1 INTRODUCCIÓN A ARDUINO

Software para Robots

Cristian González García: gonzalezcristian@uniovi.es
Basado en el material original de Jordán Pascual Espada

Introducción

El objetivo de esta práctica es adquirir los conocimientos básicos necesarios para comenzar a manejar la plataforma Arduino. Para realizar esta práctica cada alumno dispondrá de un kit Arduino.

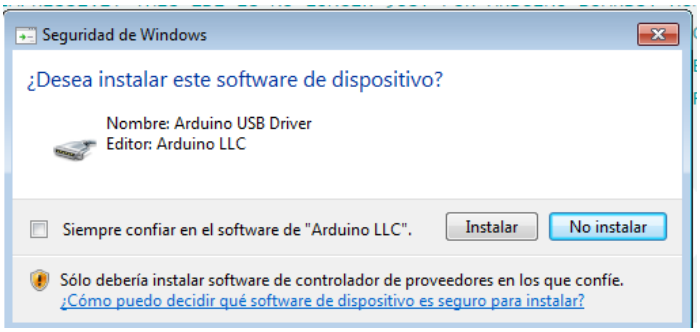
Se recomienda también el uso del entorno de emulación https://www.tinkercad.com para realizar diferentes pruebas. Funciona con cuenta Autodesk. Hay que hacerse una cuenta Autodesk, en caso de que no se tenga. Después, seleccionar «Circuits». En este apartado se tienen diferentes tutoriales y el editor gráfico y simulador de la placa Arduino. También se puede introducir el código en este simulador y así probarlo en este simulador como funciona antes de subirlo al Arduino.

Instalación y configuración

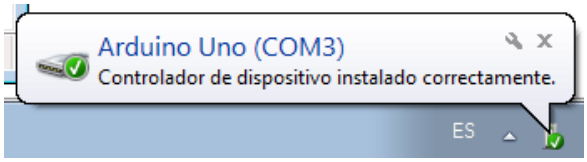
Estos pasos, muy posiblemente, ya estén realizados en los ordenadores del laboratorio. Comprobad primero si está el IDE y si conecta la placa. Si algo falta, entonces seguid estos pasos.

Descargamos y ejecutamos el Software Arduino: https://www.arduino.cc/en/main/software. La última versión es la 1.8.6.

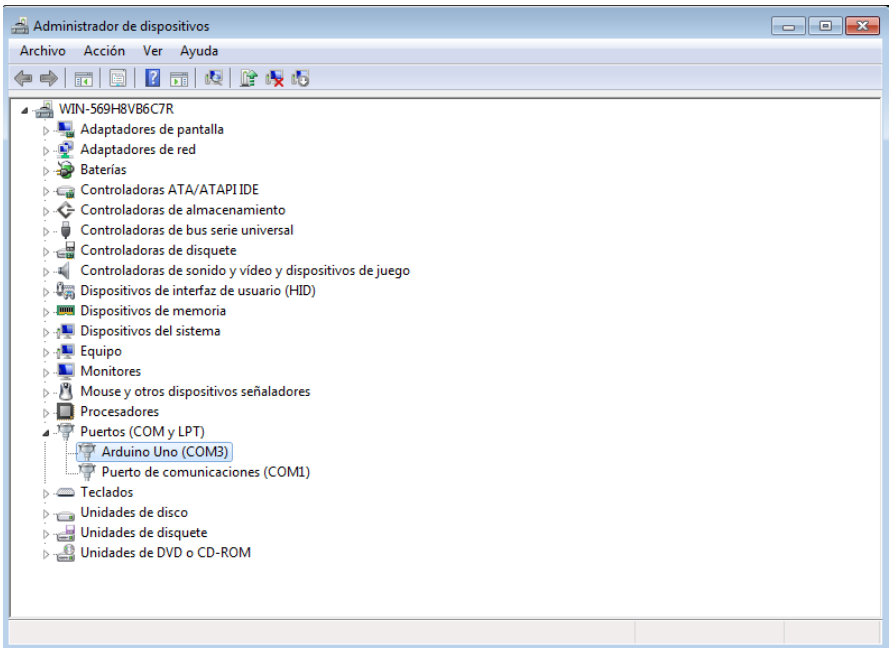
Durante el proceso nos preguntará si queremos instalar varios drivers, los aceptamos todos.

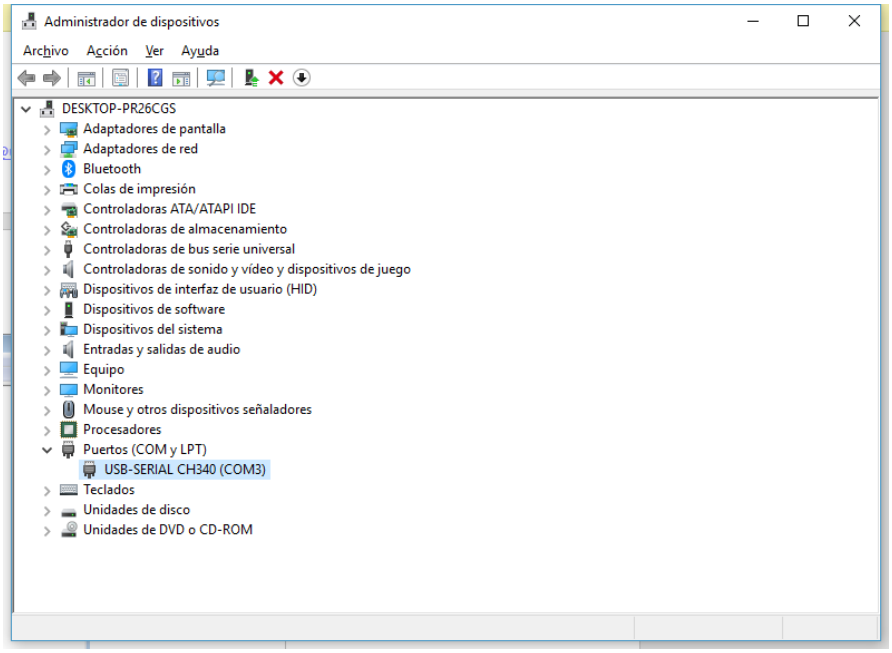


Conectamos la placa al equipo por USB, el cuál debería reconocerlo correctamente.



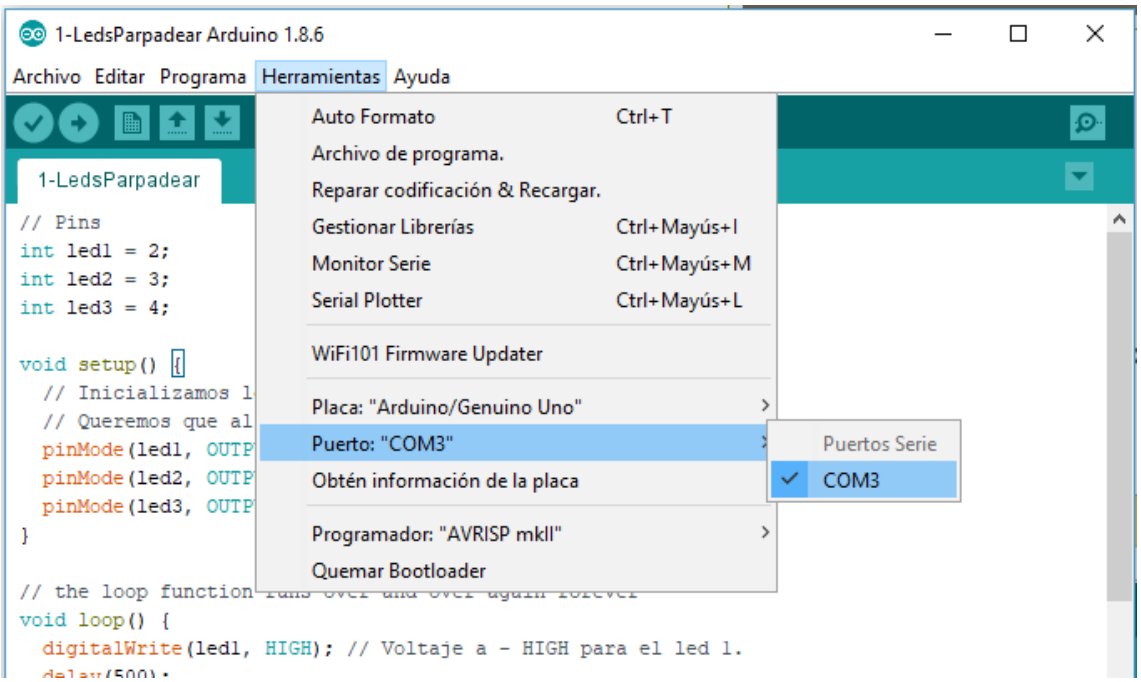
Abrimos el **administrador de dispositivos**, vamos a la categoría **Puertos,** debería aparecernos el dispositivo **Arduino Uno** con uno de los puertos COM asignados, o bien el nombre **CH340** junto a su puerto COM.





Ejecutamos el software de Arduino que instalamos previamente.

