

Estación 2:

Tema 2: Introducción a la electrónica

En esta sección aprenderemos la electrónica básica para aprender a realizar circuitos y resolver problemas a través de la electricidad. Existen diversos sensores que se utilizan para conocer parámetros o valores de variables que influyen directamente en un proceso de producción.

Por ejemplo: en una granja quieren aumentar la producción de gallinas durante el invierno, el problema es que con las temperaturas bajas de Mendoza las incubadoras pierden mucho calor y muchos de los huevos no se desarrollan a pollitos. Se propone como solución agregar calefacción y mantener la temperatura cálida en todo momento. Vamos con algunas preguntas:

¿Cómo desarrollarían este sistema? ¿Debemos saber de electricidad? ¿Podemos solucionarlo con un programa de Python? ¿Qué materiales necesitamos? ¿Cuánto cuesta? ¿Qué cosas debemos cobrar?

¿Necesitamos sensores? ¿Necesitamos actuadores?

Las respuestas a todas estas preguntas serán respondidas durante el desarrollo de esta sección.

Tema 2.1- ¿Qué es la energía eléctrica?

La energía eléctrica es la que proviene del movimiento de electrones de un lugar a otro, existen muchos tipos de energías como la térmica, lumínica, eólica, de movimiento (cinética), potencial, etc.

La energía eléctrica fluye a través de los conductores, como los cables. Se puede transformar en otras formas de energía para hacer algo interesante, como encender una luz (energía lumínica) o reproducir sonidos a través de un parlante.

Tema 2.2-Transductores, sensores y actuadores

Los componentes encargados de transformar la energía, como los altoparlantes o leds, son los transductores eléctricos. Los transductores transforman otros tipos de energía en energía eléctrica y viceversa.

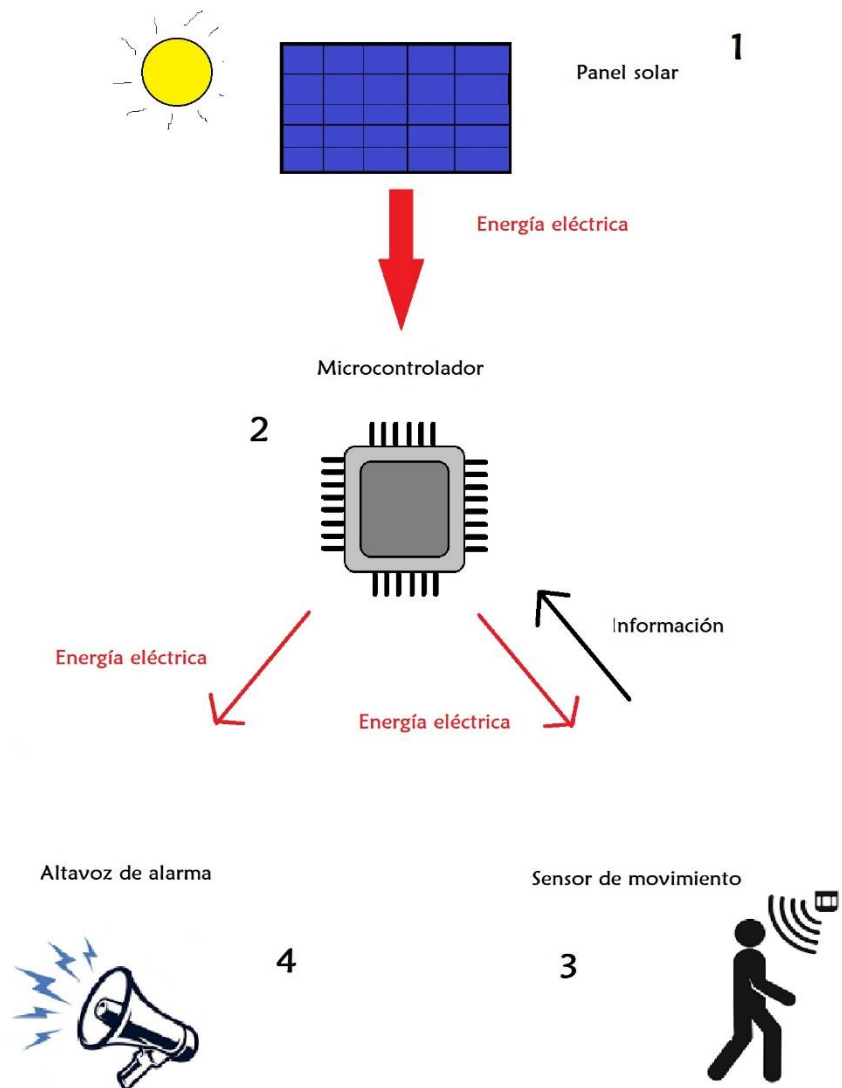
Hay dos tipos de transductores. Los componentes que transforman otras formas de energía en energía eléctrica son llamados sensores, y los componentes que convierten la energía eléctrica en otras formas de energía se conocen con el nombre de actuadores.

Por ejemplo, un micrófono y un parlante: si los observamos son muy similares, ambos tienen un imán y un bobinado. Al suministrar electricidad al parlante, vibra y genera un sonido. El micrófono vibra por el sonido y lo transforma en electricidad.

[Cómo Funciona un Micrófono y un Altavoz !\[\]\(b4eeff342f60cc7bcd67d869b4fedca2_img.jpg\) !\[\]\(7cbfaf281ed50ce10ba1259f16ecca5e_img.jpg\) Cómo se Propaga el Sonido \(ft. @Wikiseba \)](#)

Analizaremos el siguiente sistema de alarma para hogares para ver qué función cumple cada elemento y cómo podemos categorizarlas según lo planteado anteriormente.

El primer elemento señalado (1) es la mayor fuente de energía del sistema; un panel solar. Este es un transductor que convierte energía solar en energía eléctrica. Está conectado al microcontrolador (2), que es el “cerebro” del sistema de alarma. Se encarga de recibir información del exterior y tomar las



decisiones para que la alarma funcione correctamente. Cabe aclarar que este microcontrolador es previamente programado por alguien para que cumpla la función deseada.

A este microcontrolador se le conectan dos dispositivos transductores. Un sensor de movimiento (3) y un altavoz (4). Como dice su nombre, el sensor de movimiento es un transductor del tipo sensor.

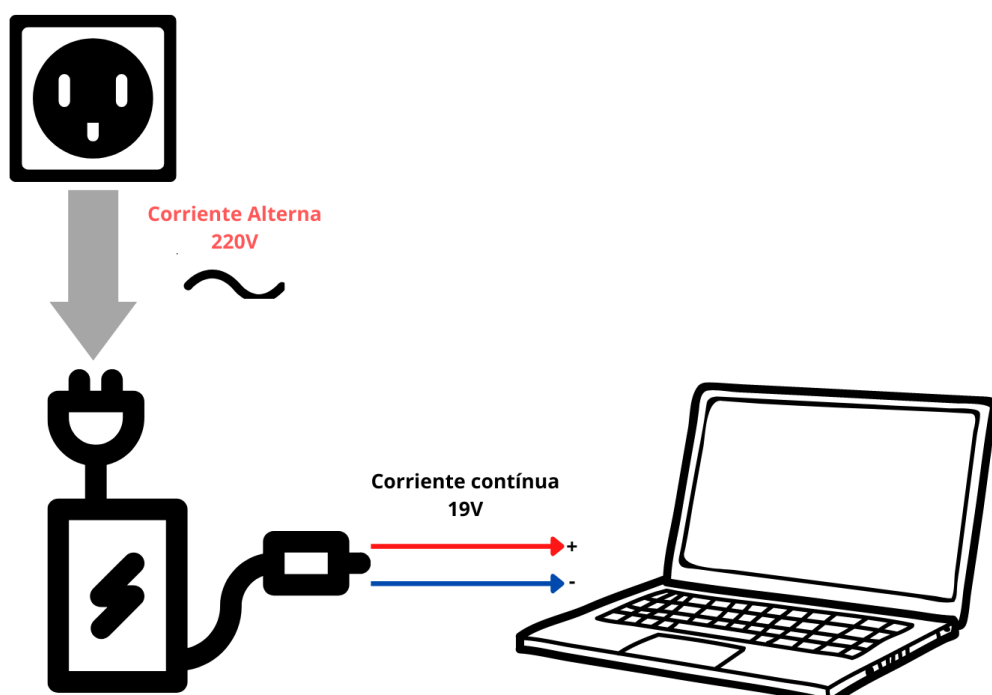
¿Cómo funciona este sensor?

Cada vez que detecta movimiento en un rango acotado de distancia, capta esa información, la convierte en energía eléctrica y la envía hacia el microcontrolador. Este último recibe esa información para tomar decisiones. Por ejemplo, podría programarse para que cumpla la siguiente función: “Cada vez que se detecte movimiento, se encenderá la alarma sonora”. De esta manera, cada vez que el microcontrolador reciba la información del sensor de movimiento, este encenderá el altavoz que funciona como explica en el video compartido anteriormente “Como funciona un micrófono y un altavoz”.

Tema 2.3- Electricidad y circuitos eléctricos

En este curso aprenderás a construir circuitos para hacer que la electricidad circule a través de diferentes componentes. Se trata de circuitos en bucles cerrados mediante cables con una fuente de energía (como una batería) y algún componente que haga algo útil con la energía; este componente se suele llamar carga. En un circuito, la electricidad fluye desde un punto con el potencial de energía más alto (normalmente se conoce como el positivo o “+” de la fuente de energía) a un punto con el potencial de energía más bajo. La masa (a menudo representada con el signo “-” o GND) es el punto con el menor potencial de energía en un circuito. En los circuitos a construirse, la corriente eléctrica solo circula en una dirección. Este tipo de circuitos se llaman de corriente directa (DC). Por otro lado, en los circuitos con corriente alterna (AC) la electricidad cambia de dirección 50 veces por segundo. Este es el tipo de electricidad se puede encontrar en un enchufe de la pared. Estos dos tipos de energía tienen distintos usos. Por ejemplo, la corriente alterna suele ser más energética que la corriente directa y sirve para encender electrodomésticos, grandes luces, televisores, el aire acondicionado, etc. En cambio, la corriente directa suele tener menor cantidad de energía y se utiliza en pequeñas cosas, como teléfonos celulares, auriculares inalámbricos, microcontroladores, sensores, mouse de computadora, etc.

En el siguiente gráfico se puede ver los distintos tipos de corriente que se utilizan a diario. Primero, se conecta un cargador de computadora al enchufe de pared. Este enchufe que tiene corriente alterna (AC) de 220 volts (mucho potencia) alimenta el cargador de la computadora, quien se encarga de convertir esa corriente alterna de gran energía a corriente directa (DC) de menor energía (19 volts). Finalmente estos 19V de corriente continua alimentan a la computadora para que pueda funcionar a través de dos cables; el positivo (+) y el negativo o masa (-).



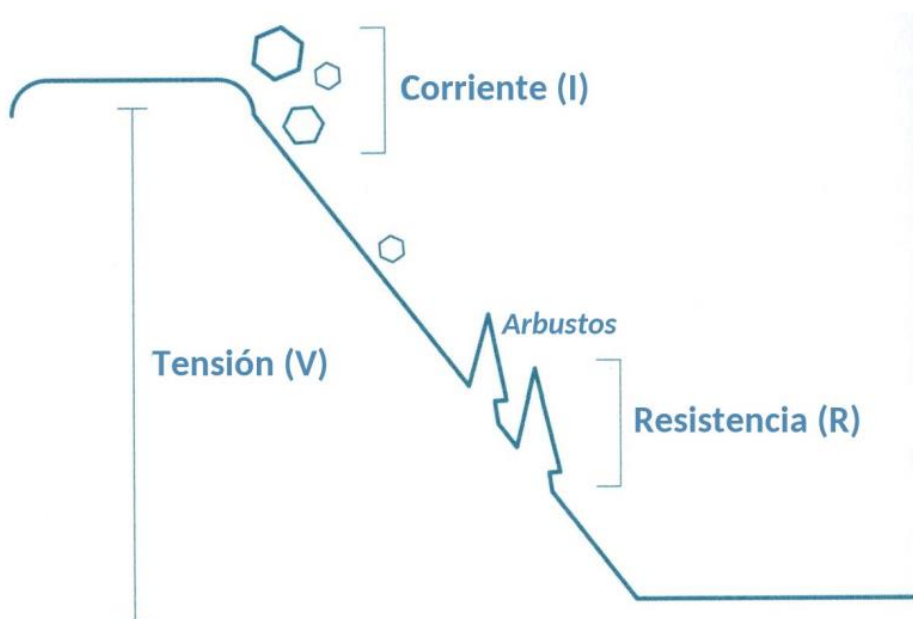
Actividad debate - conceptos sobre electricidad: (básica)

14. ¿Qué pasa si se conecta la computadora directamente al enchufe de pared sin pasar por el cargador?

Puesta en común con conclusiones Existen una serie de términos con los cuales debe de familiarizarse cuando trabaje con circuitos eléctricos.

- Corriente (medida en amperios, o amps; con el símbolo A) es la cantidad de energía eléctrica que circula a través de un circuito.
- Tensión (medido en voltios; con el símbolo V) es la diferencia de energía entre un punto de un circuito y otro que se toma como referencia.
- Resistencia (medida en ohmios -u ohms-; con el símbolo Ω) representa cuanto se opone un componente a que la energía eléctrica fluya a través de él.

Una forma de entender cómo funciona todo lo explicado con anterioridad es imaginar un acantilado con una pendiente por donde se deslizan piedras cuesta abajo, como se muestra el gráfico. Mientras más grande sea el acantilado, mayor energía tendrán las rocas para llegar hasta el suelo. La altura del acantilado es como la tensión (V) en un circuito: cuando mayor es el voltaje en la fuente de energía, mayor energía puede usar. Cuanto más rocas tenga, más cantidad de energía se desplaza hacia abajo por el acantilado. El número de rocas es como la corriente (A) en un circuito eléctrico. Las piedras al bajar por la pendiente del acantilado chocan con los arbustos y pierden algo de energía al aplastarlos y poder pasar por encima de ellos. Los arbustos son como las resistencias en un circuito (Ω), ofreciendo una oposición al paso de la electricidad a la vez que la convierten en otras formas de energía (como calor, sonido, etc).



A tener en cuenta

En un circuito es necesario que exista un camino desde la fuente de energía (alimentación) hasta el punto de menor energía (masa). Si no existe un camino por donde la energía se pueda mover, el circuito no funcionará. Es decir, el camino de la energía debe ser un circuito cerrado.

- La corriente eléctrica siempre busca el camino de menor resistencia hacia la masa (o el negativo). Si existen dos caminos posibles, la mayoría de la corriente eléctrica circula por el camino con menor resistencia.

- Si dispone de una conexión en donde se conectan los puntos de alimentación y masa juntos directamente y sin resistencia, se producirá un cortocircuito; la corriente será demasiado grande al no disponer de una resistencia que reduzca su valor. En un cortocircuito, la fuente de alimentación y los cables convierten la energía eléctrica en luz y calor, se producirán chispas o una explosión. Si alguna vez has cortocircuitado una batería y has visto chispas sabrás lo peligroso que puede ser un cortocircuito.

