
Taller de electrónica digital,
programación y Arduino

# 1. Presentación del taller

El taller “Introducción al Electrónica digital”, está enmarcado en el Plan Nacional Integral de Educación Digital (PLANIED). Propone un recorrido para reconocer los entornos de programación como recursos educativos que favorecen la integración de las competencias PLANIED.

Este taller está destinado a docentes del país que buscan aproximarse a los conceptos básicos de la electrónica, programación y microcontroladores digitales. Tiene como objetivo presentar a los docentes una herramienta y sus posibilidad de crear distintos ejercicios de electrónica y programación. Cada ejercicio está acompañado de un conexionado electrónico y un programa funcional.

# 2. Metodología

El trabajo que se lleva a cabo en este taller busca integrar instancias de reflexión sobre los conceptos básicos de la electrónica y la programación. Así como también la ejercitación a través de una serie de prácticas con ejercicios básicos. Cada ejercicio y ejemplo está acompañado de un diagrama del conexionado y su programa correspondiente

El taller se divide en 5 módulos:

**1. Electricidad:** Se enfoca en los siguiente puntos:

*- ¿Qué es la electricidad?*

*- Corriente, tensión  y resistencia.*

*- Cómo usar un protoboard.*

**2. Primer programa:** Se realiza el primer programa en Arduino, la primera conexión y variaciones.

**3. Entradas digitales:** Se conecta y analiza botones en los desarrollos.

Sumando el concepto de variables y números azarosos.

**4. Sentencias de control:** Se realizan programas con toma de decisiones. Y se analizan los primeros árboles lógicos.

**5. Introducción al servomotor:** Programación de control del motor.

## 3. Objetivos Generales

Introducción a la electrónica digital.
Introducción a la plataforma Arduino
Iniciación en la programación digital.
Salidas y entrados.
Introducción y uso del Servomotor.

**Modalidad:** Presencial.

**Duración:** 4 horas.

**Destinatarios:** Docentes de escuelas primaria.

**Equipamiento:** Kit Arduino, Notebook o Netbook.

**Materiales Didácticos**

Para el formador:

- *Guía de Formadores (en formato pdf).*
- *Presentación.*
- *Programas y gráficos de conexiones.*
- *IDE Arduino.*

## 4. Competencias de Educación Digital

Se orientan a lograr el desarrollo integral de los alumnos como personas y ciudadanos del siglo XXI, capaces de construir una mirada responsable y solidaria para transitar con confianza por distintos ámbitos sociales. (Ripani, M. F. (2016). Competencias de Educación Digital. Ministerio de Educación y Deportes de la Nación. Buenos Aires, Argentina).

Un proyecto de programación con Arduino integra las seis dimensiones de las compentencias de educación digital.

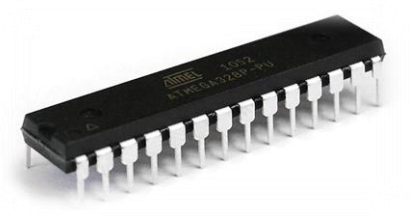

## ¿Lo eléctrico y lo electrónico?

Existen una variedad de objetos que enchufamos en nuestras casas. Algunos son eléctricos y otros electrónicos. Los eléctricos son aquellos que funcionan con electricidad o la conducen. Por lo general transforman la electricidad en otras energías. La estufa transforma la energía eléctrica en calórica, un motor en energía cinética, un ventilador en energía eólica.

Los objetos electrónicas utilizan la electricidad como medio de control, orientados al tratamiento de información. Pueden controlar a los aparatos eléctricos. Todos los aparatos electrónicos son eléctricos, pero no todo los aparatos eléctricos son electrónicos.

## ¿Qué es un microcontrolador?

Es un integrado electrónico capaz de ejecutar un programa grabado en su interior. Este programa que se aloja en una de sus memoria puede borrarse y volverse a grabar miles de veces. Pero solo puede alojar un solo programa por vez.