Desde la opción «herramientas» **comprobamos que la placa seleccionada es Arduino Uno**, **y** que **el puerto** con el que estamos trabajando es realmente el puerto del Arduino (en este caso, COM3).



Tipo de placa: estamos utilizando los modelos **Arduino Uno** (Ilustración 1), **Arduino BT** (Ilustración 2), o BQ Zum (Ilustración 3). Es importante seleccionar el modelo que se utiliza correctamente.

Ilustración 1 Placa Arduino Uno

* La placa **Arduino BT** tiene un interruptor de encendido en el lateral.

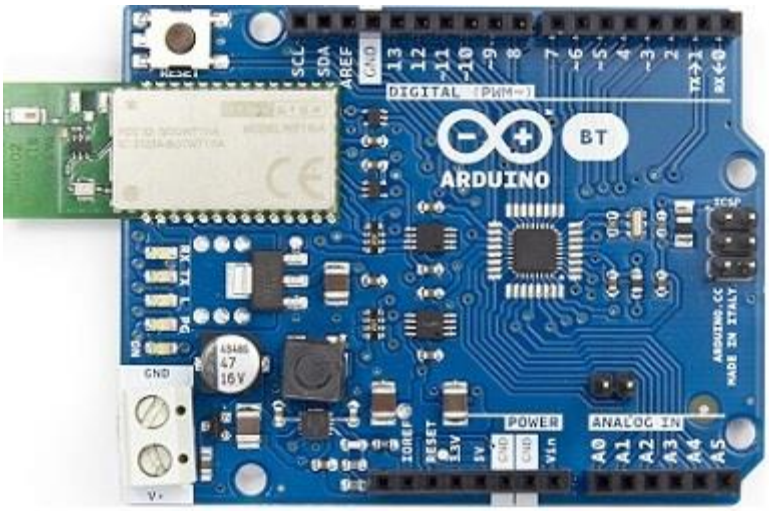


Ilustración 2 Placa Arduino BT

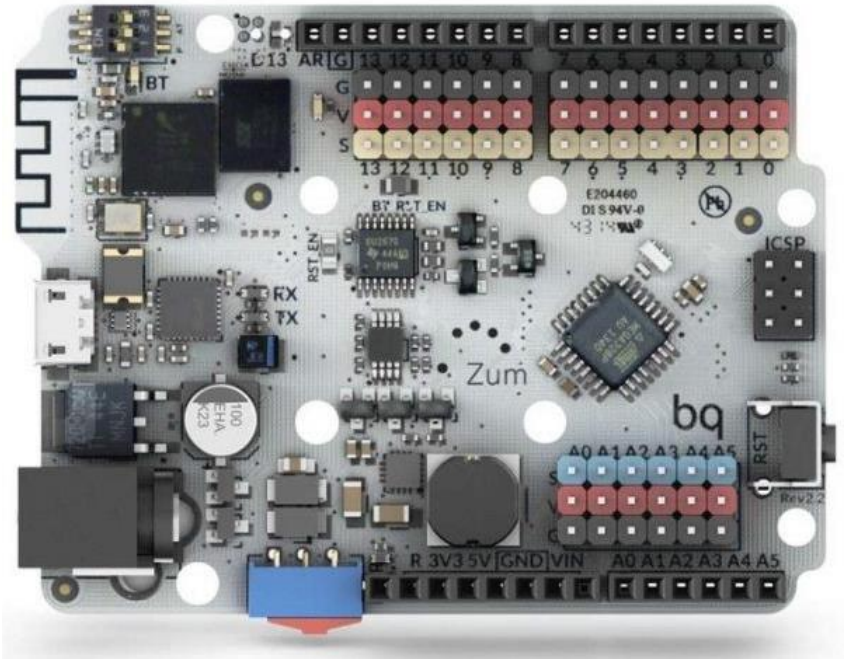
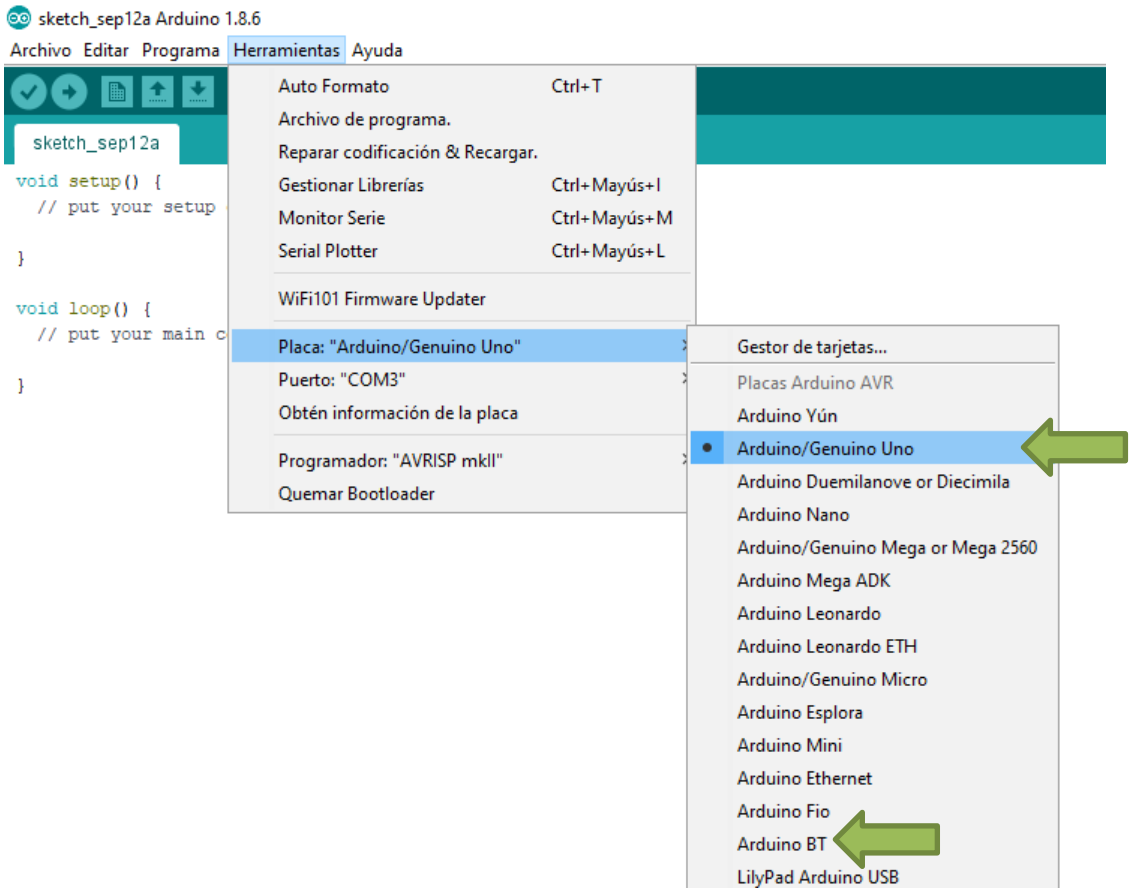
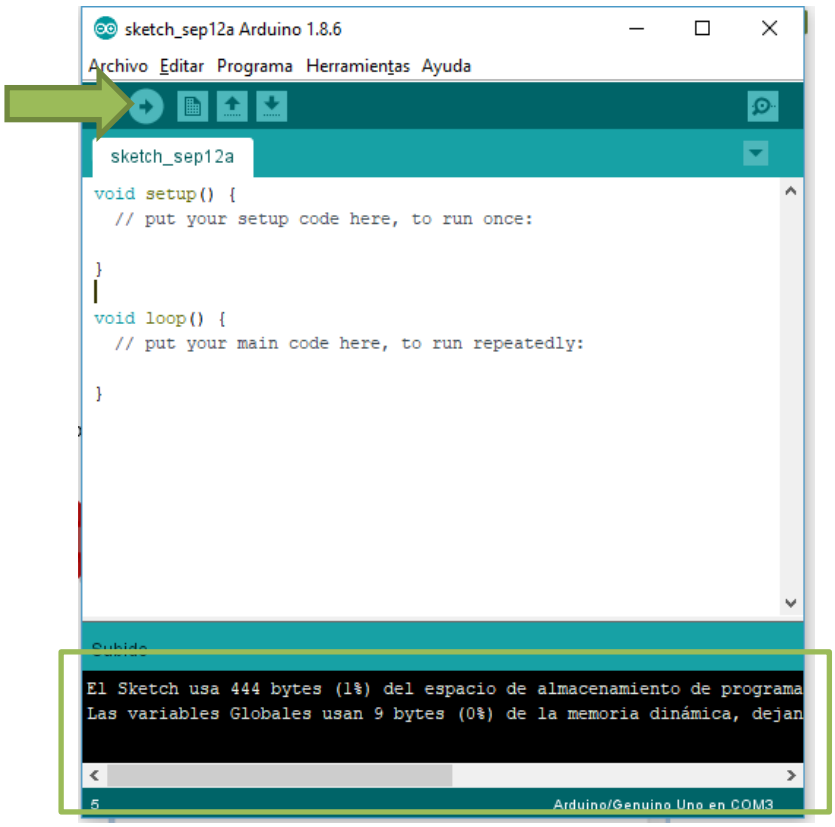


Ilustración 3 Placa BQ Zum

Seleccionamos el tipo de placa en el IDE. Si fuera la BQ Zum seleccionaremos Arduino Uno.