Tema 2.4- Ley de Ohm

Existe una ley que relaciona todos los conceptos explicados anteriormente y es la ley por la cual se rigen todos los circuitos electrónicos. Esta ley establece una relación matemática entre voltaje (V), corriente (A) y resistencia (Ω).

$$V = I \cdot R$$

Esta ecuación puede descomponerse en dos ecuaciones más:

$$I = \frac{V}{R}$$

$$R = \frac{V}{I}$$

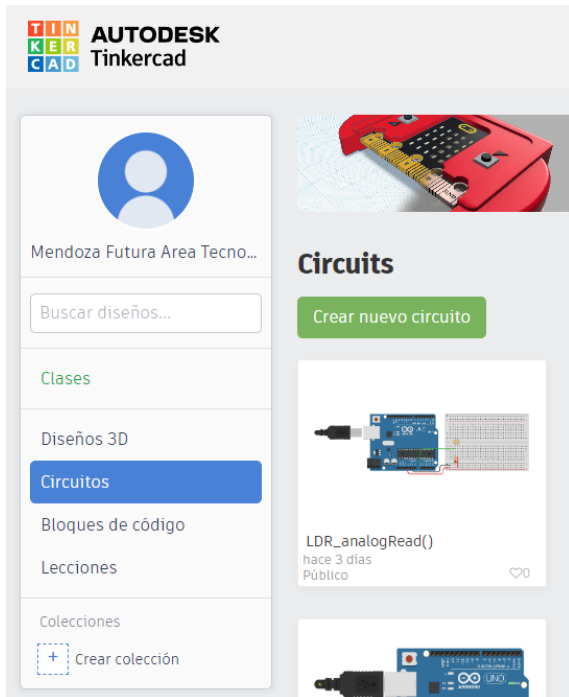
I

Estas ecuaciones indican que mientras mayor sea la resistencia, menor será la corriente que circule por el circuito y viceversa. Esto tiene sentido, pues recordando el ejemplo de las piedras y el acantilado, mientras mayor arbustos o árboles haya en la pendiente, las rocas perderán energía y se dispersarán.

Tema 2.5- [Tinkercad](#)

Ahora se abordarán los conceptos teóricos de electrónica explicados anteriormente a través Tinkercad y de una forma práctica.

Tinkercad es una plataforma para hacer diseños 3D y simular circuitos electrónicos, sensores y microcontroladores. En esta última parte se desarrollarán los contenidos de electrónica y robótica.



La gran ventaja de Tinkercad es que al ser todo simulado, ¡podemos quemar cuantos componentes queramos y tenemos nuestro laboratorio de electrónica en todos lados!

Antes que nada, hay que crear una cuenta ([ver tutorial](#)).

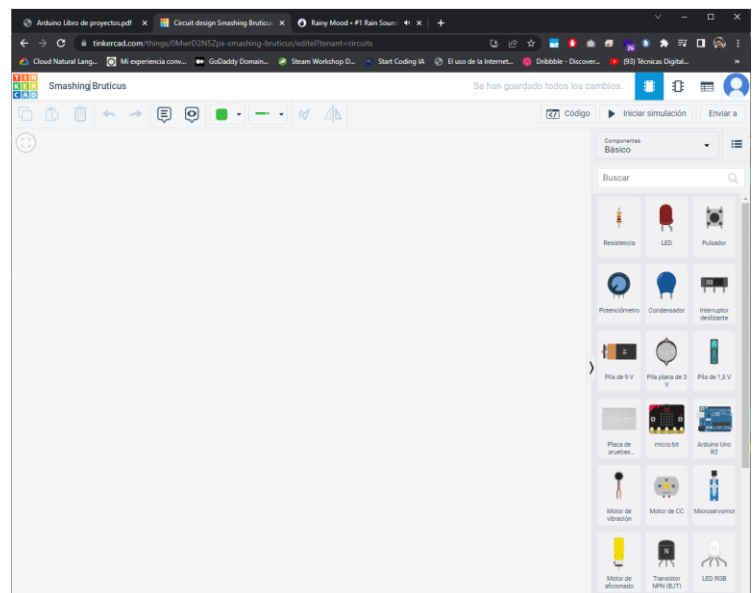
Una vez creada la cuenta, hacemos click en la pestaña “Circuitos” y luego en “Crear nuevo circuito”.

Inmediatamente después, podemos ver el espacio de trabajo de la plataforma. A la derecha del logo se encuentra el título del circuito a desarrollar.

Haciendo click en él, se puede cambiar el nombre.

En el panel de la derecha, hay una pestaña de “Componentes”.

Podemos filtrar por “Básico” y “Todos”. Para explorar la herramienta, seleccionamos “Todos”.



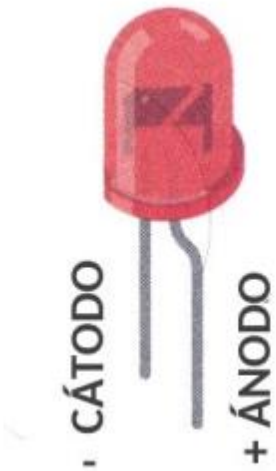
Podemos ahora arrastrar cualquier elemento y soltarlo sobre el espacio en blanco de trabajo.

Debajo del nombre, hay varios botones. Una vez seleccionado un componente, éste se puede copiar y luego pegar en otro lugar del espacio de trabajo. Puede eliminarse seleccionando y presionando el ícono de basura. Si cometimos un error y queremos volver hacia atrás, se puede deshacer una acción presionando el ícono de la flecha hacia la izquierda. Lo mismo se cumple para rehacer una acción.

Componentes básicos

Ahora veremos más elementos de circuitos electrónicos y comprenderemos en profundidad cómo funcionan.

LED



Un LED, o diodo emisor de luz , es un componente que convierte la energía eléctrica en energía luminosa. Los LEDs son componentes que tienen polaridad, esto quiere decir que sólo circula corriente a través de ellos en una sola dirección.

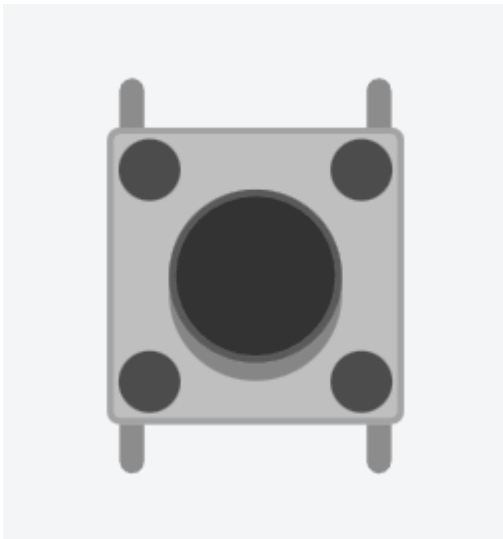
El terminal más largo del LED es llamado ánodo, se conectará a la alimentación. El terminal más corto es el cátodo y se conectará a masa. Cuando la tensión es aplicada al ánodo del led y el cátodo está conectado a masa, el LED emite luz.

Resistencia



Una resistencia es un componente que se opone al paso de la energía eléctrica (posee un código de colores para). Transforma parte de la energía eléctrica en calor. Si se coloca una resistencia en serie con un componente como un LED, el resultado será que el diodo led recibe menos energía al consumir la resistencia esa energía que el LED no recibe. Esto permite poder alimentar a los componentes con la cantidad de energía que necesitan. Puede usar una resistencia en serie con un LED para evitar que reciba demasiada tensión.

Pulsador

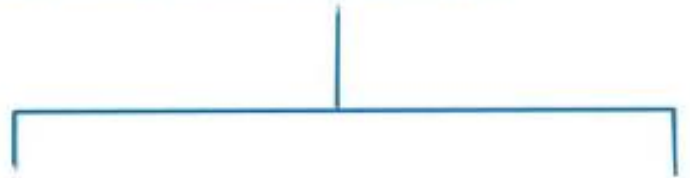


Un pulsador interrumpe la circulación de la electricidad, abriendo y cerrando el circuito. Cuando un interruptor está cerrado, permite que el circuito se alimente.

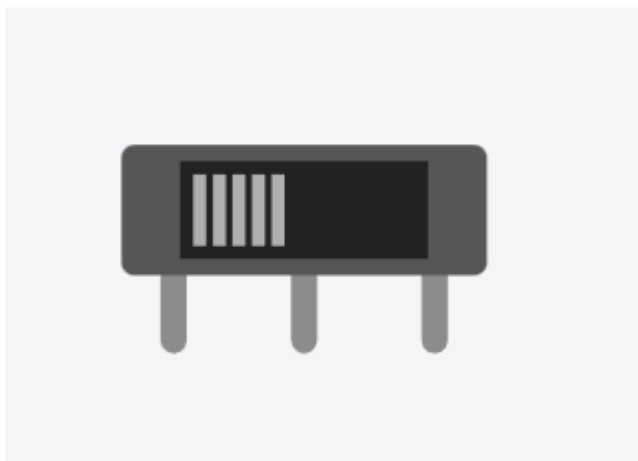
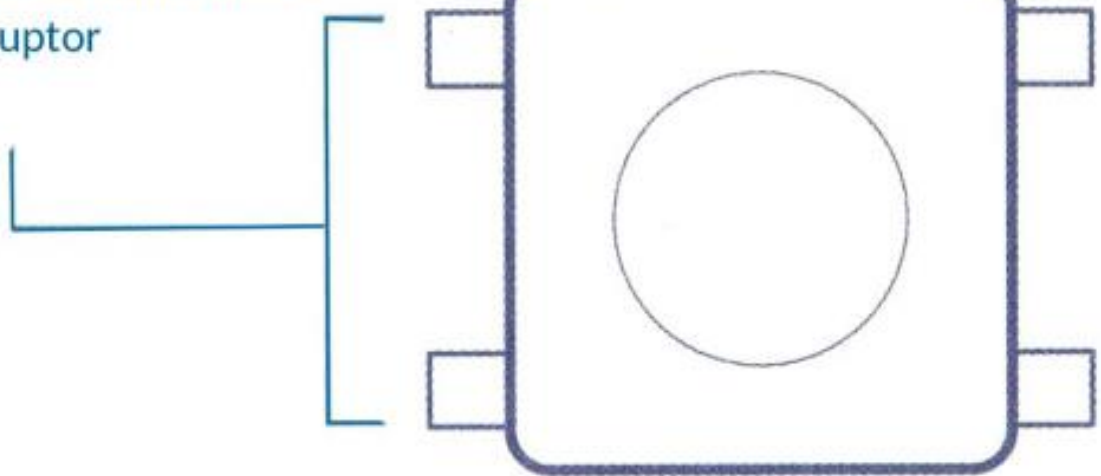
Hay muchos tipos de interruptores, pero todos tienen la misma función.

CONEXIONES DEL INTERRUPTOR

Estos dos terminales de un interruptor están conectados entre sí



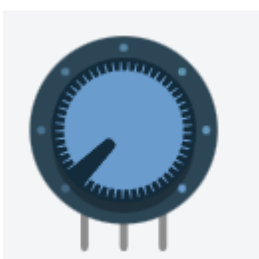
Estos dos no están conectados.
Forma el interruptor



Switch

Es un interruptor. Cumple la misma función del pulsador, solamente que éste tiene un botón deslizante que permite seleccionar entre dos estados.

Permite hacer contacto entre alguno de los extremos y el terminal común del medio.



Potenciómetro

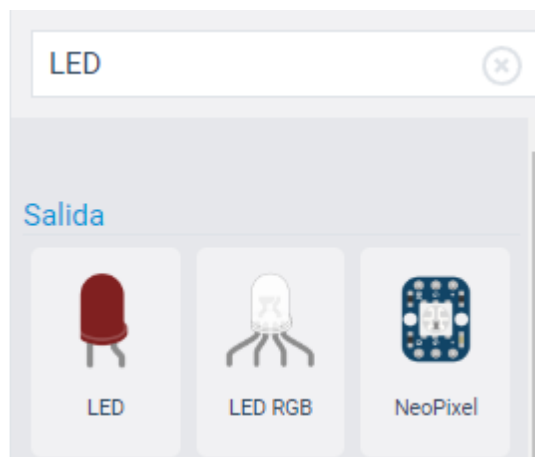
Es una resistencia variable. Tiene un dial que gira, permitiendo seleccionar un valor de resistencia entre cero y el máximo (depende de cada potenciómetro). Se conecta igual que una resistencia como las ya utilizadas, con los conectores de un extremo y medio.

Mi primer circuito electrónico

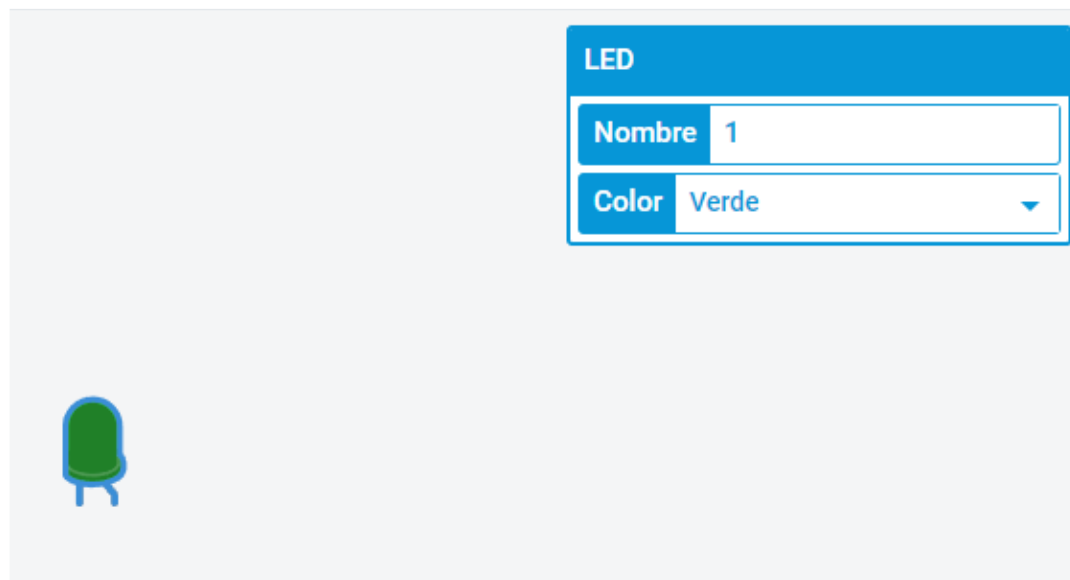
¿Te gusta quemar cosas? ¡Para nuestro primer circuito electrónico, quemaremos componentes y luego veremos cómo arreglarlo para que sobrevivan!

Una luz LED es un pequeño foco que requiere muy poca energía para activarse. Para encenderlo, sólo necesitamos una fuente de alimentación y un par de cables.

Primero, buscamos el LED en la barra lateral de componentes. Lo seleccionamos y lo llevamos al espacio de trabajo.



