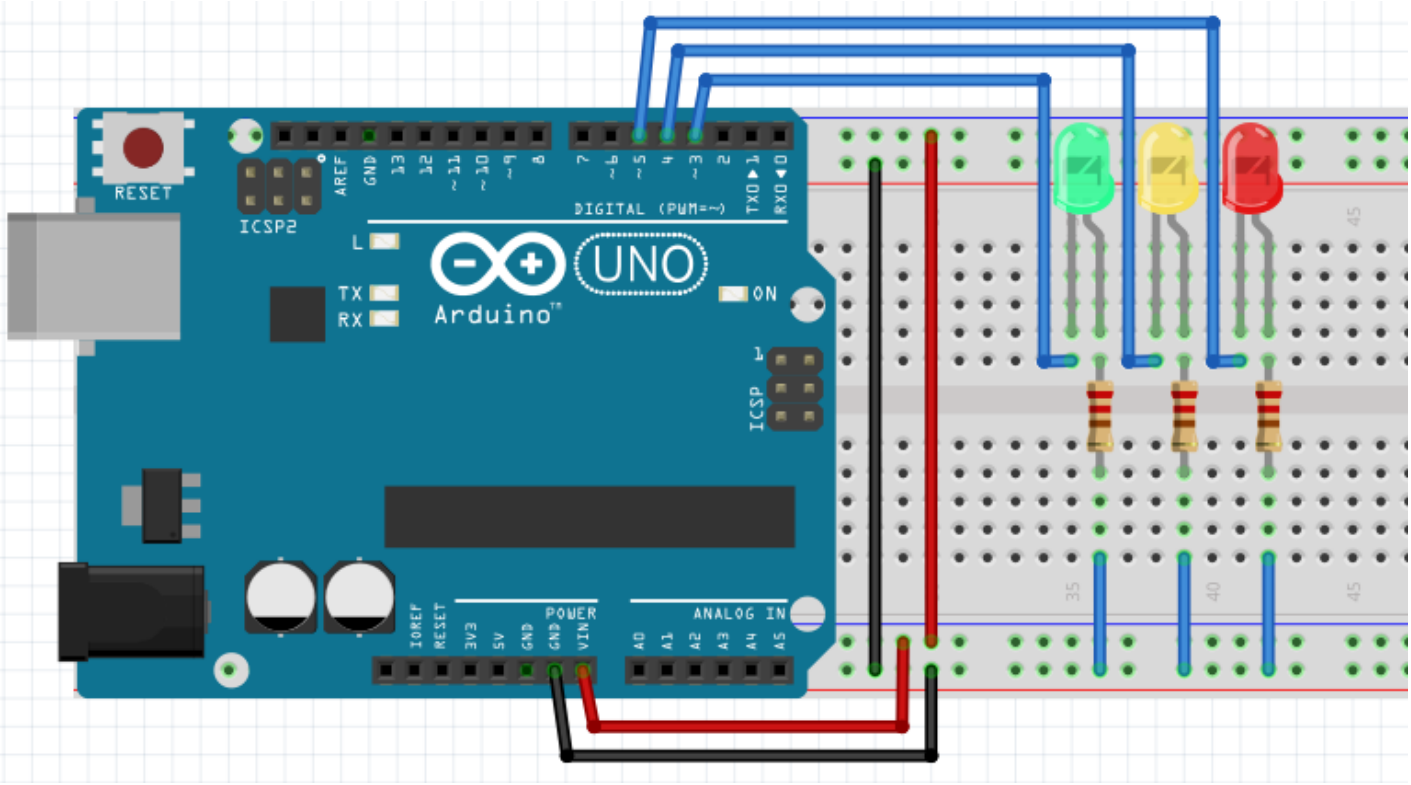
## Montaje 3: SEMAFORO CON EDUBASICA

Montaremos un semáforo con los tres leds de la EduBásica. Aquí es muy fácil, no hay que hacer ningún montaje.

## Montaje 3: SEMÁFORO SIN EDUBASICA

La EduBásica es opcional y podemos montar el circuito correspondiente con una protoboard, pero EduBásica nos ahorra trabajo. Necesitamos añadir una resistencia entre el pin y el led, para evitar que el led se funda.






en este caso tienes libertad de utilizar D3 D4 D5 o otros que quieras.