Si todo es correcto, al pulsar en subir el programa la acción debería realizarse con éxito.



NOTA:
En ocasiones estos valores (COM y Placa) se desconfiguran solos provocando errores en la compilación.
En caso de errores de compilación tendréis que revisar que la configuración sea correcta.

Ejemplos

Circuito básico

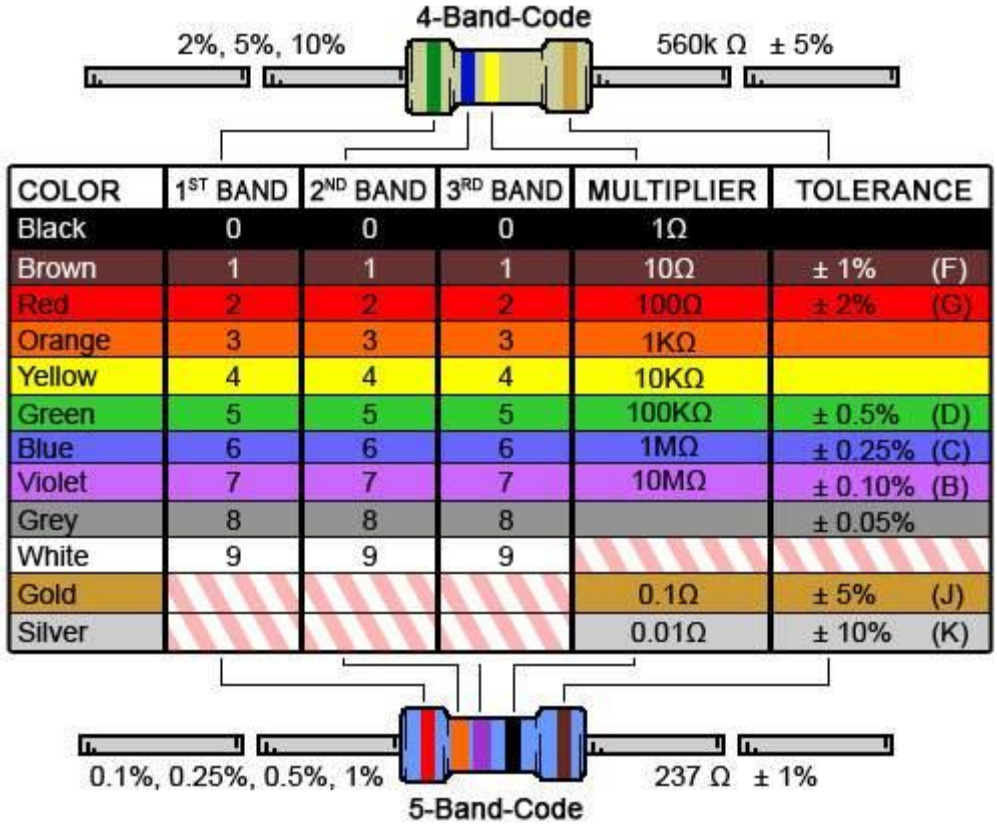
Vamos a crear un circuito muy básico para conectar un Led.

Para ello utilizaremos los siguientes componentes:

Diodo LED: es un componente electrónico que solo permite pasar la corriente en una dirección. En la dirección del positivo al negativo (la parte ancha del triángulo). La pata positiva del LED es la más larga.

Resistencias: agrega una oposición a los electrones, no tienen polaridad. Podemos conocer su valor gracias al código de colores. En este caso para hacer que el Led reciba la potencia adecuada utilizaremos una resistencia de *220 Ω*. Salvo que se utilice el pin 13 del Arduino, pues este pin ya incluye una resistencia.

Calculadora: https://www.digikey.es/es/resources/conversion-calculators/conversion-calculator-resistor-color-code-5-band



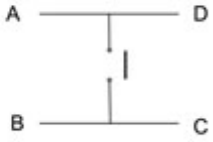
COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)

Pulsador: cuando está abierto rompe el circuito eléctrico, cuando está cerrado activa el circuito. Sirve simplemente para cortar o no la corriente.

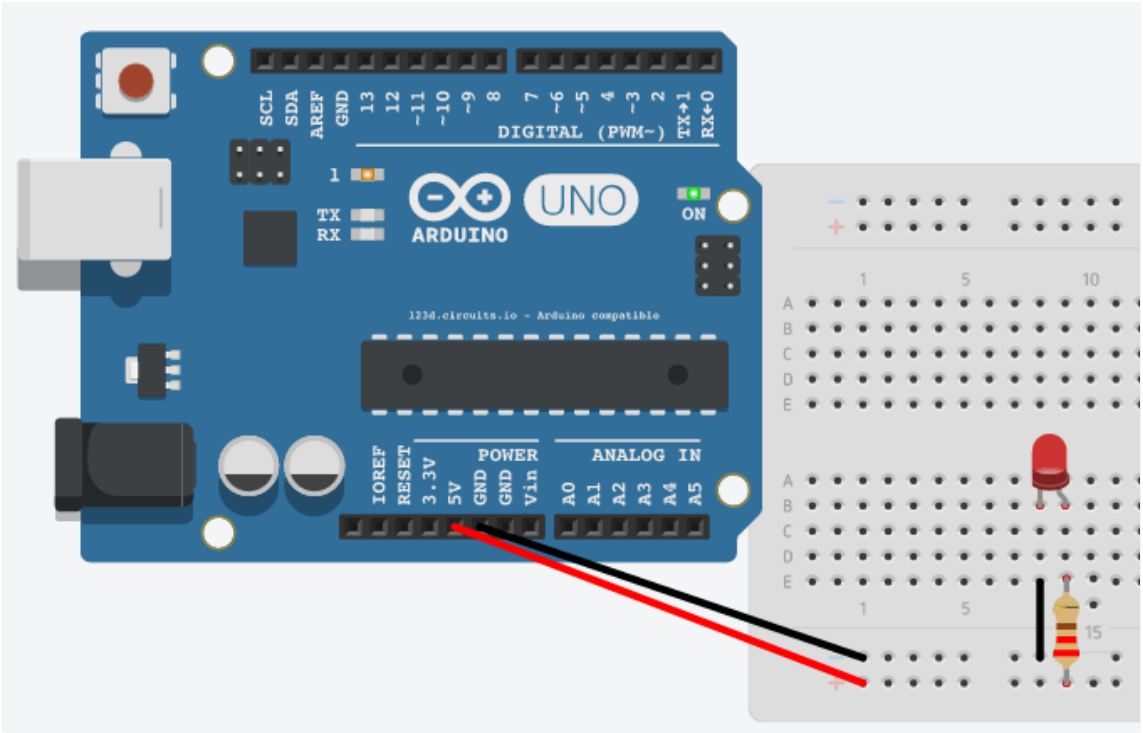
Siempre necesita una resistencia el circuito que lleve un interruptor, salvo que los elementos utilizados cuenten con una o hagan suficiente resistencia al paso de la corriente. Esto es debido a que el pulsador actúa, cuando se pulsa, como si fuera un cable, Luego, conecta positivo y negativo directamente, lo que provoca un cortocircuito salvo que haya una resistencia. El cortocircuito podría estropear el Arduino y otros elementos del circuito. Luego, si se pone un pulsador con un elemento que no tenga resistencia apenas, debería de usarse, para una salida de 5 V (si es de 3,3 V serviría una más pequeña), al menos, una resistencia de 250 Ω, pues el Arduino trabaja en cada pin 0,02 A. Hay gente que usa resistencias de 470 Ω. Ambas son válidas, pero la primera está al límite.

Este modelo de pulsador tiene cuatro patillas, por lo que podría ser utilizado para situaciones «complejas».

En nuestro caso, vamos a utilizarlo como un pulsador simple, haciendo uso únicamente de dos patillas A - B o D - C.



Siempre utilizaremos el cable rojo para el positivo y el negro para el GND (también llamado negativo, -, o tierra). No obstante, se llama así por el tema positivo-negativo, pero no tiene polaridad negativa.



Construcción del circuito:

1. Conectamos el pin **5V** al carril positivo de la ProtoBoard (Rojo).
2. Conectamos el pin **GND** al carril negativo del ProtoBoard (Azul).
3. Colocamos el Led rojo en la ProtoBoard. Cuidad con la polaridad del Led.
4. Utilizamos la resistencia para abrir un carril positivo que se conecte con la parte positiva del Led (Extremo largo). Mirar que resistencia es la adecuada.

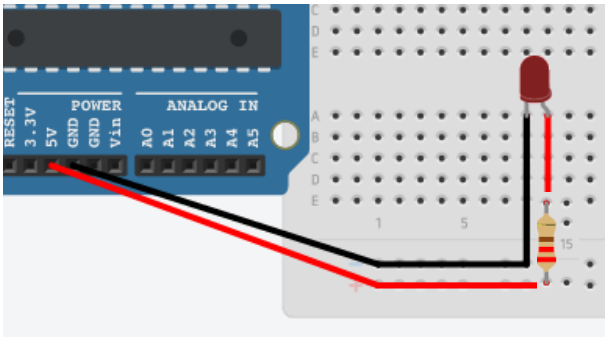
> Los Leds apenas presentan resistencia, luego, si toda la corriente pasa por el led este se quemará. Para evitar esto

debemos colocar primero una resistencia de *220-330 Ω*, depende de la luminosidad que queramos.