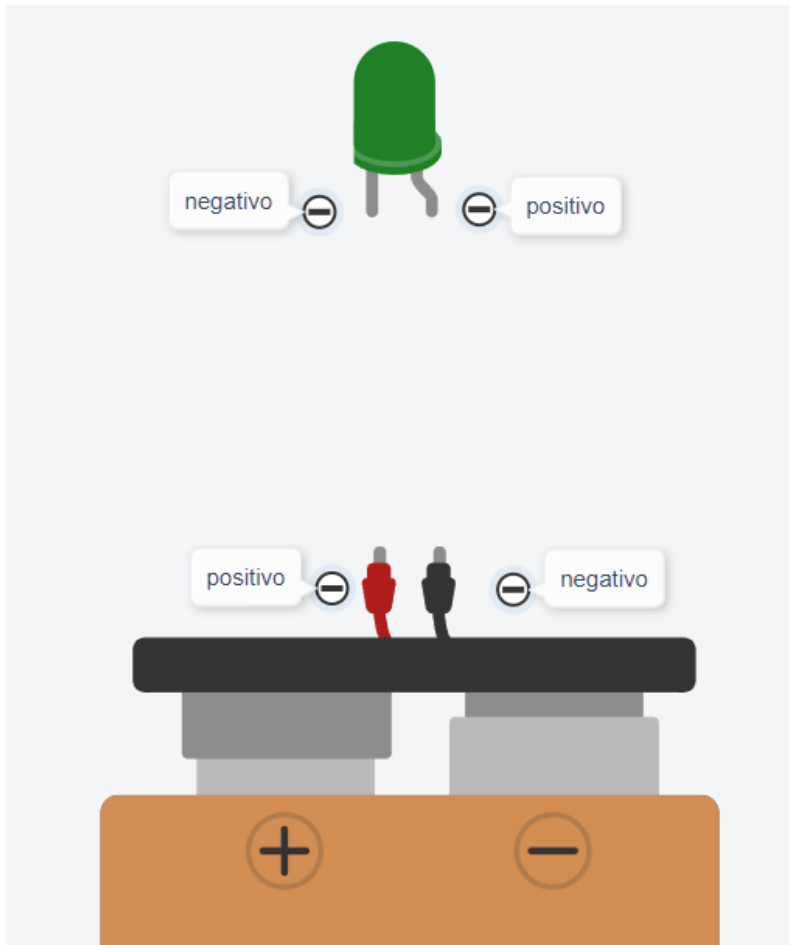
Si presionamos el LED, vemos que se despliega un menú de opciones. Allí se pueden cambiar los atributos del LED, como su color y su nombre.



Luego, buscamos una pila. Llevamos la pila de 9V al lado del LED.

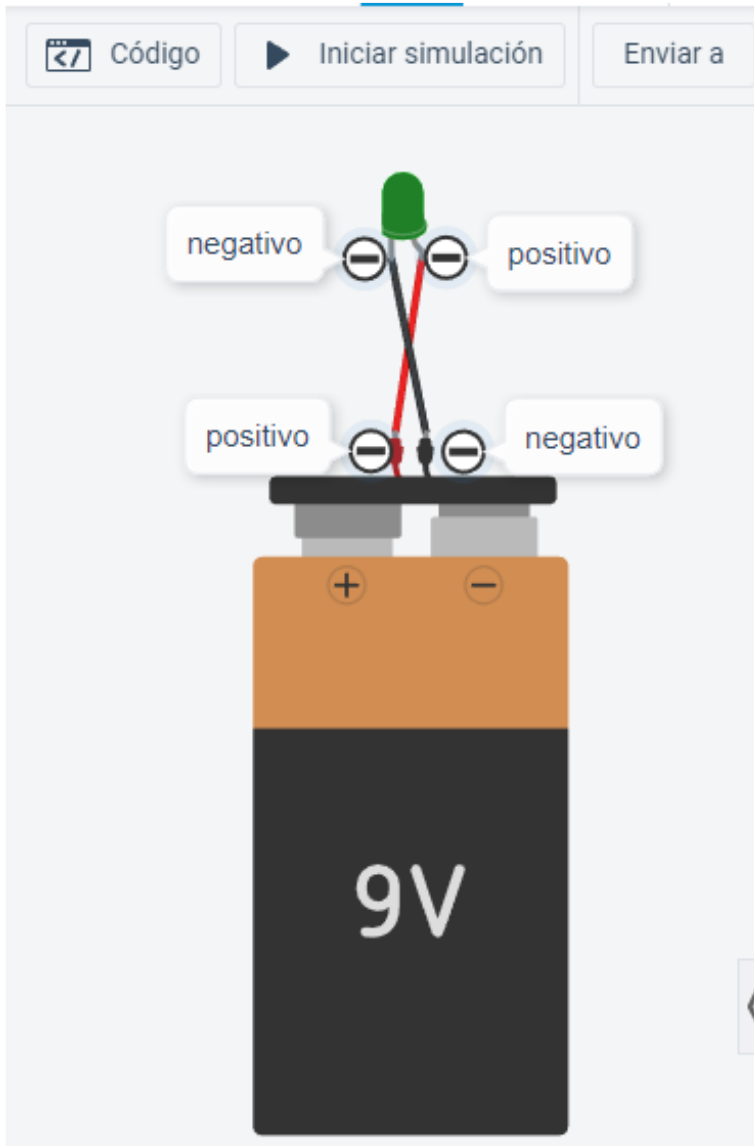


Podemos girar la pila seleccionándola y presionando el botón de girar en la barra de tareas de arriba.



Es hora de conectar el LED a la pila. Como se explicó en la teoría, todos los dispositivos electrónicos se alimentan de energía a través de un cable positivo y uno negativo (o masa). En el caso del LED, el terminal positivo (ánodo) es el más largo, y el terminal negativo (cátodo) el que no lo está.

Sólo resta unir los respectivos terminales e iniciar la simulación. Nótese que cuando unimos dos terminales con un cable, en la barra de tareas de arriba podemos cambiarle el nombre. Para más comodidad, de ahora en adelante los cables rojos serán los de alimentación positiva y los cables negros los de alimentación negativa.



Una vez conectado, clickeamos “Iniciar simulación” y observamos qué sucede.



Aquí es donde más se nota la ventaja de usar simulaciones, porque de conectarlo así en la vida real, el LED se hubiera roto. Si pasamos el mouse arriba del LED, vemos un cartel que nos da una pista, indicando que la corriente que circula (915 mA) es mucho mayor a la que puede soportar (20 mA).

Veremos a continuación qué significa esto.

Primero, ¿qué es un “mA”?

Es una unidad de medición de corriente. Recordemos que la corriente se mide en Amperes, y mA significa mili Amperes, es decir, una milésima parte de un Ampere.

Es lo mismo decir 20 mA que 0.02 A , pues

$$20\text{mA} = 20$$

$$1000\text{ A} = 0.02\text{ A}$$

$$20\text{mA} = 20/1000\text{ A} = 0.02\text{ A}$$

Recordemos que se divide por 1000 porque “mili” se refiere a la milésima parte.

Ahora, ¿por qué se rompió el LED?

Como dice el cartel, circula mucha más corriente a través del LED de la que debería. Esto produce que internamente el LED se sobrecaliente y (en el peor de los casos) explote o deje de funcionar. Esta ley se cumple para todos los dispositivos electrónicos. Si los alimentamos con más energía de la que soportan, se quemarán.

¿Cómo puedo solucionarlo?

Todos los circuitos electrónicos se rigen por la ley de OHM. Tinkercad dice que la corriente máxima de ese LED es de 20mA o 0.02 A . También se que la fuente de alimentación (pila) genera 9 V de tensión. Con estos datos, puedo averiguar la resistencia que necesita el LED para no quemarse.

¿Cómo funciona una resistencia eléctrica < que es una resistencia eléctrica?

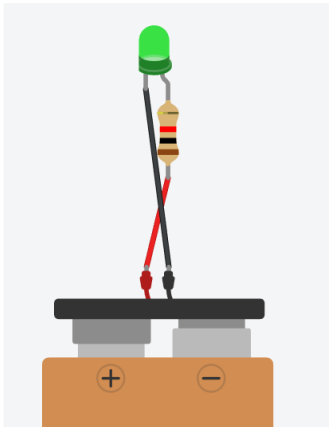
$$R = V/I$$

$$R = 9\text{ V} / 0.02\text{ A} = 4500\ \Omega$$

En conclusión, se debe agregar una resistencia al circuito entre la pila y el LED de **al menos** $4500\ \Omega$. Si la resistencia es mucho mayor, el brillo del LED será menor, pues menos energía tendrá.

Para agregar la resistencia o cualquier otro componente se debe detener la simulación.

Queda de la siguiente manera, y se puede observar que el LED enciende sin explotar.



Protoboard - placa de pruebas

Cómo usar una Protoboard

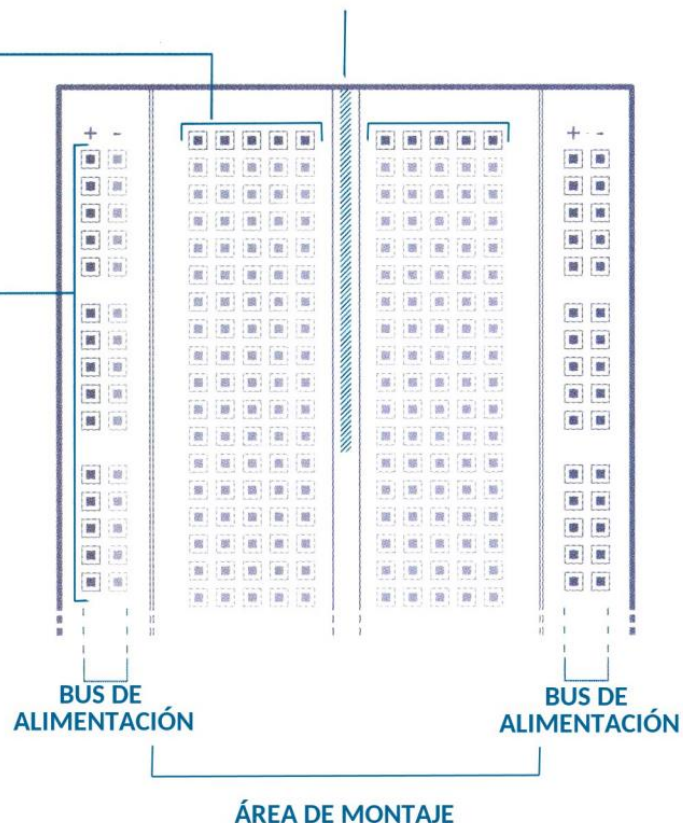
La placa de pruebas es el primer lugar en donde se crearán los circuitos. Tiene el propósito de ordenar los circuitos, pues mientras más componentes tenga, más cables tendrá.

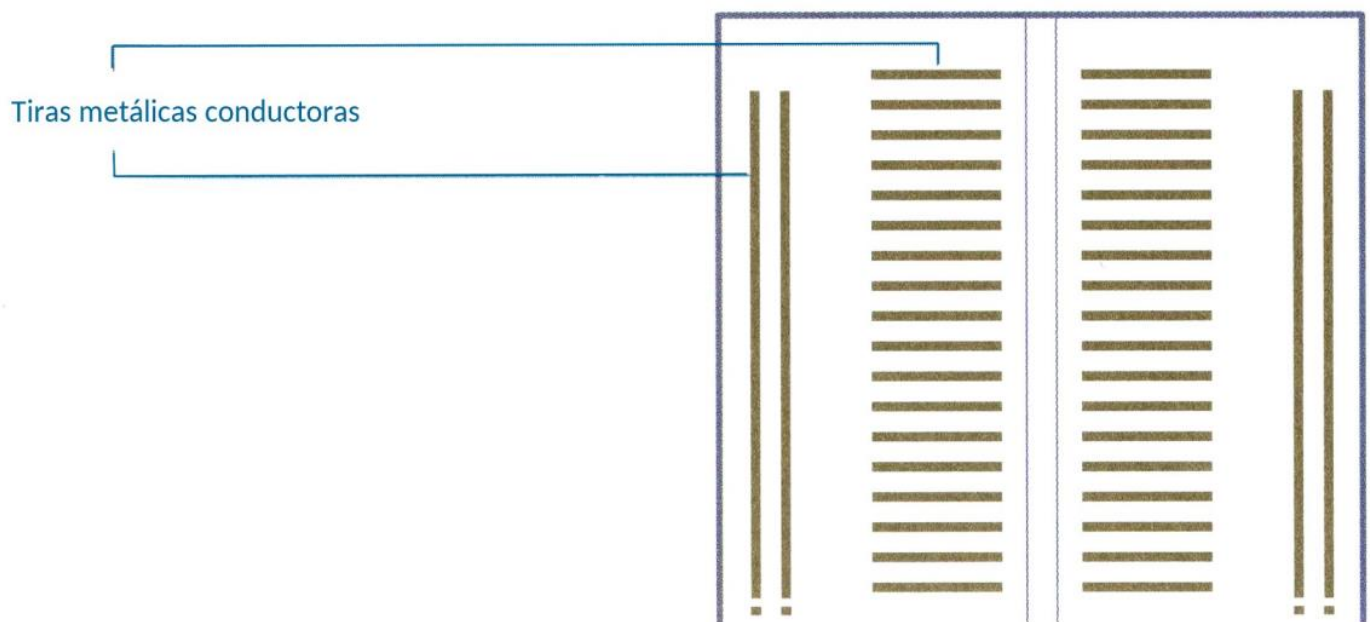
Las filas verticales y horizontales de la placa de pruebas, como se muestra en la imagen, conducen la electricidad a través de los conectores de metal fino que hay debajo del plástico con agujeros.

Los 5 agujeros de cada fila horizontal están conectados eléctricamente a través de las tiras de metal en el interior de la placa de pruebas

La fila del medio rompe la conexión entre los dos lados de la placa

Las tiras verticales que recorren toda la longitud de la placa están eléctricamente conectadas. Estas tiras se suelen usar para las conexiones de alimentación y masa.

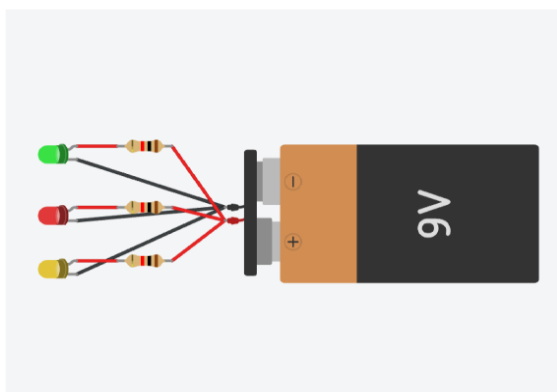




Pero... ¿Realmente es necesario usarla?

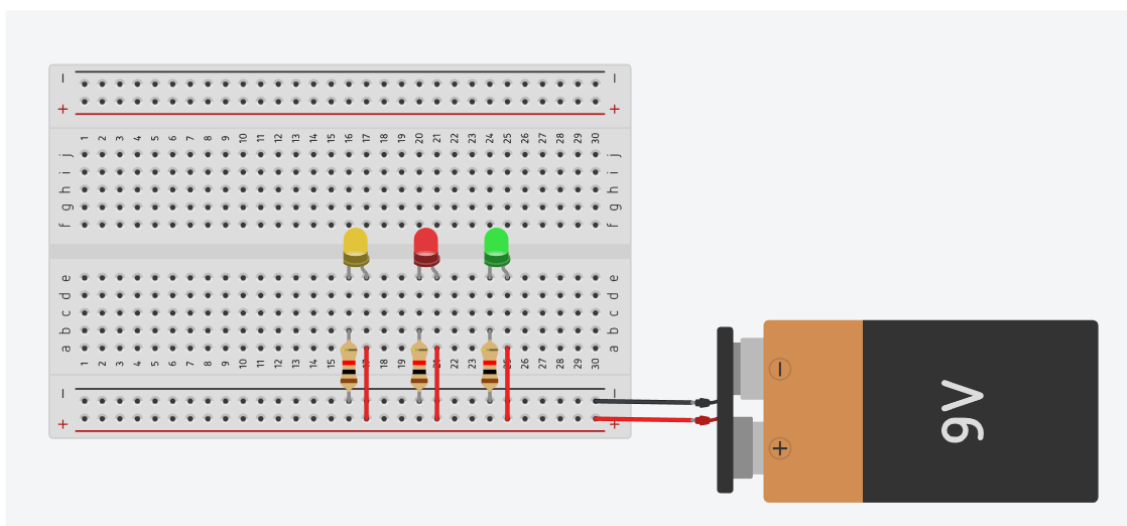
Se responderá a esta pregunta a través de un ejemplo práctico en Tinkercad. Haremos un circuito “sencillo” muy similar al primer circuito. Consiste en encender 3 LEDs, uno amarillo, uno verde y uno rojo, con una pila de 9V.