Consta de varia partes:

- **Unidad central de procesamiento o cpu**: posee como objetivo interpretar a los programas. Está integrado por dos unidades interna: Alu, unidad aritmética lógica y la Cu o unidad de control.

- **Memoria:** es una unidad de almacenamiento de datos informáticos de estado sólido. Existen varios tipos de memorias:

- **Flash:** donde se almacena los programas. La información se borra cuando se almacena otro programa.

- **SRam:** donde se almacenan valores durante la ejecución del programa. Esta información se borra cuando se apaga el microcontrolado.

- **EEprom:** se guarda información a largo plazo. Solo se borra cuando otro programa escribe sobre ella.

- **Puertos de entradas y salidas I/O:** Son grupos de pines o patas que poseen los microcontroladores que les permite comunicarse con el exterior. Estas patas pueden definirse como entradas (para recibir datos) o salidas (para generar datos).

# ¿Qué es Arduino?

Arduino es una plataforma de prototipos electrónicos de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Está diseñado para artistas, diseñadores y hobby.

El potencial de Arduino radica en la facilidad con la que se puede acceder a la programación del microcontrolador que posee, un lenguaje de programación fácil de aprender, multiplataforma (Windows, Osx, Linux), la gran cantidad de documentación compartida por desarrolladores y la buena relación costo beneficios.

## ¿Cómo se programa un Arduino?

Para programar un Ardunio se requiere una computadora. En ella se debe instalar el Entorno de desarrollo Arduino (IDE). La misma se descarga desde su web: https://www.arduino.cc/en/Main/Software seleccionando la versión que corresponda a las computadora de se utilice.

Este entorno permite escribir el programa que se desea cargar al Arduino en el lenguaje propio de Arduino. Antes de realizar la carga el IDE compila el programa: revisa que el programa este formalmente bien escrito y lo transformara en un código hexadecimal interpretable por el Arduino. Para realizar la carga se debe conectar el Arduino a la computadora mediante un cable USB.

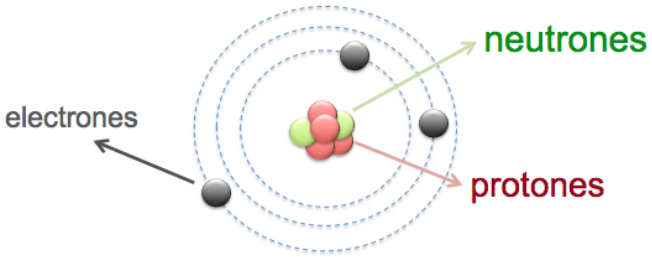
## ¿Qué conocimientos debo tener para utilizar Arduino?

Se debe tener conocimientos básicos de electrónica y programación. Esta guía te ayudará a obtener estos conocimientos. Lo importante es hacer experiencia paso a paso para llegar a los objetivos planteados. Tener en cuenta que una conexión errónea de los componentes electrónicos al Arduino puede causar la ruptura total o parcial del mismo.

# 01 // Conceptos básicos sobre electricidad

## ¿Qué es la electricidad?

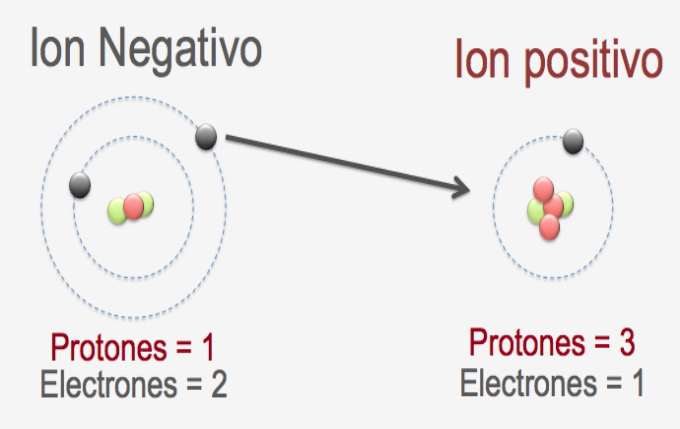
La electricidad es un conjunto de fenómenos producidos por los movimientos y relaciones de las cargas eléctricas positivas y negativas en la materia.