5. Cerramos el circuito uniendo el carril de la pata negativa del led con el carril negativo del circuito.

¿Cómo sería este circuito si no estuviésemos utilizando la ProtoBoard?

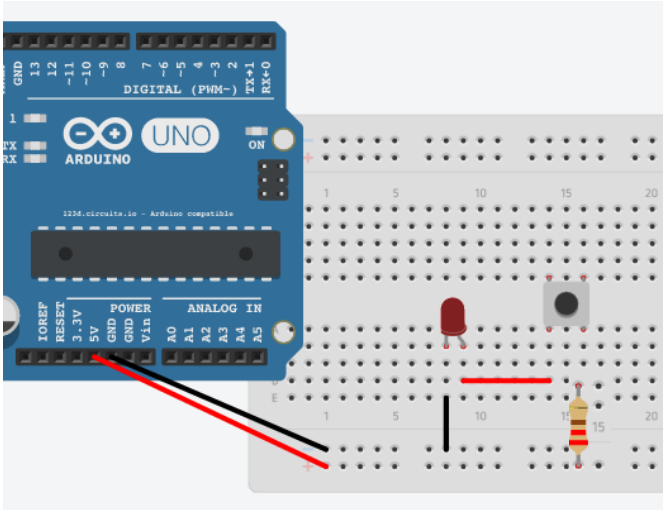
Sería algo similar a lo siguiente, con un cable en la zona de polaridad y un cable en cada carril.



Una vez comprobamos que el circuito funciona correctamente vamos a extenderlo, agregando un pulsador para cortar / abrir la corriente.

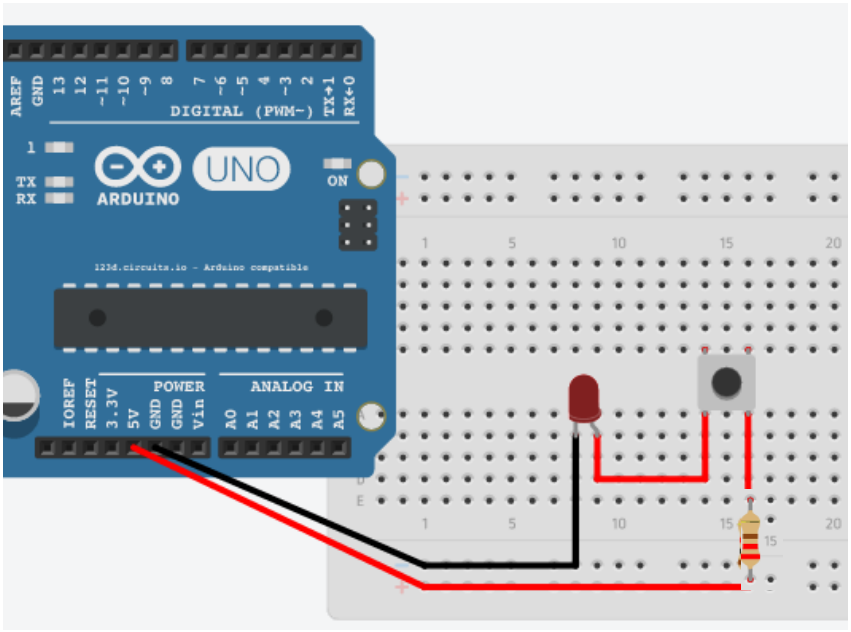
Incluir un pulsador para cortar / abrir el circuito

Si conectamos una patilla del pulsador a cada carril este hará pasar la electricidad de uno a otro cuando esta pulsado, ya que, al pulsarlo, se cierra el circuito. Sino, solo pasaría la electricidad a la patilla del mismo lado.



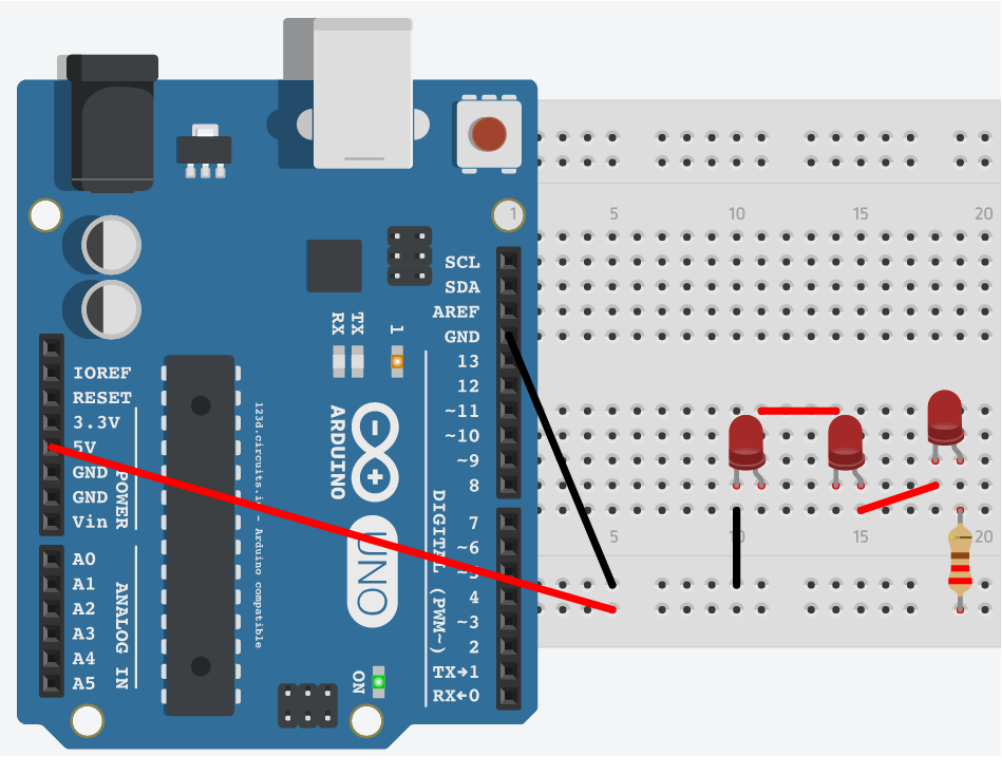
¿Cómo sería este circuito si no estuviésemos utilizando la ProtoBoard?

Sería algo similar a lo siguiente, con un cable en la zona de polaridad y un cable en cada carril.



Circuito en serie

Vamos a probar a conectar varios leds en serie para ver el efecto de la corriente eléctrica.

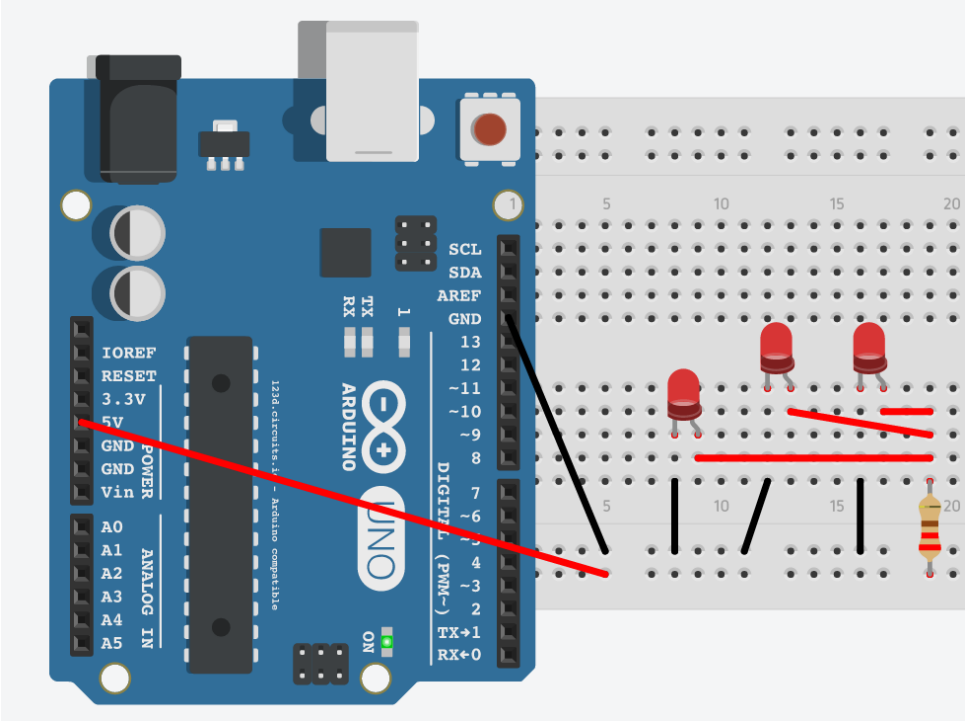


Como comentamos anteriormente, un Led hace caer el voltaje, luego, si colocamos varios leds en serie las caídas de voltaje se suman**. Esta mayor caída de voltaje unida a la resistencia hace que los leds apenas se iluminen o ni se iluminen**.