Primero lo haremos sin la placa de pruebas.



Si bien funciona, los cables se ven muy desordenados y es más probable equivocarse y más difícil revisar si se ha cometido un error y se quiere cambiar algo.

Ahora, se hace el mismo circuito pero utilizando la placa de pruebas.



Se puede observar mucha más claridad en el circuito, y resulta mucho más sencillo agregar componentes a la placa.

Actividades: componentes básicos

15. Actividad: En una protoboard, colocar un LED, una pila de 9V y un pulsador para que el LED sólo encienda cuando el mismo sea presionado. Explica con tus palabras qué está sucediendo

16. Actividad complementaria: Cambiar el pulsador del ejercicio anterior por un switch, y hacer que el LED encienda cuando el switch esté hacia la derecha.

17. Actividad prioritaria: Reemplazar la resistencia del LED correspondiente al ejercicio 1 por un potenciómetro de 150 k Ω (150.000 Ω)* utilizando el conector del extremo izquierdo y medio. Una vez simulado, mover el potenciómetro desde la izquierda hasta la derecha y observar qué sucede con el brillo del LED.

*nota: 150 k Ω es lo mismo que 150.000 Ω (ciento cincuenta **mil** Ohms). Para modificar el valor, hay que seleccionar el switch y escribir sobre la ventana que se abre.

Potenciómetro	
Nombre	2
Resistencia	10 k Ω ▼

18. Actividad: Realizar el ejercicio 3, pero en vez de utilizar el extremo izquierdo, utilizar el derecho. Observar qué sucede con el brillo del LED mientras se mueve el dial.

19. Actividad complementaria: Hacer un circuito que encienda un LED amarillo cuando se presione un pulsador **y** un switch esté hacia la derecha. Observar qué pasa si se presiona el botón cuando el switch está hacia la izquierda.

20. DESAFÍO!!

En el laboratorio del Banco de Sangre separan los componentes de la sangre para los distintos usos y pacientes.

Se necesita diseñar un circuito eléctrico que permite controlar dos motores, uno que vibra y otro que centrifuga. Cuenta con dos llaves, una para encender el sistema completo y la otra que elige qué motor se desea utilizar. El motor centrifugado cuenta con un potenciómetro para elegir la velocidad, el motor de vibración siempre vibra al máximo.

[Podrás acceder a las respuestas actividades componentes básicos en el siguiente link:](#)

<https://bit.ly/3INeAG9>

Introducción al microcontrolador Arduino

Arduino Uno – Es la tarjeta de desarrollo del microcontrolador Atmega328p y será el corazón de tus proyectos. Arduino es una computadora muy pequeña y básica, con la que construiremos circuitos e interfaces para hacer proyectos y decirle al microcontrolador como trabajar con otros componentes como sensores y actuadores.

¿Qué es Arduino?

Arduino es una plataforma OpenSource de desarrollo de hardware y software, que contiene una mini computadora (microcontrolador) que puede programarse para que cumpla una función determinada. Permite la conexión de sensores y actuadores para sensar/intervenir en el ambiente inmediato al controlador. Los sensores captan información del entorno, y los actuadores generan un cambio físico en el entorno. Por ejemplo, se le conecta un micrófono y una luz LED. Se programa de forma tal que si detecta un sonido fuerte a través del micrófono, se encienda la luz.

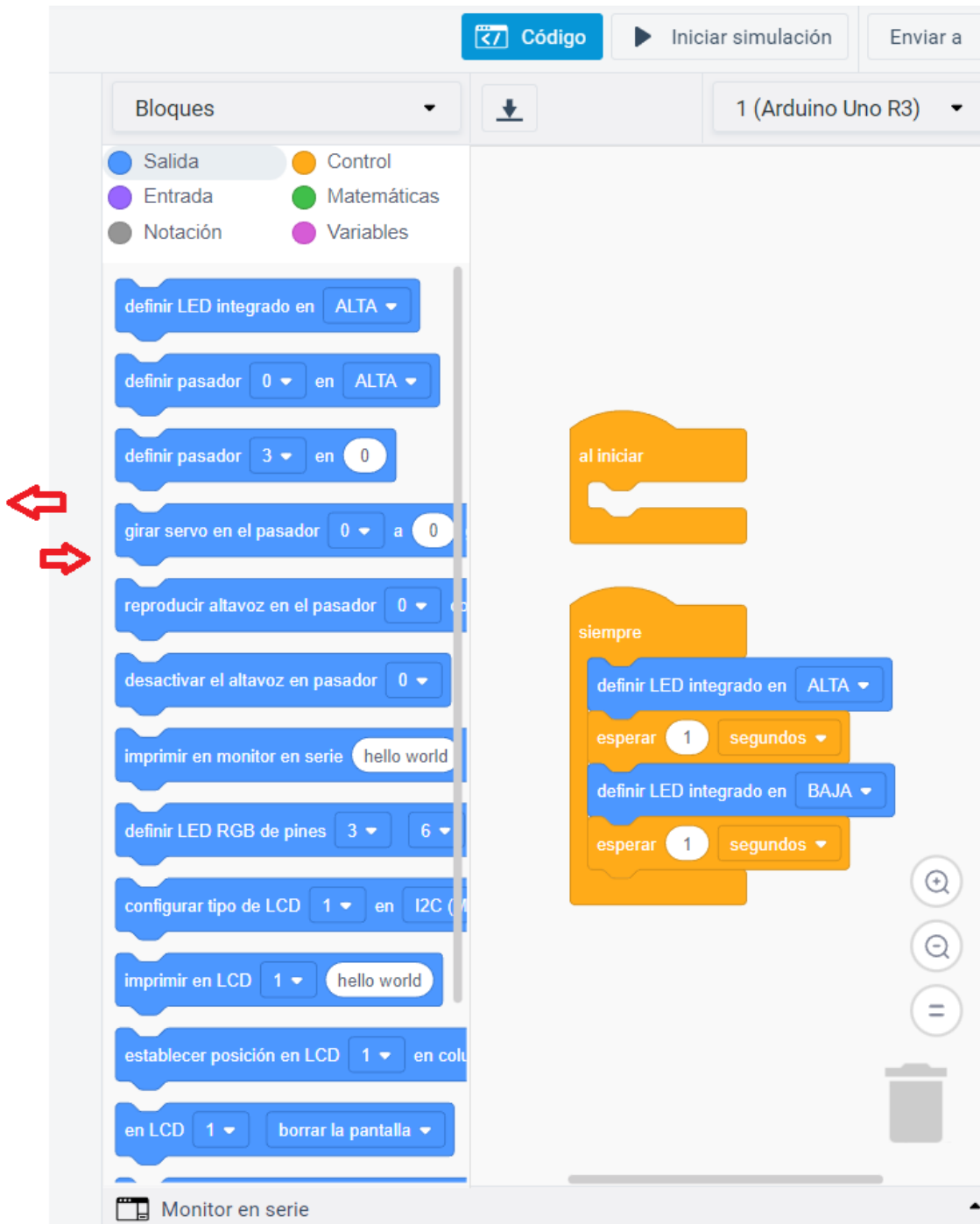
Como resultado, se obtiene un **autómata**. Este proceso puede complejizarse para obtener aplicaciones reales y muy útiles de robótica o dispositivos autónomos.

Programando Arduino

Creando un nuevo circuito en Tinkercad, agregaremos un Arduino UNO al espacio de trabajo.



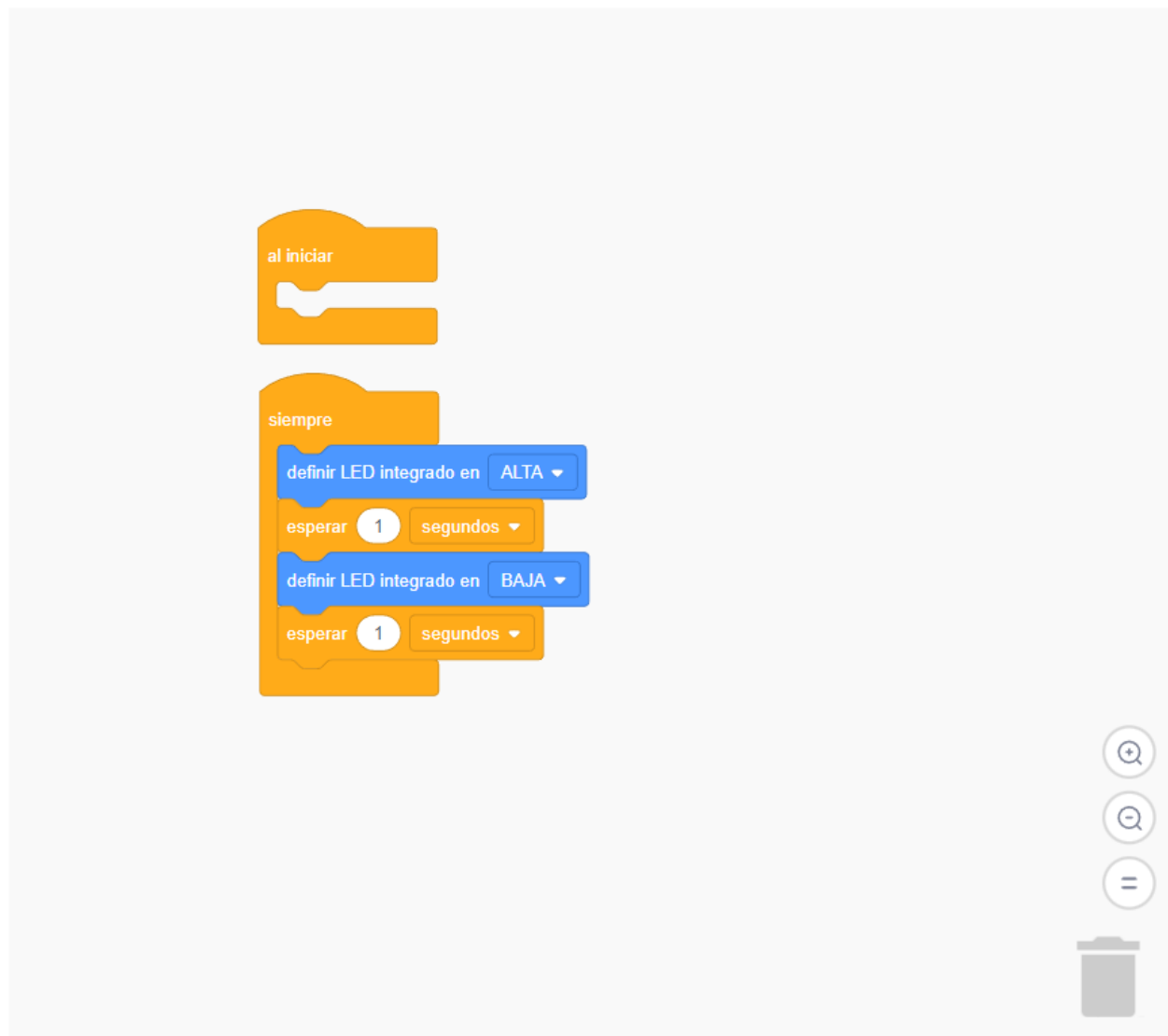
El botón “Código” abre un panel lateral. Este panel puede agrandarse si se posiciona el mouse sobre el costado izquierdo del panel.



Ahora nos encontramos con un entorno conocido, casi idéntico al de mBlock, esto nos proporciona una ventaja, ya sabemos generar programas en mBlock, por lo tanto estamos a un paso de generarlos para Arduino.

Al igual que en mBlock el programa nos muestra bloques a la izquierda que podemos arrastrar y agregar al espacio de programación a la derecha para armar el programa deseado en Arduino.

Inicialmente muestra un código por defecto.



Pueden borrarse bloques arrastrandolos hacia el ícono de basurero.



Existen 3 opciones para programar Arduino: bloques, bloques + códigos y código. La programación en bloques tiene una gran similitud con la herramienta usada en el Episodio 1: “mBlock”.

MODO DE EDICIÓN

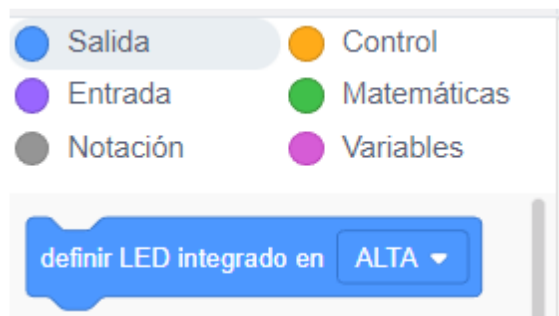
Bloques

Bloques + Texto

Texto

Se verá el siguiente desarrollo en “Bloques + Texto”.

Se puede seleccionar entre distintos tipos de bloques. Cada uno cumple una función. Ahora se explorarán las distintas funciones que tiene Arduino.



Estructuras de control

Son funciones que pueden ejecutarse al principio del programa, una y otra vez o un número determinado de veces.

El setup y el loop

Cada programa de Arduino tiene dos funciones básicas principales. Las funciones son partes de un programa de ordenador que ejecuta instrucciones específicas. Las funciones tienen un único nombre y son “**llamadas**” cuando se necesitan. Estas dos funciones principales en un programa de Arduino son llamadas con **setup()** y **loop()**, las cuales vienen declaradas por defecto en cualquier script de Arduino.

Al colocar una placa Arduino en Tinkercad se habilita la opción de “Código” en la sección derecha de la herramienta de simulación.



```
1 //Espacio para declarar variables
2
3
4 void setup()
5 {
6     //Configuración de Pines
7     //Código que se ejecuta una vez
8 }
9
10 void loop()
11 {
12     //Código principal en constante repetición
13 }
```

“**Al iniciar**” es la función llamada “**setup()**”, tiene la tarea de configurar los pines digitales de arduino en entradas o salidas, inicializar comunicación Serial y ejecutar cualquier declaración o inicialización que desea realizarse **una sola vez al encender Arduino**. Esta función no vuelve a ejecutarse, solo se ejecuta al encender o re-startear Arduino (la placa posee un pulsador que permite reiniciar su funcionamiento con cortar la alimentación).

Esta función define un bloque, el cual se abre con una llave “{” y se cierra con otra llave “}”, en donde se escriben las instrucciones que configuran los pines digitales de Arduino como entradas o como salidas, usando para ello una función llamada pinMode(). Los pines conectados a los LEDs serán OUTPUTs (salida) y el pin de un pulsador será una INPUT (entrada).

La función “**siempre**” o “**loop()**” se ejecuta inmediatamente después de que la función setup() se haya completado.