## Visualización de datos por el puerto serie. Serial.print

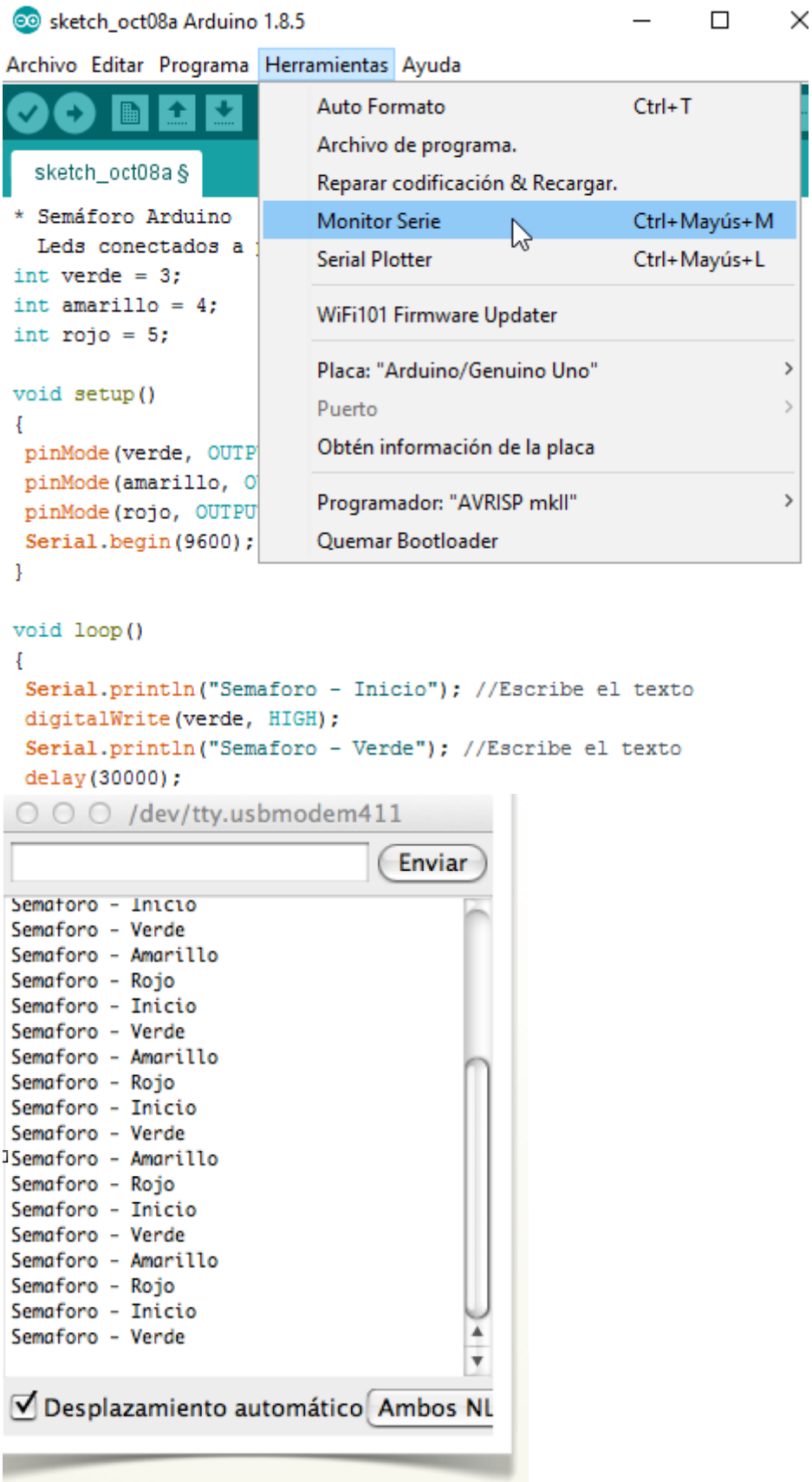
Queremos enseñarte un nuevo comando: **Serial.print**.

Este comando nos manda un texto al puesto serial por el que nos comunicamos con Arduino. De esta manera podemos depurar un programa sabiendo siempre por que línea está.