- Este circuito requeriría un mayor voltaje para encender todos los Leds, o una resistencia más baja.

- Si un Led se rompe, el circuito dejará de funcionar y todos los Leds se apagarán.
- **La intensidad del circuito se mantiene respecto al primer ejemplo, por lo que consume la misma electricidad**.

Circuito en paralelo



Cuando conectamos los Leds en paralelo, todas las ramas reciben el mismo voltaje: el voltaje original menos la caída de un único led.

Los caminos con menor resistencia son los que mayor intensidad reciben (en este caso todos tienen la misma). En este caso, **todos los caminos reciben la misma intensidad provocando que los Leds se iluminen igual que antes, lo que aumenta aquí es el gasto de electricidad**.

- La intensidad que necesita este circuito en paralelo es mayor que la que vimos anteriormente en el circuito en serie.
- Incluso con un voltaje pequeño podríamos hacer funcionar muchos leds.
- Si un led se funde no le pasará nada al resto.

Circuitos electrónicos y programación

En nuestros proyectos futuros primará la parte de la programación del microcontrolador frente a la construcción de circuitos electrónicos.

La placa Arduino se utilizará para conectar varios elementos (sensores y actuadores), formando cada uno de ellos un pequeño circuito muy simple. La programación de las entradas y las salidas será la parte encargada de coordinar los elementos electrónicos.

Salidas digitales - Leds que parpadean

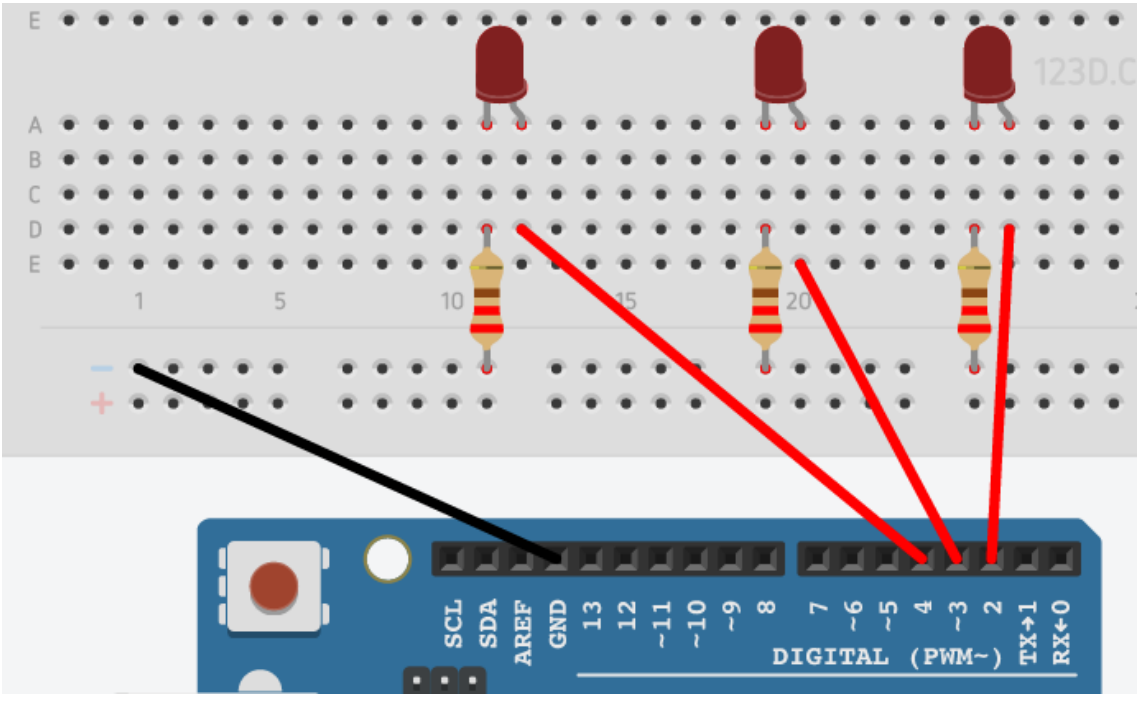
Vamos a crear un sistema con 3 leds que se enciendan y se apaguen de forma ordenada, primero el A, luego el B y finalmente el C, una vez terminada la secuencia está se volverá a repetir de forma indefinida.

Para ello utilizaremos los siguientes componentes:

Diodo LED: es un componente electrónico que solo permite pasar la corriente en una dirección. En la dirección del positivo al negativo (la parte ancha del triángulo). La pata positiva del LED es la larga.

Resistencias: agrega una oposición a los electrones, no tienen polaridad. Podemos conocer su valor gracias al código de colores.

En este caso, para hacer que el Led reciba la potencia adecuada utilizaremos una resistencia de *220 Ω*.



En este sistema tenemos tres pequeños circuitos:

- Led 1 (Pin 2 - [salida 0V/5V] - Led 1 - Resistencia - GND).
- Led 2 (Pin 3 - [salida 0V/5V] - Led 2 - Resistencia - GND).
- Led 3 (Pin 4 - [salida 0V/5V] - Led 3 - Resistencia - GND).

Los tres circuitos tienen conectado un pin de Arduino, que en este caso servirá para abrir/cerrar la alimentación eléctrica de los leds. Desde un programa ejecutado en Arduino podremos controlar cuando abrir o cerrar la alimentación de cada pin.

Construcción del circuito:

1. Colocamos 3 leds en la ProtoBoard.
2. Conectamos el pin digital 2 a la parte positiva del primer led, el pin digital 3 a la parte positiva del segundo led y el pin digital 4 a la parte positiva del tercer led.
3. Conectamos el pin GND al carril negativo de la ProtoBoard.
4. Ahora debemos cerrar el circuito de cada uno de los tres leds. No podemos hacerlo directamente ya debemos incluir una resistencia para evitar que los leds se quemen. Añadimos la resistencia, en este caso en la parte negativa del led y aprovechamos la propia resistencia para conectar el led al carril negativo y así cerrar el circuito.

Programación de Arduino:

1. Inicializamos tres variables con los pins digitales que vamos a utilizar.

```
// Pins
int led1 = 2;
int led2 = 3;
int led3 = 4;
```

2. Dentro del método `setup()` declaramos los pins digitales que vamos a utilizar para darle electricidad a los leds.

```
void setup() {
  // Inicializamos los pins digitales como salida
  // Queremos que alimenten electricamente los leds
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
}
```

3. Dentro del `loop()` comenzamos a enviar las salidas digitales con el método `digitalWrite`.
   a. Esta función recibe el número del pin y el voltaje que se enviara (HIGH o LOW). En caso de HIGH se envían 5 voltios y de LOW 0 voltios, es decir, actúa como GND.
   b. https://www.arduino.cc/en/pmwiki.php?n=Reference/DigitalWrite

4. Encendemos el led 1 dentro de la función loop

```
digitalWrite(led1, HIGH); // Voltaje a - HIGH para el led 1.
delay(500);
```

5. Apagamos el led 1 y encendemos el led2

```
digitalWrite(led1, LOW); // Voltaje a - LOW para el led 1.
digitalWrite(led2, HIGH); // Voltaje a - HIGH para el led 2.
delay(500);
```

6. Apagamos el led2 y encendemos el led3

```
digitalWrite(led2, LOW); // Voltaje a - LOW para el led 1.
digitalWrite(led3, HIGH); // Voltaje a - HIGH para el led 2.
delay(500);
```

7. Apagamos el led3

```
digitalWrite(led3, LOW); // Voltaje a - LOW para el led 3.
```

```
// Pins
int led1 = 2;
int led2 = 3;
int led3 = 4;

void setup() {
  // Inicializamos los pins digitales como salida
  // Queremos que alimenten electricamente los leds
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
}

// El loop es llamado una y otra vez sin parar
void loop() {
  digitalWrite(led1, HIGH); // Voltaje a - HIGH para el led 1.
  delay(500);

  digitalWrite(led1, LOW); // Voltaje a - LOW para el led 1.
  digitalWrite(led2, HIGH); // Voltaje a - HIGH para el led 2.
  delay(500);

  digitalWrite(led2, LOW); // Voltaje a - LOW para el led 1.
  digitalWrite(led3, HIGH); // Voltaje a - HIGH para el led 2.
  delay(500);

  digitalWrite(led3, LOW); // Voltaje a - LOW para el led 3.
}
```

Entradas digitales - Controlando los leds con un botón

Vamos a crear un sistema con 2 LEDs y un botón. Cuando se pulse el botón se encenderá el LED 1 y cuando el botón se vuelva a pulsar se encenderá el led 2. Si se vuelve a pulsar el botón, se apagarán los dos leds y se volverá al estado inicial.

Para ello utilizaremos los siguientes componentes:

Diodo LED: es un componente electrónico que solo permite pasar la corriente en una dirección. En la dirección del positivo al negativo (la parte ancha del triángulo). La pata positiva del LED es la larga.

Resistencias agrega una oposición a los electrones, no tienen polaridad. Podemos conocer su valor gracias al código de colores.