Esta función se ejecuta una y otra vez. Si queremos realizar una acción continua en el tiempo, al colocarla dentro del loop lograremos que sea ejecutada muchas veces por segundo, dando la apariencia de que se está ejecutando todo el tiempo (de forma continua y sin pausas).

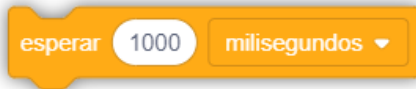
Esperar o delay



```
void loop()
{
    delay(1000); // Espera 1000 milisegundos
}
```

Como bien lo dice su nombre, sirve para que el programa espere o se detenga por un determinado tiempo. Es importante saber que durante el tiempo que dure esta instrucción, el Arduino no realiza ninguna acción y espera sin hacer nada.

El parámetro de la función delay() es un número en milisegundos. Es decir, para que el programa se detenga 5 segundos, el argumento debe ser 5000. Si se quiere que esté detenido medio segundo, la instrucción es delay(500).

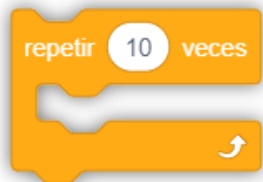


es lo mismo que



Repetir o for

Es una función para ejecutar un número determinado de veces las instrucciones colocadas en su interior



```
for (contador = 0; contador < 10; contador++)  
{  
    //realizo alguna acción 10 veces  
}
```

Si, entonces o if

Sirve para tomar decisiones en función de una condición.

Si se cumple tal condición, **entonces** hago una acción. Si no se cumple la condición, el programa saltea este bloque de código.

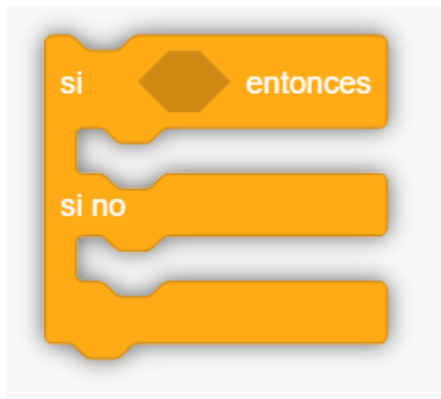


```
if(condicion)  
{  
    /*si se cumple la condición, se ejecuta  
    lo que esté entre llaves*/  
}
```

Si, entonces, sino o if, else

Sirve para tomar decisiones en función de una condición.

Si se cumple tal condición, **entonces** hago una acción. **Sino**, hago otra cosa.



```
if(condicion)
{
    /*si se cumple la condición, se ejecuta
    lo que esté entre las llaves del if*/
}
else
{
    /*Si no se cumple la condición, se ejecuta
    el código dentro de else*/
}
```

Contar arriba/abajo o for (repetidor)



```
for (contador = 10; contador > 0; contador -= 1)
{
}
}
```

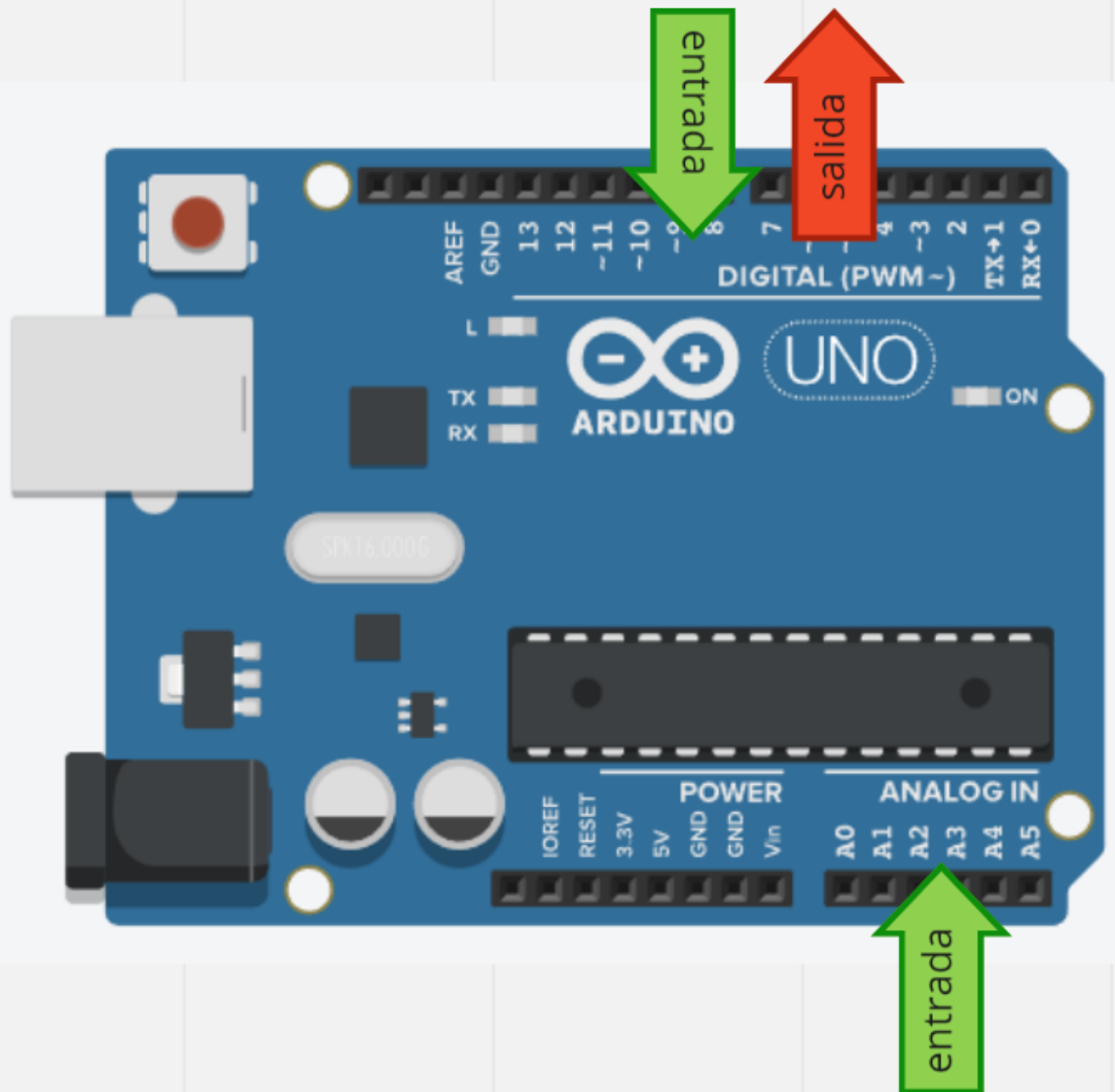
En código, es la misma instrucción que repetir, pero el bloque ahora deja seleccionar si queremos contar “arriba” o “abajo”. Es decir, modifica la variable de contador de menor a mayor o mayor a menor.

Configuración de Pines digitales:

Arduino es un sistema capaz de procesar información que entra en forma de señales eléctricas y generar señales de salida para controlar otros dispositivos.

Para entender lo que es una salida respondamos las siguientes preguntas:

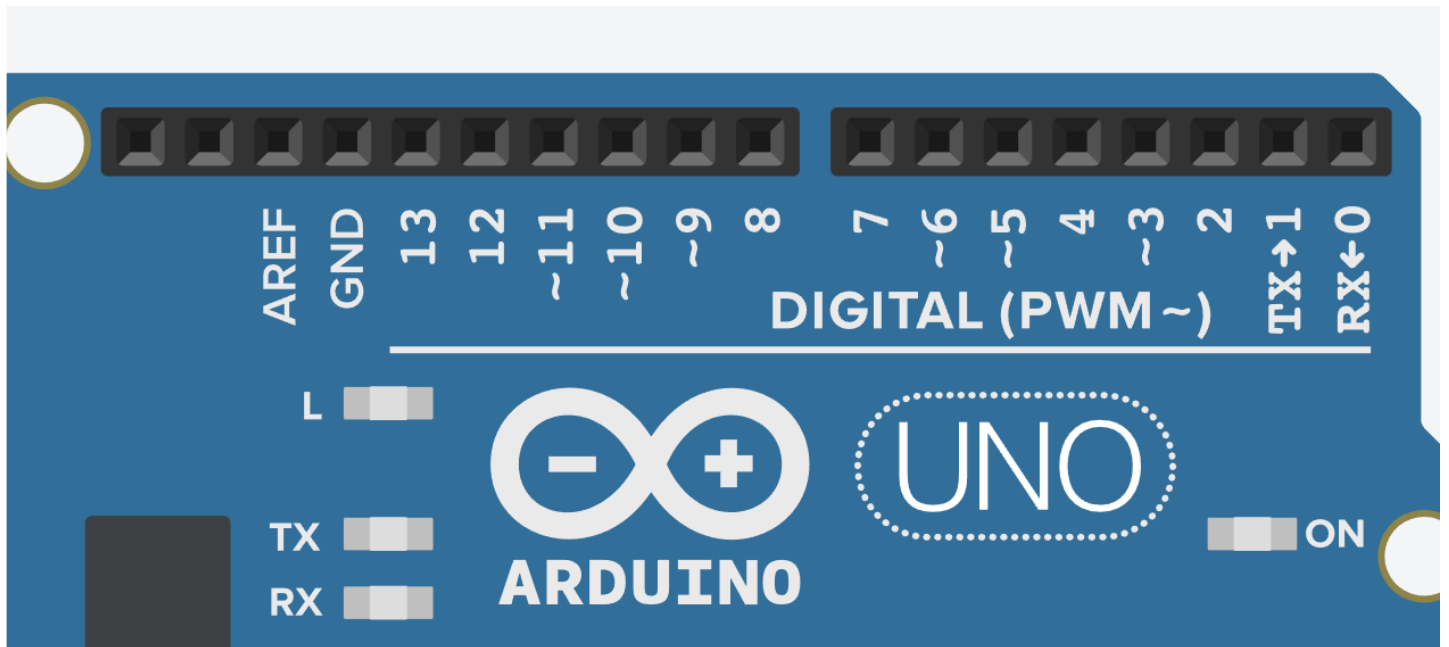
¿Qué es una salida? ¿De dónde sale? ¿Qué sale? ¿Cómo verificamos que la salida funciona?



Para programar Arduino, tenemos que imaginarnos que somos ese microprocesador que estamos programando, programaremos “en primera persona” a Arduino.

Arduino contiene 14 pines digitales los cuales pueden ser configurados como salidas o entradas, según el uso que se les dará.

Una salida es un pin configurado como tal, capaz de emitir o escribir voltajes eléctricos, estos voltajes, al ser digitales, pueden tomar solo dos valores: HIGH (+5v) o LOW (0v). Las salidas son utilizadas usualmente para encender o apagar LEDs pero básicamente podemos controlar cualquier actuador: motores, bobinas, relés, bombillas, etc.



Los pines digitales de Arduino son 14, numerados del 0 al 13. Los pines deben ser configurados como entradas o salidas en el `setup()` para poder ser utilizados, caso contrario no funcionarán.

Primer paso para que un pin digital (del 0 a 13) de Arduino sea una salida es configurarlo como tal en el `setup()` mediante el uso de la función **`pinMode(nro_pin, config)`**, la cual recibe dos parámetros que se colocan entre paréntesis y separan con coma. El **primer parámetro** tiene que ser un **número entero** del **0 al 13** indicando a qué pin hacemos referencia y el **segundo parámetro** hace referencia a si el pin será entrada o salida, en donde para que sea salida se coloca **OUTPUT** y para entrada **INPUT**.

Entonces `pinMode` sirve para configurar si un pin específico se comporta como una entrada (pulsador) o una salida (LED).

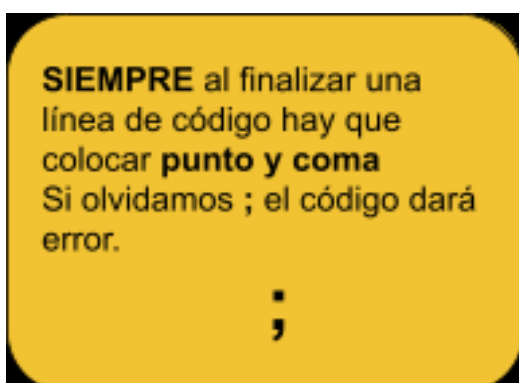
La sintaxis para configurar un pin como entrada (E) o salida (S), es:

```
pinMode(nro_pin, MODO);
```

Donde:

nro_pin: es el número entero de pin de Arduino donde se quiere establecer el modo.

modo: el modo que puede ser `INPUT`, `OUTPUT` o `INPUT_PULLUP`.



Por ejemplo: **pinMode(13,OUTPUT);** establece como salida el pin 13 del microcontrolador. Este pin está asociado a un led de pruebas que viene incorporado en la placa llamado LED_BUILTIN (porque LED_BUILTIN = 13). Las salidas sirven para entregar voltaje al exterior.

Por otro lado: **pinMode(9 , INPUT);** establece como entrada el pin 9 del microcontrolador. Una entrada lee e interpreta señales eléctricas que vienen del exterior, normalmente de un sensor.



En verde la definición de entrada y lectura y guardado del dato en memoria de Arduino. En rojo definición salida y escritura digital en HIGH. **Salidas Digitales:**

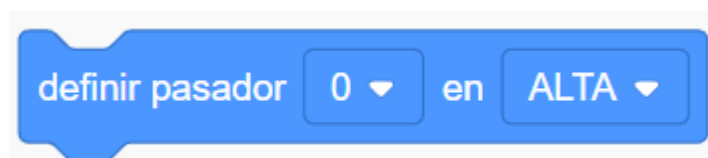
Las salidas se establecen con pines de Arduino configurados como **OUTPUT**.

pinMode(nro_pin,OUTPUT);

Una vez configurado el pin como salida podremos hacer uso de ese pin para encender o apagar dispositivos electrónicos que conectemos en él. El elemento más básico que haremos uso es un LED, por lo que el siguiente ejemplo enciende y apaga un led cada 1 segundo (1000 milisegundos), haciendo uso de las funciones digitalWrite() y delay().

LINK Ejemplo BLINK en TINKERCAD:

<https://www.tinkercad.com/things/bDqElhgGgYQ-blink-bloquescode>



El bloque de código define pin y establece estado del pin a la misma vez. En código se necesita una porción de código en setup() y otra en el loop()

La sintaxis para encender o apagar un pin es:

digitalWrite(nro_pin,estado_pin);

Donde:

nro_pin: es el **numero entero del pin** declarado como salida en el setup que queremos encender o apagar.

estado_pin: es el estado en que escribiremos el pin, las opciones son **HIGH** o **1** para habilitar **5v** en el pin y **LOW** o **0** para escribir un **GND** o **0v** en el pin.