Para que funcione debemos tener en cuenta que:

- Hay que inicializar Serial. Esto se hace poniendo **Serial.begin(9600)** dentro de la rutina de setup(). 9600 se refiere a la velocidad que se comunicará.
- **Serial.print(“xxx”)** escribe lo que ponemos entre comillas tal cual.
- **Serial.print(x)** escribe el valor que contenga la variable x.
- **Serial.println()** es similar a lo anterior pero después añade un salto de línea.

Para ver lo que nuestro Arduino nos comunica por Serial, abrimos el monitor Serial que tenemos en el programa Arduino:




https://create.arduino.cc/editor/javierquintana/28bf7eaa-441b-46fa-8a1c-7db91075f6ac/preview

https://create.arduino.cc/editor/javierquintana/28bf7eaa-441b-46fa-8a1c-7db91075f6ac/preview?embed

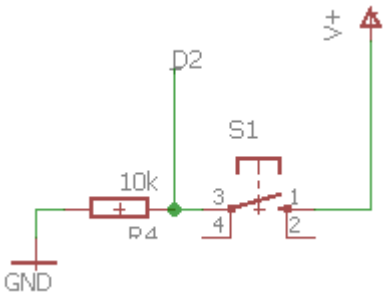
# Montaje 4: Pulsador

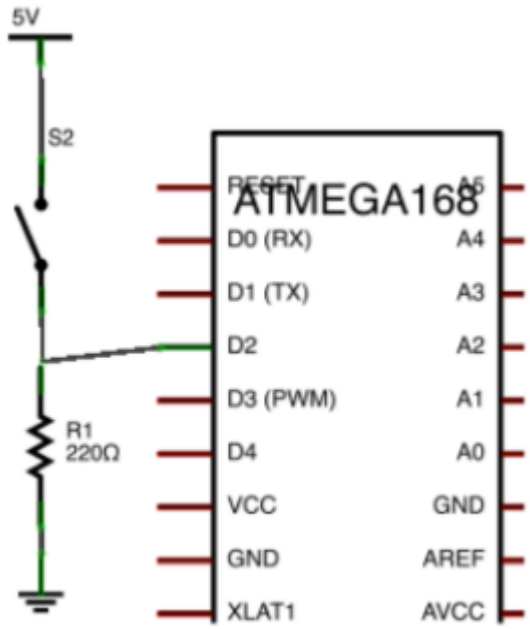
Hasta ahora hemos visto como programar Arduino para que ejecute repetitivamente acciones, pero este actuaba de manera autónoma y nosotros sólo podíamos observar.  Pero podemos interactuar con Arduino, por ejemplo, realizando una acción cuando activamos un pulsador. .

En este ejemplo, vamos a encender un led D3 verde cuando actuamos sobre el pulsador.

## Montaje 4: Pulsador sin EDUBASICA

Utilizamos por ejemplo el pin 2 corresponde al pulsador y el pin 3 al led verde, solo nos queda cargar el programa y probar.




## Montaje 4: Pulsador con EDUBASICA

Con la misma estructura (D2 es donde está el pulsador en EDUBASICA y D3 el LED VERDE) no hace falta realizar ningún cableado.

## Programa 4. Comando digitalRead ...

Aparece un comando nuevo “**digitalRead(buttonPin)**” . Retorna el valor del pin que se ha configurado como entrada y al igual que en el caso de los pines que se configuran como salida, puede tener dos valores HIGH y LOW.

Si es HIGH significa que este pin está unido a la señal de 5v, si es LOW significa que está unido a 0v.

**¿Por qué cuando el pulsador está en OFF D2 está a 0V?** Porque está en PULL-DOWN

https://create.arduino.cc/editor/javierquintana/911ef749-947b-424b-9f88-36e10e88a877/preview

https://create.arduino.cc/editor/javierquintana/911ef749-947b-424b-9f88-36e10e88a877/preview?embed

## Programa 4. con 3 leds

Otra opción es utilizar este programa donde se ve más visual:

https://create.arduino.cc/editor/javierquintana/701c5f35-8022-4b66-bca2-c99ffd7bb906/preview

https://create.arduino.cc/editor/javierquintana/701c5f35-8022-4b66-bca2-c99ffd7bb906/preview?embed

Revision #12
Created 1 February 2022 11:20:11 by Equipo CATEDU
Updated 23 December 2023 19:44:12 by Javier Quintana