En este caso para hacer que el Led reciba la potencia adecuada utilizaremos una resistencia de *220 Ω*.

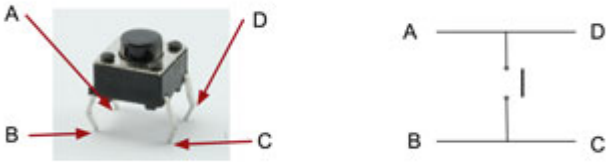
Resistencia de 10k*Ω*.

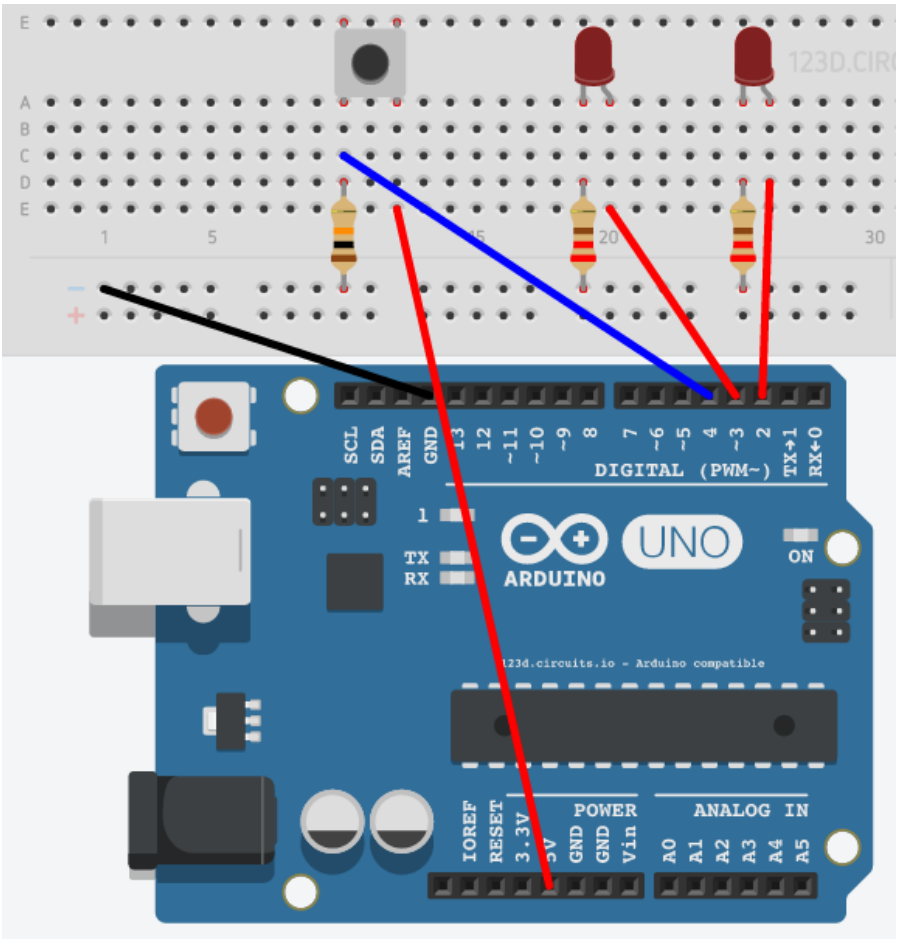
Pulsador: cuando está abierto rompe el circuito eléctrico, cuando está cerrado activa el circuito. Sirve simplemente para cortar o no la corriente.

Siempre necesita una resistencia el circuito que lleve un interruptor, salvo que los elementos utilizados cuenten con una o hagan suficiente resistencia al paso de la corriente. Esto es debido a que el pulsador actúa, cuando se pulsa, como si fuera un cable, Luego, conecta positivo y negativo directamente, lo que provoca un cortocircuito salvo que haya una resistencia. El cortocircuito podría estropear el Arduino y otros elementos del circuito. Luego, si se pone un pulsador con un elemento que no tenga resistencia apenas, debería de usarse, para una salida de 5 V (si es de 3,3 V serviría una más pequeña), al menos, una resistencia de 250 Ω, pues el Arduino trabaja en cada pin 0,02 A. Hay gente que usa resistencias de 470 Ω. Ambas son válidas, pero la primera está al límite.

Este modelo de pulsador tiene cuatro patillas, por lo que podría ser utilizado para situaciones «complejas».

En nuestro caso, vamos a utilizarlo como un pulsador simple, haciendo uso únicamente de dos patillas A - B o D - C.





En este sistema tenemos varios pequeños circuitos:

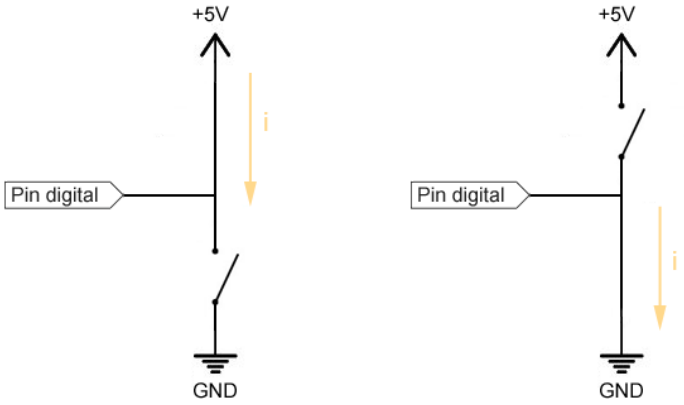
- Botón (5V - Botón - Pin 4 [entrada 0V/5V]- Resistencia - GND).
- Led 1 (Pin 3 - [salida 0V/5V]- Led 1 - Resistencia - GND).
- Led 2 (Pin 2 - [salida 0V/5V]- Led 2 - Resistencia - GND).

Los tres circuitos tienen conectado un pin de Arduino, que sirven, en el primer caso, para saber si el circuito del botón está abierto/cerrado, y en los otros dos casos para abrir/cerrar la alimentación eléctrica de los leds.

El Arduino es el elemento central que está sirviendo para manejar el funcionamiento de esos tres circuitos.

Construcción del circuito:

1. Colocamos 2 leds en la ProtoBoard.
2. Conectamos el pin digital 2 a la parte positiva del primer led, el pin digital 3 a la parte positiva del segundo LED.
3. Conectamos el pin GND al carril negativo de la ProtoBoard.
4. Ahora debemos cerrar el circuito de cada uno de los dos leds. No podemos hacerlo directamente **debemos incluir una resistencia para evitar que los leds se quemen**. Añadimos la resistencia, en este caso, en la parte negativa del led y aprovechamos la propia resistencia para conectar el led al carril negativo y así cerrar el circuito.
5. Colocamos el pulsador en la ProtoBoard y conectamos a una patilla del pulsador el cable de alimentación de 5V.
6. Conectamos a la otra patilla del pulsador el pin digital 4. Desde este pin podremos leer el voltaje de este pequeño circuito.
7. Ahora deberíamos cerrar el circuito, pero **no podemos conectar el interruptor directamente al carril negativo ya que se produciría un cortocircuito**.

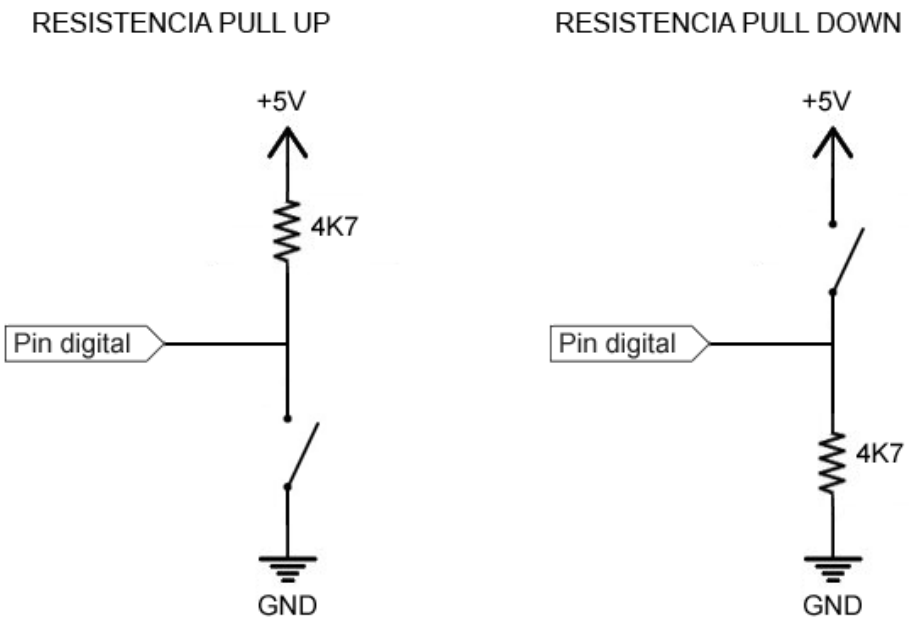


Nunca debemos conectar directamente 5V a GND o provocaremos un cortocircuito, hay que incluir una resistencia de por medio siempre y cuando los elementos que utilicemos no tengan resistencias incorporadas o no hagan apenas resistencia a la electricidad, como ocurre con el pulsador y el LED.