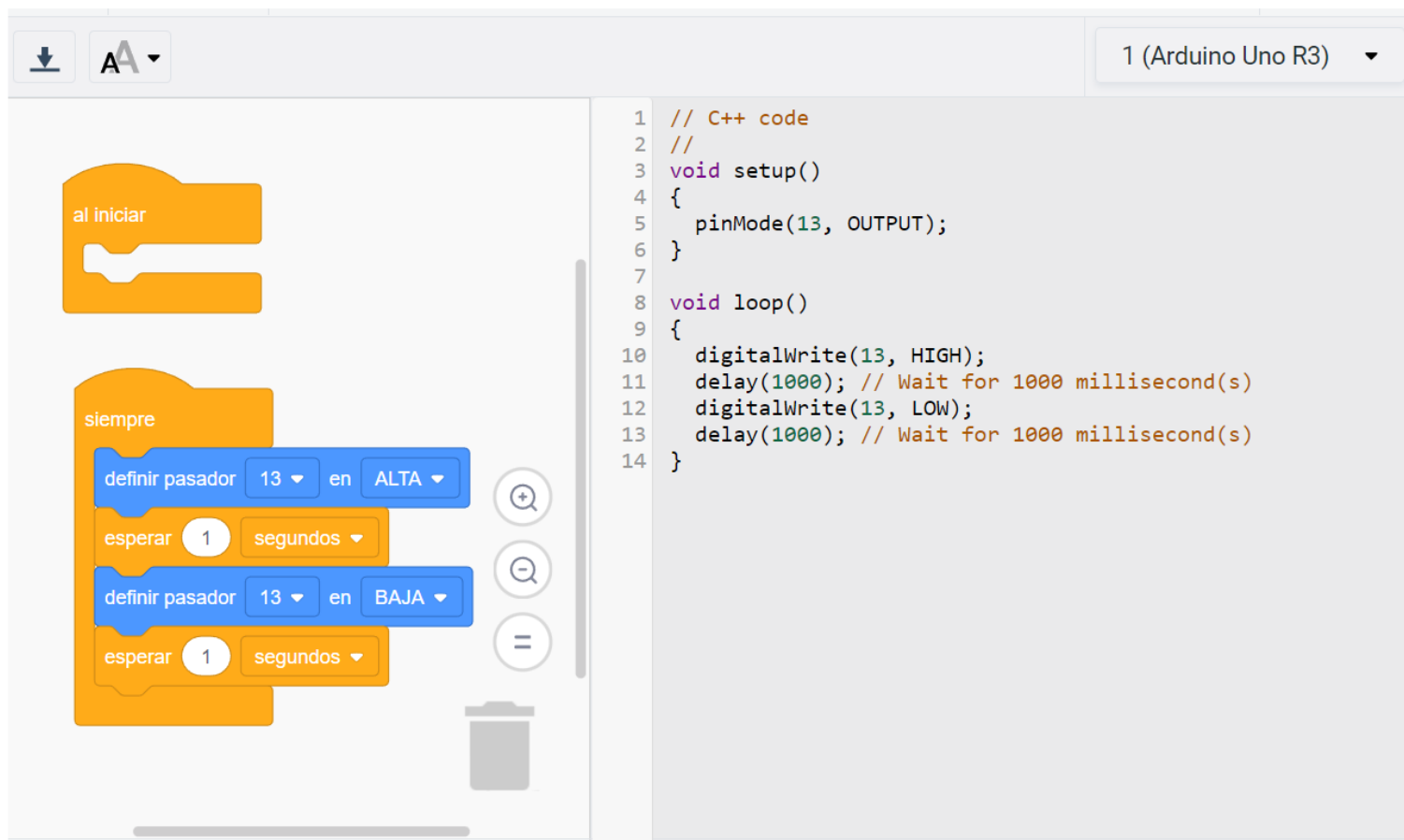
delay(tiempo_en_ms)

Esta función hace que el Arduino se pause el tiempo especificado entre paréntesis. Pausa el programa por la cantidad de tiempo (en milisegundos) especificado como parámetro. Hay 1000 milisegundos en un segundo.

Por ejemplo: **digitalWrite(13,HIGH);** entrega 5v o HIGH en el pin 13. También enciende el led integrado: LED_BUILTIN.

En cambio: **digitalWrite(13,LOW);** entrega 0v o GND en el pin 13. También apaga el led integrado:

LED_BUILTIN.



The screenshot displays the Arduino IDE interface. On the left, the block-based programming environment shows a sequence of blocks: an 'al iniciar' (when started) block followed by a 'siempre' (loop) block. Inside the loop, there are four blocks: 'definir pasador 13 en ALTA' (set pin 13 to HIGH), 'esperar 1 segundos' (wait 1 second), 'definir pasador 13 en BAJA' (set pin 13 to LOW), and another 'esperar 1 segundos' (wait 1 second) block. On the right, the C++ code is shown, corresponding to the blocks. It includes a setup function that sets pin 13 to OUTPUT, and a loop function that alternates between setting pin 13 to HIGH and LOW, with a 1000 millisecond delay between each state change.

```
1 // C++ code
2 //
3 void setup()
4 {
5   pinMode(13, OUTPUT);
6 }
7
8 void loop()
9 {
10  digitalWrite(13, HIGH);
11  delay(1000); // Wait for 1000 millisecond(s)
12  digitalWrite(13, LOW);
13  delay(1000); // Wait for 1000 millisecond(s)
14 }
```

Para poder encender o apagar una salida en Arduino debe haber una conexión que establezca un circuito cerrado entre el +5v y el GND. Caso contrario el dispositivo no sufrirá ningún cambio.

21. *Actividad:* Se necesita desarrollar la parte de automatización de un el uso en carreteras, ¡por suerte no hay peatones! Por eso necesitamos que desarrolles un programa y el circuito para el encendido secuencial de las tres luces (leds) del semáforo, como lo hace un semáforo cualquiera.

¿Te animas a coordinar dos semáforos del cruce de dos calles de simple mano? Ayudate de dos líneas de tiempo en paralelo para explicarles a los alumnos cómo se deben coordinar los semáforos.

22. *Actividad:* En la envasadora de tomates se debe colocar una alarma lumínica roja y blanca, ya que existe mucho riesgo estar cerca de ésta. Los leds rojo y blanco deben encenderse secuencialmente cuando una enciende la otra se apaga y viceversa, dale a la luz blanca un tiempo de permanencia diferente al de la roja.

23. *Actividad:* Realizar un programa que imite las balizas de la policía y a su vez la de los bomberos. Tienen distinta secuencia de encendido. Se busca poder compararlas (cada baliza es comandada por un arduino distinto).

24. *Actividad:* Realizar un programa que haga una secuencia de *ida y vuelta* para el siguiente circuito.

En estas actividades, el facilitador de habilidades blandas puede participar trabajando en la comprensión de consignas, en la resolución de ejercicios - problemas brindando orientaciones sobre cómo poder obtener la información de los que tengo, de lo que me falta, de lo que hay que averiguar, de la obtención de respuestas cerradas o concretas, respuestas abiertas, etc.

Cada situación será acorde a la realidad de cada grupo, por lo tanto este abordaje es abierto y flexible. Sí no se deben olvidar que acompañan y son mediadores en generar “puentes” de conocimiento entre habilidades.

❖ - Variables

Lo primero que hay que hacer antes de meterse en la parte principal del programa es crear una variable (primeras líneas de código) para que estén disponibles para todas las funciones del programa, son llamadas “macro” variables.

Las variables son nombres que se utilizan para guardar información dentro de la memoria de Arduino. Los valores de las variables pueden cambiar dependiendo de las instrucciones que contenga el programa. El nombre de las variables deben de ser una descripción del tipo de información que contienen. Por ejemplo, una variable llamada EstadoBoton, dice lo que está guardando: el estado de un pulsador o botón. Por otra parte, una variable llamada “x” no dice mucho acerca del tipo de información que guarda.

A diferencia de Python, en C++ debemos declarar las variables diciéndole qué tipo de dato serán. Los tipos de datos son los mismos que en Python sólo que tienen pequeñas diferencias en su nombre.

Una variable es un lugar de la memoria de una computadora o de un microcontrolador en donde se almacena información para ser usada por un programa. Las variables almacenan valores los cuales pueden cambiar a medida que un programa se ejecuta. Las variables son de varios tipos según la clase de información que almacena y el tamaño máximo de esa información, por ejemplo, una variable de tipo byte solo puede guardar 256 valores diferentes, pero una variable del tipo int (entero) puede guardar 65.536 valores. Las variables pueden ser locales dentro un determinado bloque de código, o globales para ser usadas en cualquier parte del programa.

Las variables pueden ser de dos tipos según el lugar en que las declaremos:

- globales

- locales.

Variable Global:

La variable **global se declara antes (o afuera) del setup() y del loop()**. Puede ser utilizada en cualquier parte del programa y se destruye al finalizar éste.

Variable Local:

La variable local se declara dentro del loop() o del setup(), en la función en que vaya a ser utilizada.

Sólo existe dentro de la función en que se declara y se destruye al finalizar dicha función. Si declaramos una variable en el setup (), en el loop() no existirá (siempre vamos a crearlas globales).

También son variables locales aquellas que se declaran dentro de funciones personalizadas por el usuario y que son llamadas desde el programa principal, olvidando Arduino estas variables cada vez que vuelve al programa principal.

El nombre de la variable no puede ser una palabra clave o reservada y los caracteres que podemos utilizar son las letras: a-z y A-Z (ojo! la ñ o Ñ no está permitida), los números: 0-9 y el guión bajo _ para simular espacios. Además hay que tener en cuenta que el primer caracter no puede ser un número.

Tipos de variables:

- int

- float

- char

● bool

● otras...

Ejemplo:

```
// C++ code
//
int lecturaHumedad;
int ledRojo = 5;
bool estadoBoton;
float distancia;
char charEntrante = ' ';
void setup()
{
  pinMode(ledRojo,OUTPUT);
}
void loop()
{
  digitalWrite(ledRojo,HIGH);
  delay(500);
  digitalWrite(ledRojo,LOW);
  delay(500);
}
```

25. *Actividad:* Analizar y debatir sobre el funcionamiento del ejemplo anterior e indicar el uso que le darían a las variables declaradas. Puesta en común de lo realizado.

26. *Actividad:* Hacer el ejercicio 26 (ida y vuelta) en código C++, asignando nombres en variables a cada LED.

Entradas Digitales:

Las salidas se establecen con pines de Arduino configurados como **INPUT**.

pinMode(nro_pin,INPUT);

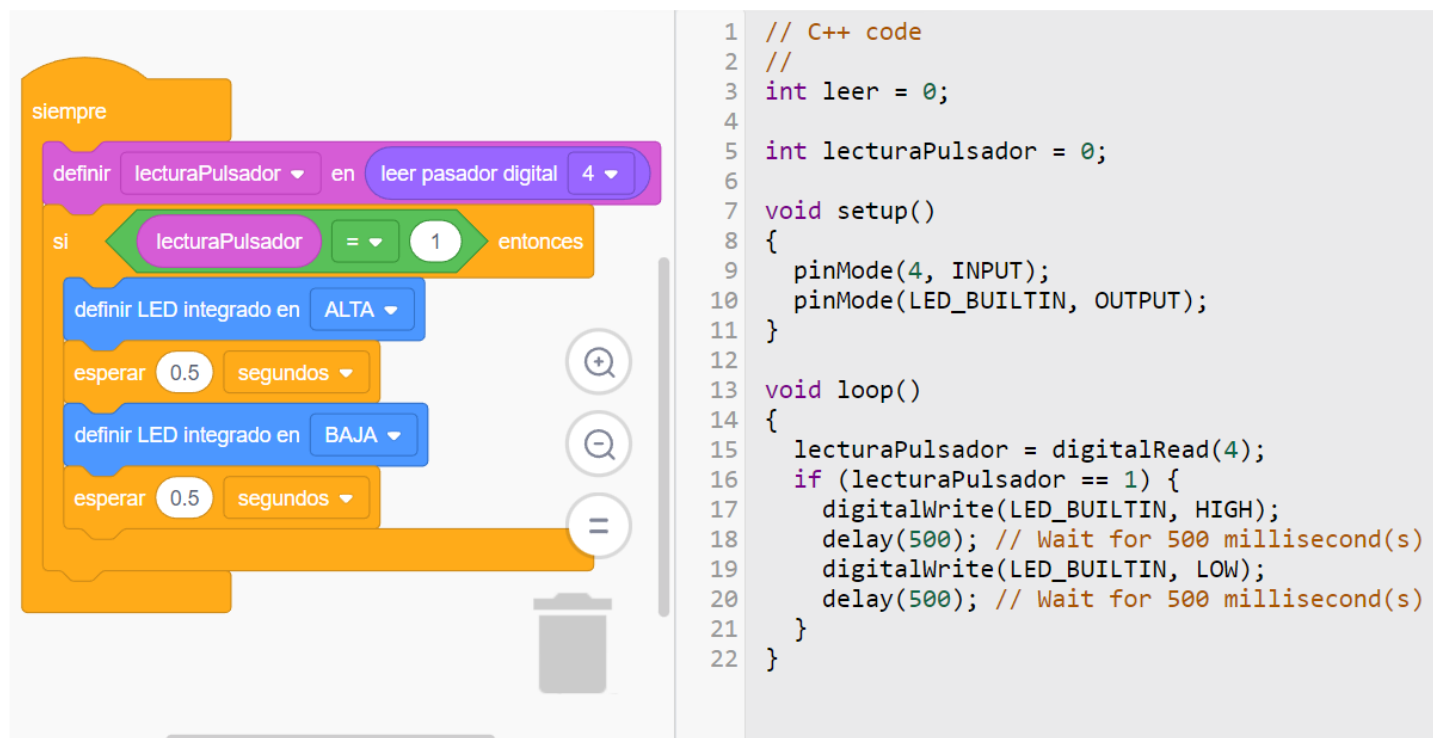
Una vez configurado el pin como entrada podremos hacer uso de ese pin para conectar botones o sensores digitales. Al ser **digital**, el pin será capaz de leer dos cosas: **HIGH** o **LOW**, dependiendo del voltaje que ingrese en él. Si ingresa un voltaje mayor a 2.1v en el pin de entrada, se lee un HIGH. En cambio si ingresa un voltaje menor a 2.1v el en pin de entrada, se leerá un LOW.

En el loop se coloca la siguiente función para leer un pin:

digitalRead(nro_pin);

Donde:

nro_pin: es el **número entero del pin** declarado como entrada en el setup que queremos leer.