Si conectamos directamente 5V y GND (0V) causaremos un cortocircuito por el elevado paso de corriente, por lo que produciría un calentamiento de los elementos, pudiendo llegar incluso al incendio.

El pin digital que tiene como misión leer el voltaje de esa conexión tampoco funcionaria correctamente, ya que estaría conectado simultáneamente a: 5V y GND (0V).

Para solucionar esta situación debemos colocar una resistencia **Pull-Down o Pull-Up**. Este tipo de resistencias se colocan entre los extremos **5V y GND** para forzar el valor de la tensión.



El esquema **Pull-Up** fuerza a HIGH el valor cuando el pulsador está abierto (Si está cerrado el valor será LOW).

El esquema **Pul-Down** fuerza a LOW el valor cuando el pulsador está abierto (Si está cerrado el valor será HIGH).

¿Qué resistencia hay que colocar?

- Una resistencia muy pequeña deja pasar mucha corriente (0 o cerca de 5), por lo que las mediciones del voltaje del circuito resultan bastante precisas, pero supone un mayor consumo y calentamiento.
- Una resistencia muy grade, deja pasar muy poca corriente (0 o lejos de 5), por lo que las mediciones de voltaje del circuito son más propensas a mediciones incorrectas (ruido).

Para la tensión de Arduino 0-5V se suelen colocar resistencias de 4k-10k, pues suponen un consumo de electricidad de 1mA y 0,5mA cuando el circuito está activo.

Así que utilizamos la resistencia de 10k para conectar el pulsador a GND siguiendo el esquema Pull-Down.

Programación de Arduino:

```
// Pins
int led1 = 2;
int led2 = 3;
int button = 4;
int buttonState= 0;
// 0: todo apagado
// 1: led1 encendido
// 2: led2 encendido

void setup() {
```

```
  Serial.begin(9600);  // Iniciar el Serial
  Serial.println("Setup");

  // Inicializamos los pines digitales como salida
  // Queremos que alimenten electricamente los leds
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  // Queremos leer la entrada
  pinMode(button, INPUT);
}

// La funcion loop se llama una y otra vez hasta el infinito
void loop() {
  int read = digitalRead(button);

  if(read == HIGH){
    Serial.println("Valor HIGH");
    if(buttonState == 2){
      buttonState = 0;
    } else {
      buttonState++;
    }

    switch(buttonState){
      case 0:
        digitalWrite(led1, LOW);
        digitalWrite(led2, LOW);
        break;
      case 1:
        digitalWrite(led1, HIGH);
        digitalWrite(led2, LOW);
        break;
      case 2:
        digitalWrite(led1, LOW);
        digitalWrite(led2, HIGH);
        break;
    }

  }
}
```

Probamos el programa y detectamos un problema

Cuando pulsamos el botón, los LEDs comienzan a parpadear, y al soltarlo, los LEDs se quedan en un estado «aleatorio».

¿Porque está funcionando de esta forma? Mientras mantenemos el botón pulsado el método `loop()` se sigue ejecutando, por lo que probablemente se haya ejecutado cientos de veces y el estado de los leds habrá cambiado en cada una de esas ejecuciones.

Tenemos dos estrategias para mejorar el sistema:

Estrategia 1: la más simple. Cada vez que se produce un cambio de estado en los LEDs colocar un «delay(1000);». De esta forma nos aseguramos que el botón tenga que estar pulsado al menos 1 segundo para realizar un nuevo cambio en el estado de las luces.

```
      case 2:
        digitalWrite(led1, LOW);
```

```
        digitalWrite(led2, HIGH);
        break;
    }
    delay(1000);
  }
}
```

Estrategia 2: creamos una variable que controle que la lógica del botón no se ejecute más de una vez, a no ser que el usuario levante el dedo del botón, en cuyo caso si se podrá volver a ejecutar.

```
...
int pushed = 0;
// 0: botón sin pulsar
// 1: botón pulsado

void loop() {
  ...

  // Si han pulsado el botón y estaba sin pulsar ->
  if(read == HIGH && pushed == 0){
    // botón pulsado!
    pushed = 1;

    ...
    ...

  // Si han quitado el dedo del botón cambio su estado
  } else if (read == LOW && pushed == 1){
     Serial.println("Valor LOW");
     pushed = 0;
   }
}
```

Monitor de serie:

El monitor de serie nos permite enviar y recibir datos desde Arduino. Para comenzar a utilizarlo lo inicializamos en el método **setup().**

```
Serial.begin(9600);  // Iniciar el serial
```

A partir del momento de su inicialización podemos escribir mensajes que se mostrarán por el monitor de serie.

```
Serial.println("Mensaje");
```

Para ver los mensajes debemos abrir el monitor de serie. **Solo se puede abrir si la placa está conectada.**

El método println() no realiza **conversiones automáticas de tipos de datos**. Si queremos incluir un valor no String en la cadena debemos convertirlo previamente. Ejemplo:

```
Serial.println("Valor de la entrada: "+ String(value));
```

Lectura de entrada analógica

Vamos a crear un sistema que regule la intensidad de un diodo LED utilizando un potenciómetro. Al mover la rueda del potenciómetro cambiaremos la intensidad de la luz del LED.

Para ello utilizaremos los siguientes componentes:

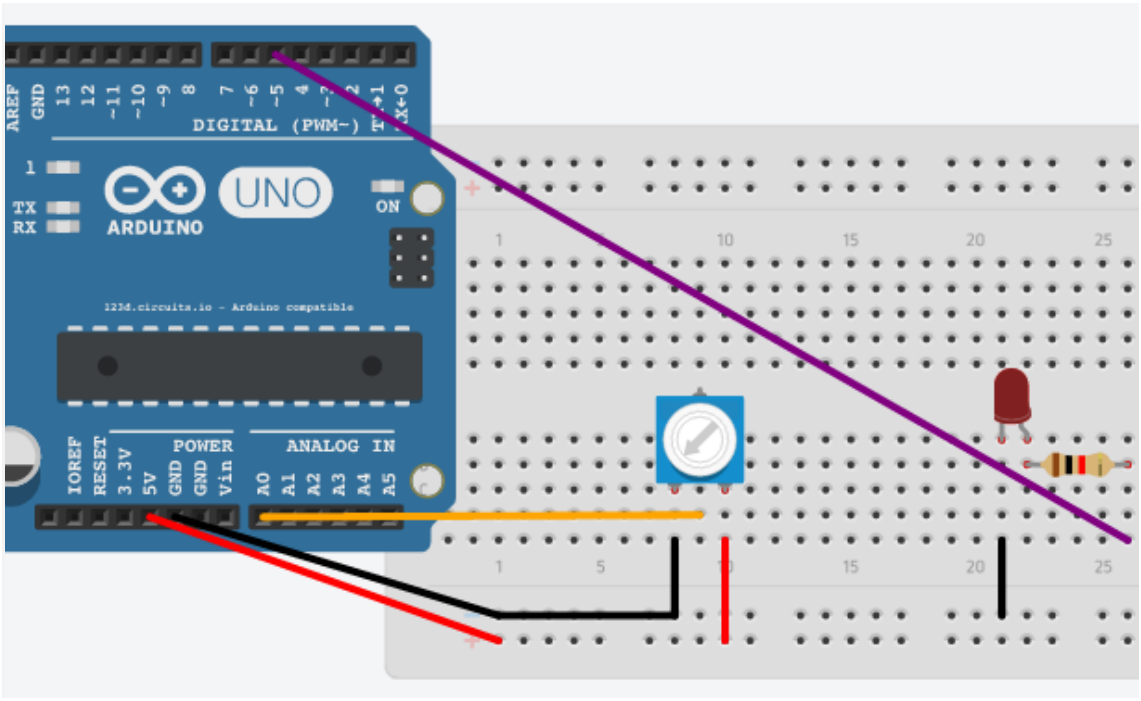
Diodo LED: es un componente electrónico que solo permite pasar la corriente en una dirección. En la dirección del positivo al negativo (la parte ancha del triángulo). La pata positiva del LED es la larga.

Resistencias: agrega una oposición a los electrones, no tienen polaridad. Podemos conocer su valor gracias al código de colores.

En este caso para hacer que el Led reciba la potencia adecuada utilizaremos una resistencia de *220 Ω*.

Potenciómetro de 10K *Ω*: es una resistencia variable que sirve para variar la intensidad de la resistencia girando la rueda.

Nota: **si se conectan del revés el positivo y el negativo al potenciómetro, entonces, el potenciómetro, funcionaría del revés**.