The image shows the Arduino IDE interface. On the left, a block-based program is visible. It starts with a 'siempre' (always) loop block. Inside, there is a 'definir lecturaPulsador en leer pasador digital 4' block. This is followed by an 'if' block: 'si lecturaPulsador == 1 entonces'. Inside the 'if' block, there are two 'definir LED integrado en' blocks: one set to 'ALTA' and another to 'BAJA', each followed by an 'esperar 0.5 segundos' block. On the right, the corresponding C++ code is shown. It includes variable declarations for 'leer' and 'lecturaPulsador', a 'setup()' function that configures pin 4 as an input and the built-in LED as an output, and a 'loop()' function that reads the digital value of pin 4. If the value is 1, it sets the built-in LED to HIGH and delays for 500ms; otherwise, it sets it to LOW and delays for 500ms.

```
1 // C++ code
2 //
3 int leer = 0;
4
5 int lecturaPulsador = 0;
6
7 void setup()
8 {
9   pinMode(4, INPUT);
10  pinMode(LED_BUILTIN, OUTPUT);
11 }
12
13 void loop()
14 {
15   lecturaPulsador = digitalRead(4);
16   if (lecturaPulsador == 1) {
17     digitalWrite(LED_BUILTIN, HIGH);
18     delay(500); // Wait for 500 millisecond(s)
19     digitalWrite(LED_BUILTIN, LOW);
20     delay(500); // Wait for 500 millisecond(s)
21   }
22 }
```

A diferencia de digitalWrite(), la función digitalRead() sólo tiene un parámetro que es el pin a leer, lo que leamos debemos guardarlo en una variable entera como muestra el siguiente ejemplo:

int estadoPulsador = digitalRead(nro_pin);

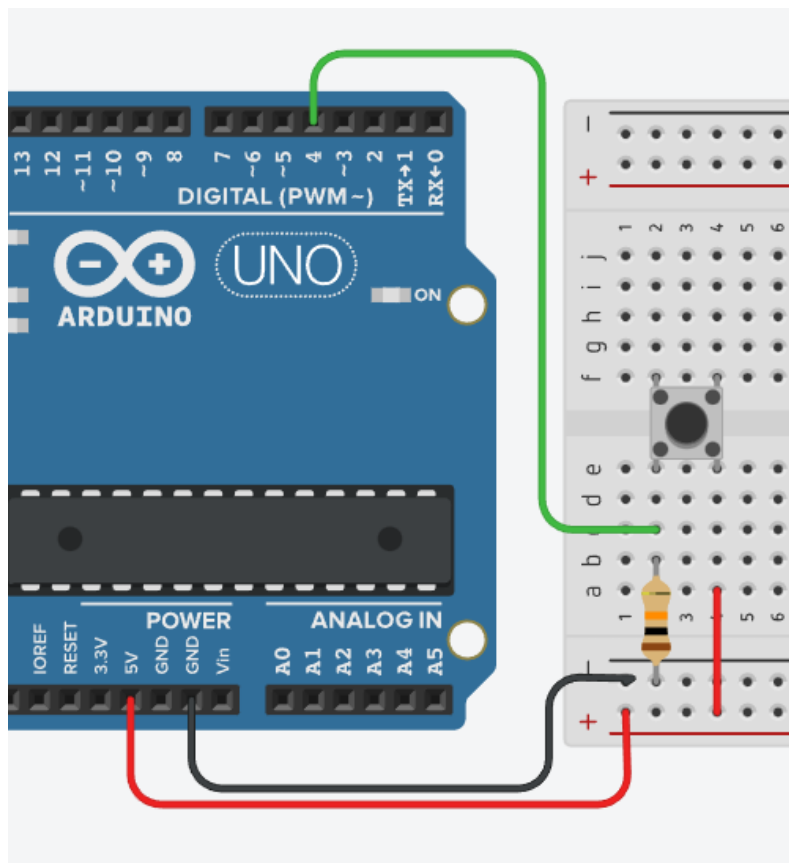
Se recomienda *siempre* guardar el valor de lectura del pin en una variable entera.

Los valores HIGH serán interpretados como un entero 1. Los valores LOW serán interpretados como un valor entero 0.

Recordar que HIGH y LOW representan dos estados: alto y bajo, verdadero y falso, encendido o apagado, caliente o frío, blanco o negro ... Cómo lo interpretes depende de tu código. En realidad, HIGH y LOW son dos constantes que están definidas en la sintaxis de Arduino.

Actividad :

[27. Analizar el siguiente ejemplo para sacar tus propias conclusiones, ¡puedes modificar y mover todo lo que quieras! \(imagen con link\)](#)



Lógica de Voltaje:

Como se vió en los Episodios 1 y 2, las estructuras de control como lo son el IF o WHILE evalúan la condición que le colocamos y si ésta es verdadera se “mete” a la estructura de control y realiza una acción. Para evaluar estas condiciones se utilizan operadores de comparación u operadores lógicos, aunque también un 1 es un verdadero (True) y un 0 es un falso (False).

Hasta ahora, en mBlock y Python hemos comparado variables y si esa comparación era verdadera o mientras esa comparación sea verdadera, se realizaba un acción específica.

Ahora la programación se basa en la electrónica y tiene sus propios True y False, los cuales son niveles de voltaje (altos o bajos). En donde +5V(HIGH) es un VERDADERO y el 0V o GND (LOW) es un FALSO. Al igual que lo son el 1 y el 0.

Los pines digitales de Arduino, son 14, y están numerados del 0 al 13, éstos pueden ser usados como entradas (INPUT) o salidas (OUTPUT), según cómo se los configure en el setup() del programa.

Pin digital de Arduino como OUTPUT o salida:

Para los pines digitales de Arduino configurados como salida, podemos establecer su voltaje de salida digital con dos estados posibles, HIGH y LOW, en donde HIGH es +5V y LOW es GND o 0V.

Pin digital de Arduino como INPUT o entrada:

En Arduino los valores de alimentación habituales son 0V y 5V. En este caso el cambio entre lo que se reconoce como Verdadero o Falso está en 2,5V. Por tanto si medimos una tensión con un valor intermedio **entre 0 a 2,5V Arduino devolverá una lectura LOW**, y si medimos un **valor entre 2,5V y 5V, devolverá HIGH**.

La lectura dará un valor "HIGH" si el valor de tensión medido es superior a una tensión umbral, y "LOW" si el valor de tensión es inferior.

Nunca introducir una tensión fuera del rango 0V a 5V en una entrada digital o analógica o podemos dañar el pin correspondiente y dejarlo permanentemente inutilizado.

Todos los sensores son entradas a Arduino, por ejemplo: botones, sensor PIR, fines de carrera, sensor de humo, sensor Infrarrojo, etc.

Cada sensor tiene su manera de conectarse, hay cuales vienen con los pines correspondientes para conectarlos directamente, pero hay otros que debemos aprender a conectar. El sensor más básico de todos es el **botón o pulsador**, hay que conectarlo de tal manera que **cuando no lo estemos presionando mande una señal LOW o falsa** para indicar ese estado, y **al presionarlo pasa a un estado de HIGH o verdadero**, para eso *hay que conectarlo de manera tal que asegure esos dos niveles lógicos, mediante una resistencia Pull-Down*.

La forma de conexión es la siguiente:

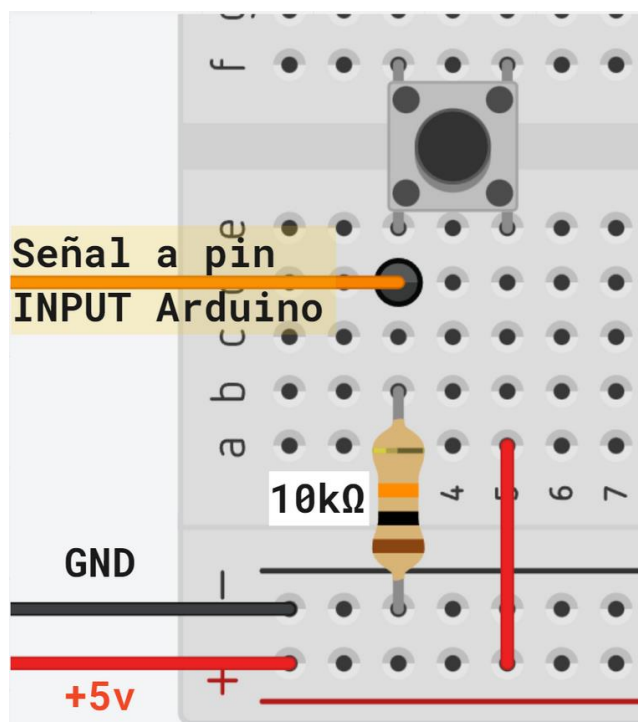
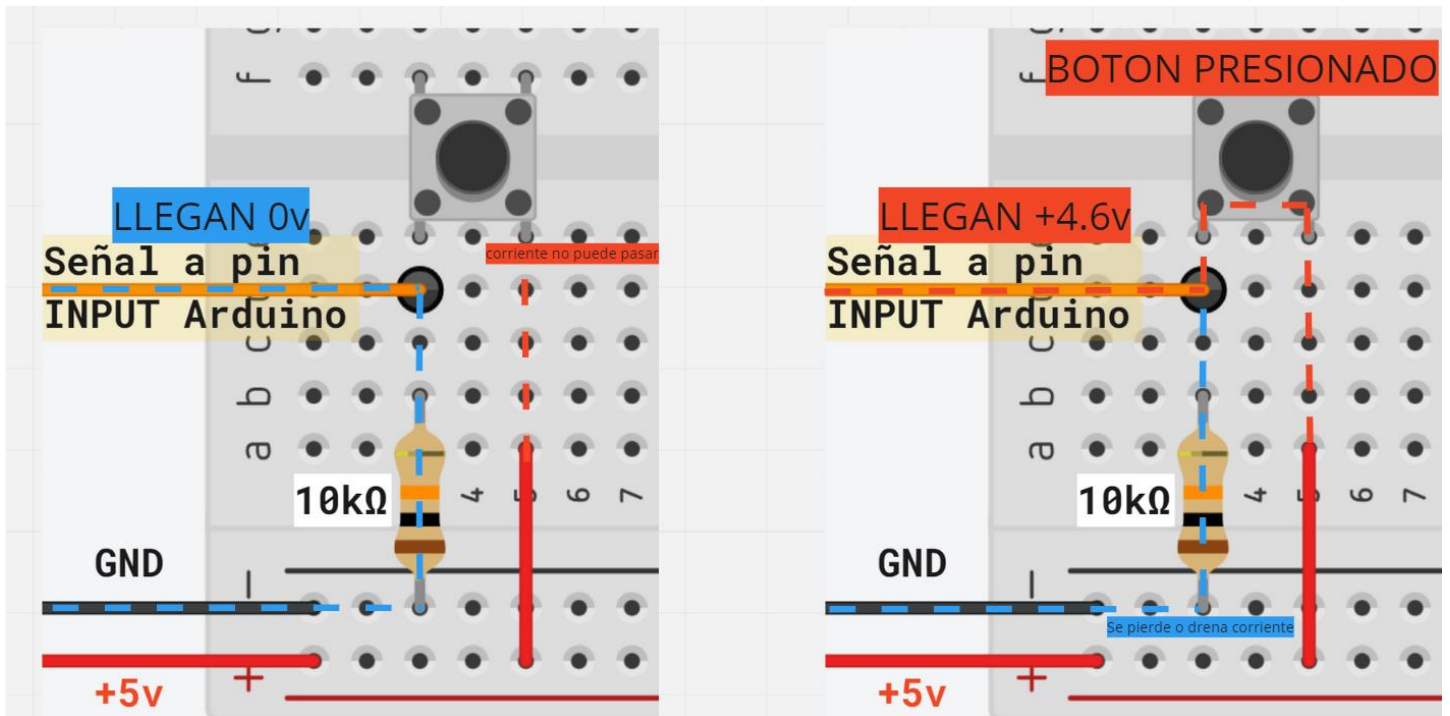
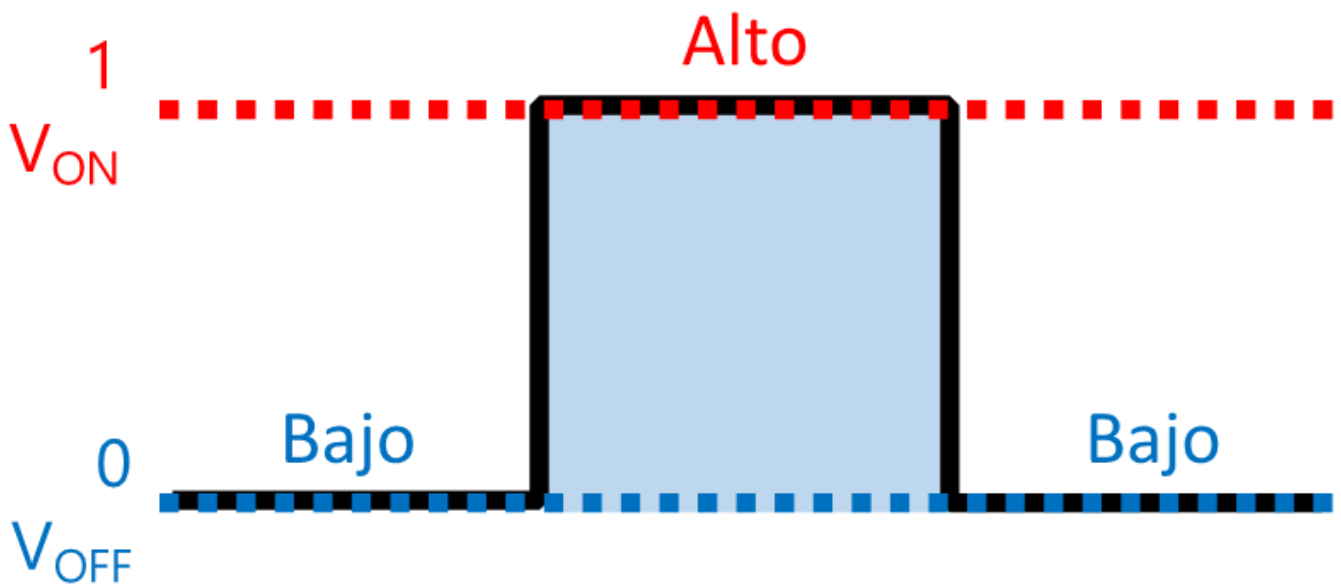


Diagrama de conexión de un botón.

Pulsador con resistencia Pull-Down: En estado de reposo se encuentra en LOW, (0) o bajo; cuando se pulsa cambia el estado a HIGH, (1), 5v o alto.



Flujo de corriente con botón sin presionar y con botón presionado respectivamente.



El voltaje de entrada en el pin de arduino se encuentra bajo mientras el botón esté sin presionar, se encontrará en alto mientras se presione el botón.

El valor de resistencia tiene que ser alto para asegurar un nivel débil de GND, pero lo suficientemente fuerte como para establecer una señal de LOW. Se recomienda el uso de resistencias de 10kΩ o más.

RESISTENCIA PULL-UP y PULL-DOWN

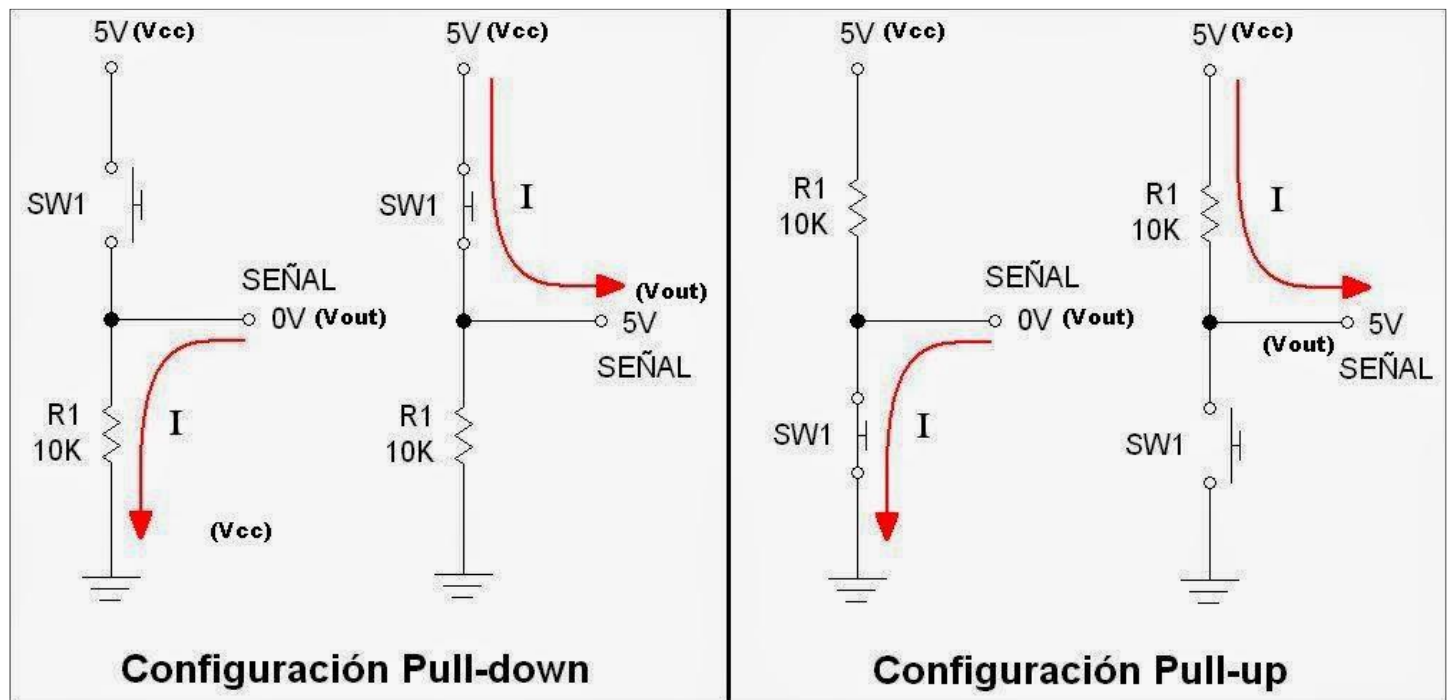
Las resistencias Pull-Up y Pull-Down cumplen un papel muy importante dentro de la electrónica, son las encargadas de establecer el **HIGH** y el **LOW**. Se las utiliza para aplicar lógica booleana en circuitos electrónicos y en Arduino.