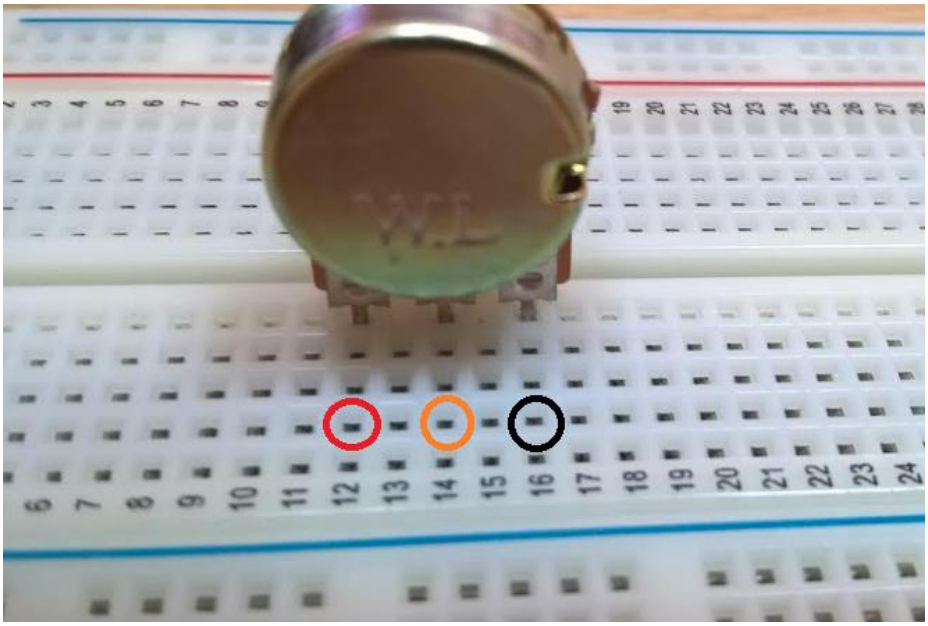


Ilustración 4 Conexión de potenciómetro en ProtoBoard. Rojo a 5V, Naranja a pin A0 y Negro a GND. Si + y -se conectan del revés, funcionaría el giro del revés.

En este sistema tenemos dos pequeños circuitos

- Led (Pin 5 - [salida analógica entre: 0V - 5V]- Resistencia - Led - GND).
- Potenciómetro 3 (5V - Potenciómetro – GND + Pin A0 [entrada analógica]).

Construcción del circuito:

1. Colocamos el potenciómetro en la ProtoBoard, conectamos su pin derecho e izquierdo a los carriles positivo y negativo.

2. Conectamos el pin central del potenciómetro a la entrada analógica A0.
3. Colocamos el Led en la ProtoBoard, en su polo positivo conectamos una resistencia de 220 Ω.
4. Conectamos el inicio de la resistencia al pin analógico ~5, que además de ser una fuente de voltaje, este pin es capaz de «regularlo», pues permite una escritura analógica.
    a. https://playground.arduino.cc/Learning/Pins
    b. https://www.arduino.cc/en/Tutorial/DigitalPins
5. Para cerrar el circuito conectamos el polo negativo del LED al carril negativo.

Programación de Arduino:

1. Inicializamos dentro del método `setup()` el puerto serial para poder escribir mensajes.

```
Serial.begin(9600);  // Iniciar el Serial
```

2. También declaramos los pines digitales que vamos a utilizar. En este caso el pin 5 como salida para darle electricidad al Led.

```
// Los pins digitales se declaran - LED
pinMode(5, OUTPUT); // Salida
```

3. Creamos dos variables globales para almacenar el valor del potenciómetro y el valor que le daremos a la salida analógica.

4. Dentro del `loop()` leemos la entrada analógica A0 donde está conectado el potenciómetro.

    a. https://www.arduino.cc/en/Reference/analogRead

```
valuePotenciometer = analogRead(A0); // 0 - 1024
```

5. La función analogRead() devuelve valores entre 0 - 1024, pero la salida analógica admite valores de 0 – 255. Por ello, realizamos una conversión antes de escribir la salida.
    a. https://www.arduino.cc/en/Reference/Map

```
// Transformar de 0 a 1023 -> 0 a 255
valueAnalogPin = map(valuePotenciometer, 0, 1023, 0, 255);
```

6. Escribimos el valor convertido en la salida analógica 5.

```
// Escribimos el valor en el pin 5 - LED
analogWrite(5, valueAnalogPin);
```

7. Código de ejemplo:

```
int valuePotenciometer;
int valueAnalogPin;

void setup() {
  Serial.begin(9600);  // Iniciar el Serial
  Serial.println("Setup");

  // Los pins digitales se declaran - LED
  pinMode(5, OUTPUT); // Salida
}
```

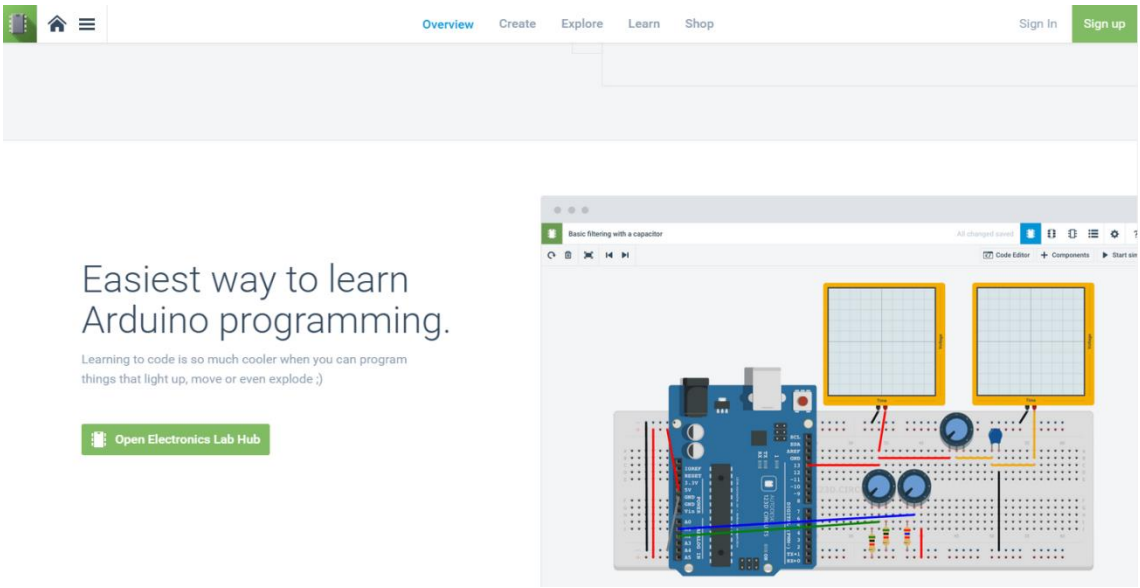
```
void loop() {
  // Leemos el valor del pin A0 - Potenciometro
  valuePotenciometer = analogRead(A0); // 0 - 1024
  Serial.println("valor: "+String(valuePotenciometer));

  // Transformar de 0 a 1023 -> 0 a 255
  valueAnalogPin = map(valuePotenciometer, 0, 1023, 0, 255);
  // Escribimos el valor en el pin 5 - LED
  analogWrite(5, valueAnalogPin);
}
```

Ampliación:

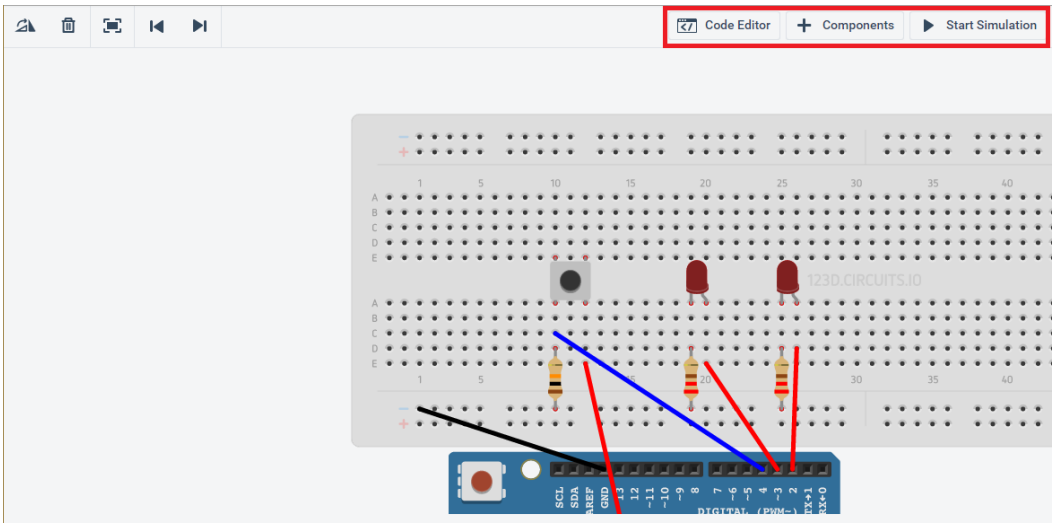
Conectar 3 LEDs al circuito y hacer que en función del valor del potenciómetro se enciendan 0, 1, 2 o 3 LEDs. En este caso, todos los leds se deben encenderse siempre con la misma intensidad. Utilizar las salidas digitales para encenderlos.

Tinkercad

Es una plataforma web que permite crear circuitos y compartirlos. El emulador de circuitos soporta gran parte de los elementos básicos de Arduino y la programación de la placa.



Resulta muy útil para hacer pruebas, esquemas eléctricos y ver sistemas diseñados por otras personas.


Code Editor
Components
Start Simulation
123D.CIRCUITS.IO