Pull-Down significa llevar hacia abajo, es decir, LOW. Una resistencia Pull-Down se encarga de dar un 0, LOW, bajo, LOW, GND o 0v en todo momento.

Pull-Up significa llevar hacia arriba, es decir, HIGH. Una resistencia Pull-Up se encarga de dar un 1 lógico, HIGH, un alto o 5v.

En un botón con resistencia Pull-UP, en el momento que se activa el pulsador, el estado cambia a 0, LOW, GND, bajo o 0v, en reposo se encuentra en HIGH.

En los siguientes esquemas se simbolizan las resistencias Pull Up y Pull Down. El terminal "SEÑAL" es la entrada a otro sistema, en este caso el microcontrolador de Arduino.



28. *Actividad:* Realizar la conexión del botón con resistencia Pull-Down como se indica en las imágenes anteriores, conectar un voltímetro entre "señal" y GND. Verificar que se cumplan los estados HIGH y LOW. Luego leer con Arduino.

29. *Actividad.* Conectar el botón con resistencia Pull-Up: cuando no lo estemos tocando da un HIGH y cuando lo tocamos da un LOW. Leer con voltímetro o programa de Arduino.

30. *Actividad:* VER LOS SIGUIENTES CÓDIGOS Y COMENTAR CONCLUSIONES:

[Ejemplo marcha parada con dos pulsadores](#)

[Ejemplo de enclavamiento con 1 pulsador.](#)

31. *Actividad:* Replicar los dos ejemplos anteriores. Puedes intentar realizarlos con bloques... ¡O en código!

32. *Actividad:* Con el fin de monitorear las especies en peligro de extinción suelen dejarse cámaras instaladas junto con cebos para que distintos animales se acerquen y así poder

avistarlos, por ejemplo, el gato andino avistado muy recientemente en Villavicencio, el cuál hacía décadas no se veía.

Para captarlo hay que esperar mucho tiempo y grabar tantos datos es un problema, ya que hay que tener grandes cantidades de memoria y energía para la cámara. Florencia es investigadora y fotógrafa, quiere encender su cámara cada vez que haya movimiento para ahorrar memoria y batería, para eso compró un sensor PIR (Infrarrojo Pasivo), configurar el Arduino para que cada vez que haya movimiento se encienda (simulando que es la cámara) un LED. Basarse en este programa.

33. *Actividad prioritaria:* Desarrollar un programa (en bloque o código) que imprima el valor de una variable llamada “contador”, en la que con un botón B1 se aumenta en 1 su valor y con un botón B2 se disminuye en 1 su valor (dos botones). Si te animás, podés Usar comunicación Serial.

34. *Actividad complementaria:* Desarrollar un programa que, continuando el programa anterior:

- a. Si contador vale 1: Se enciende un led verde (solo este led)
- b. Si contador vale 2: se enciende un led amarillo (solo este led)
- c. Si contador vale 3: se enciende un led rojo (solo este led)
- d. Si contador vale 4: contador se reinicia. (contador = 0)

Puesta en común de toda la ejercitación. destacar aspectos que tuvieron mayor dificultad y atenderlos para poder seguir avanzando.

❖ - Operaciones matemáticas, lógicas y de comparación (recordando Python)

Las operaciones matemáticas son las mismas que las utilizadas en mBlock y Python, se pueden clasificar en 3:

● Operadores de Comparación

● Operadores Lógicos

● Operadores Matemáticos

Operador de Comparación	Significado del Símbolo
Elemento1 < Elemento2	“es menor a....”
Elemento1 <= Elemento2	“es menor o igual a....”
Elemento1 > Elemento2	“es mayor a”
Elemento1 >= Elemento2	“es mayor igual a....”
Elemento1 == Elemento2	“es igual a....”
Elemento1 != Elemento2	“es distinto a...”

Operadores Lógicos Significado del símbolo

Elemento1 **&&** Elemento2

Es un **and**. La conjunción o **&&**, tendrá una salida ALTA (**1** o Elemento1 **and** Elemento2 **true**), únicamente cuando los valores de ambas (o todas) entradas sean ALTOS. Si alguna de estas entradas no son ALTAS, entonces tendrá un valor de salida BAJA (**0** o **F**).

Elemento1 **||** Elemento2

Elemento1 **or** Elemento2

Cuando todas sus entradas están en 0 (cero) o **F** su salida está en **0** o en **Falso**, mientras que cuando **al menos una o ambas** entradas están en 1 o en ALTA (**true** o **verdadero**), su SALIDA va a estar en 1 o en ALTA (**V**).

! Elemento1

not Elemento1

Significa la negación de la proposición que ponemos a su derecha. *Si negamos algo Verdadero (HIGH), se hace Falso (LOW). Si negamos algo Falso, se hace Verdadero.*

Operador Matemático	Significado del Símbolo
+	suma
-	resta
*	multiplicación
/	división
%	resto
**	potencia
/	división con resultado entero

Con la función de Python “input()” fuimos capaces de ingresar datos desde el teclado hacia el programa de Python, el teclado era nuestra entrada principal, aunque también lo son los datos de los archivos a los que accedemos. **Ahora, con Arduino, las entradas no son datos introducidos por personas sino que sensores o dispositivos que envían señales eléctricas.** Se hará uso de los operadores para comparar las entradas con algún límite guardado en el programa, para acondicionar o modificar un valor de una señal y para establecer condiciones lógicas para establecer el flujo del programa.

[Visualiza el EJEMPLO sobre uso de operadores matemáticos](#)

Se desarrollará un ejemplo que combine estructuras de control, variables, contadores, comparadores lógicos y salidas en Arduino.

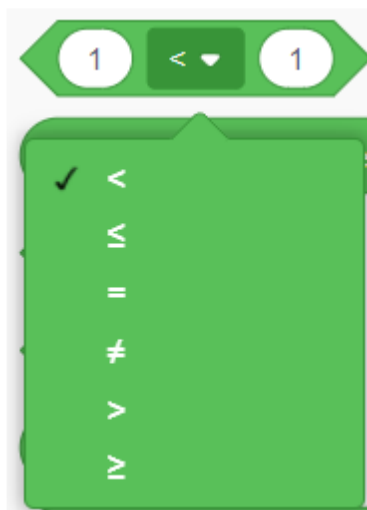
Se quiere lograr tener una variable llamada “test” que incremente su valor en 1 unidad por segundo en cada ciclo de ejecución del loop(). Cuando esta variable tenga el valor 4, se activará un altavoz conectado al Arduino en el pin digital 3.

Primero, vamos a familiarizarnos con las estructuras **matemáticas** de tinkercad.

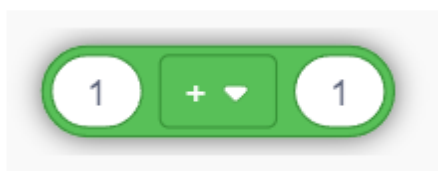


Comparador lógico: compara los dos ítems que se le coloquen (formas circulares)

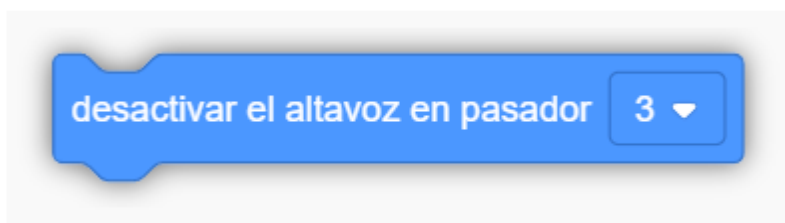
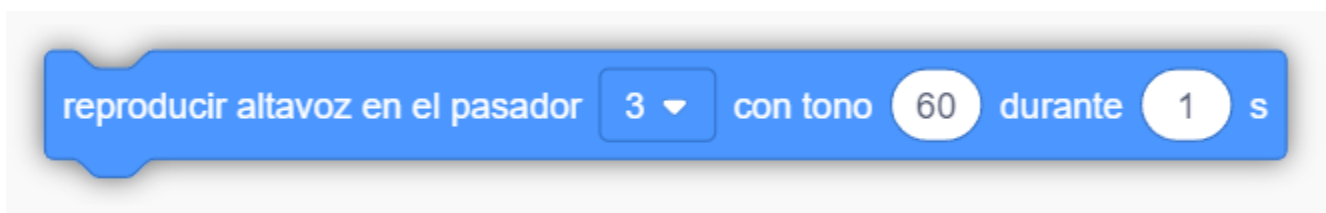
Permite seleccionar entre varios comparadores.

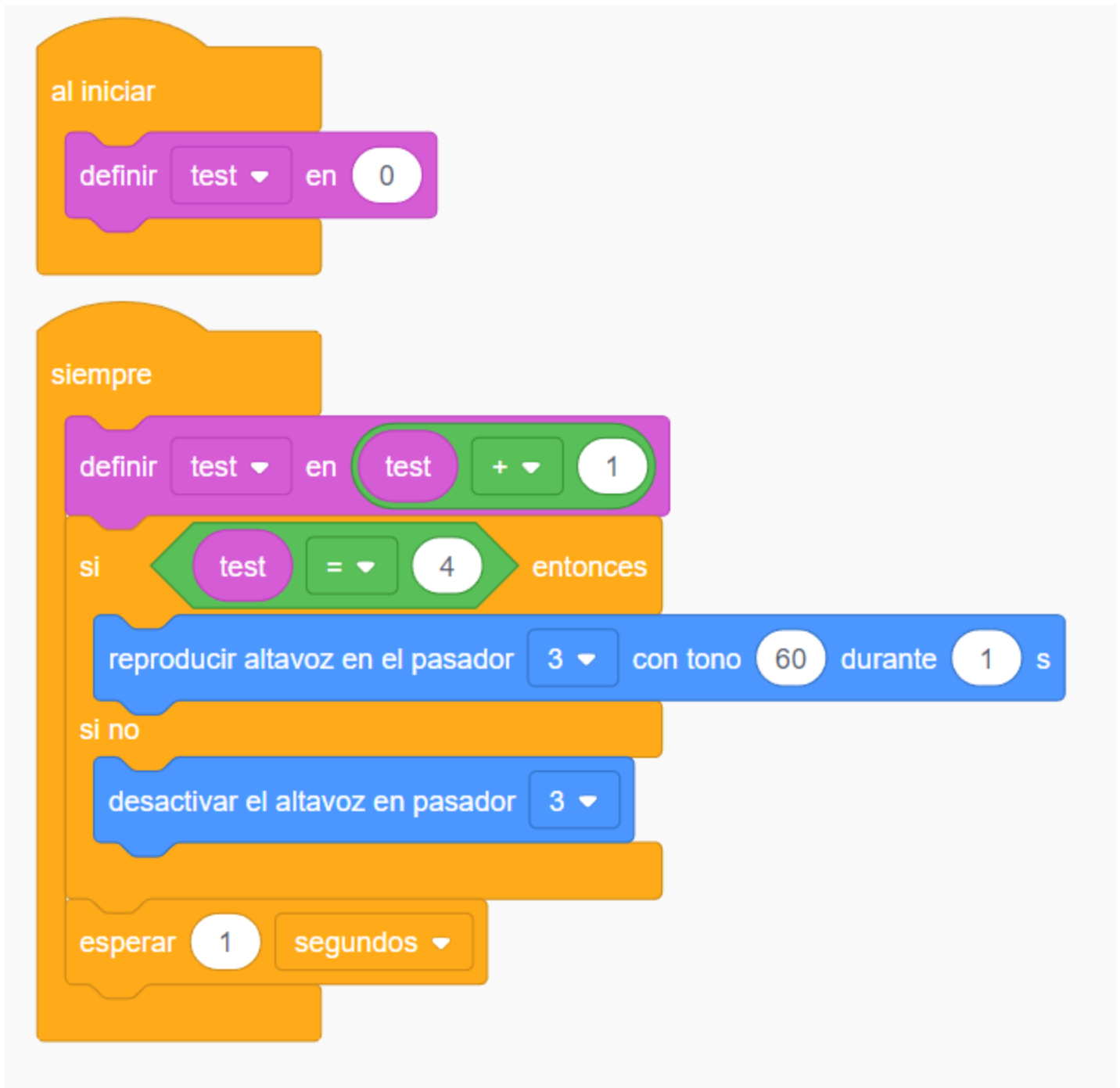


Operador aritmético: permite realizar operaciones matemáticas entre los dos ítems que se le coloquen.



Control del altavoz: El primer bloque reproduce un sonido en el altavoz que se le conecte al pin (pasador) especificado. Permite seleccionar un tono y una duración del mismo. El tono hace referencia a la frecuencia del sonido reproducido por el altavoz. Si el número del tono es mayor, el sonido será más agudo y si es menor, es más grave. El segundo bloque es para dejar de reproducir sonidos en el altavoz.





Nota: el ejemplo está resuelto en el siguiente link:
https://www.tinkercad.com/things/jRXYxNfjirL-ejemplo-matematica-variables?utm_source=matematica%20y%20variables&utm_campaign=Episodio%203

35. *Actividad:* Desarrollar un programa con dos botones y usar el operador lógico “and” para que si tocamos los dos botones se encienda un led, sino el led permanece apagado.

36. *Actividad:* Modificar el programa anterior haciendo uso de la compuerta “or” ¿Ahora que sucede? ¿Cómo se diferencia con la “and”?

37. *Actividad:* Se desea contar la cantidad de gente que entra a un local, para eso se coloca un sensor en la puerta de entrada y otro en la puerta de salida. El local permite que haya 7 personas dentro como máximo. Si este límite se cumple, se enciende una señal de luz de color rojo. Si el

límite no es superado la señal es verde. Simular los sensores con dos pulsadores, uno cuenta la gente que entra y el otro resta cuando sale alguien. Hacer 4 soluciones diferentes, usando los operadores: $>$, $>=$, $<$ y $<=$.