Los átomos (que es la porción más pequeña de un elemento que aún retiene sus propiedades) constan de un núcleo con protones y electrones. Los protones poseen carga positiva y los electrones cargas negativas. Los protones giran alrededor de los protones porque se atraen por una fuerza de magnetismo.

Cuando un átomo posee la misma cantidad de electrones que protones se dice que tiene una carga neutral, se encuentra en equilibrio. Si la cantidad de electrones es superior a la cantidad de protones se dice que el átomo posee una carga positiva (ion positivo). Esto significa que los protones pueden atraer a más electrones de los que posee en su órbita.

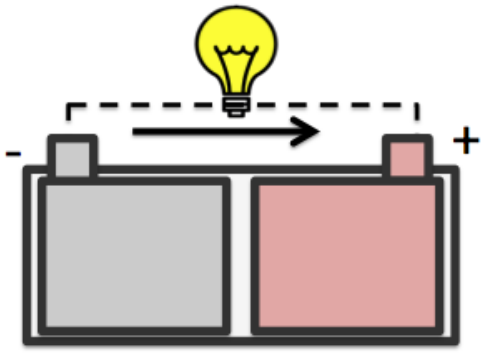
Cuando el átomo posee una cantidad mayor de electrones que protones la carga es negativa (ion negativo). Los electrones sobrantes se los denomina electrones libres, que por atracción magnética viajaran hacia otros átomos que necesiten electrones para alcanzar su equilibrio.



Puede explicase con una metáfora, a la que recurriremos varias veces. Si tenemos dos recipientes con agua conectados entre ellos. Si uno de ellos posee más agua que el otro, ambos buscarán el equilibrio trasladando agua del contenido que posee más agua al que posee menos.

## ¿Qué es una batería o fuente de energía?

Una pila o batería posee dos recipientes de materia aislados. La parte positiva (con mucha cantidad de protones) y la parte negativa (con electrones libre). Si se realiza un camino o un circuito entre el polo negativo y el polo positivo los electrones viajarán por el mismo atraídos por los átomos con carga positiva. Este flujo de electrones se lo denomina corriente.




## ¿Qué es la corriente?

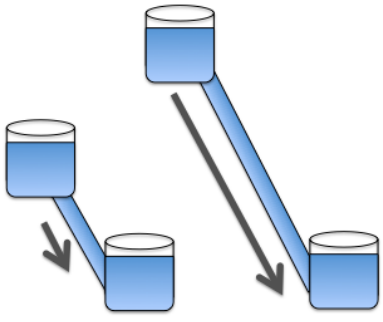
La cantidad de electrones que circulan por un punto del circuito en un segundo. En el lenguaje electrónico se la denomina con la letra ”i”.

| | |
|---|---|
| **Corriente = i** | Su magnitud se mide en Amperios o mA (milis amperios). |

En nuestro ejemplo de los recipientes con agua la corriente sería la cantidad de gotas de agua que circulan por el conexionado en un segundo.

## ¿Qué es la tensión o voltaje?

Le tensión es la diferencia de potencial eléctrico o la tensión eléctrica entre dos puntos. Se le llama



así porque tiene el potencial para hacer un trabajo que podemos medir en volts. Volvemos al ejemplo de los recipientes de agua: supongamos que tenemos dos recipientes nuevamente, uno más elevado que el otro. El agua correrá del más alto al más bajo. Como hemos visto la cantidad de agua es la corriente y la presión con la que circula es la tensión o voltaje: A mayor diferencia de altura, mayor potencia: mayor voltaje. Se lo denomina con la letra "v":

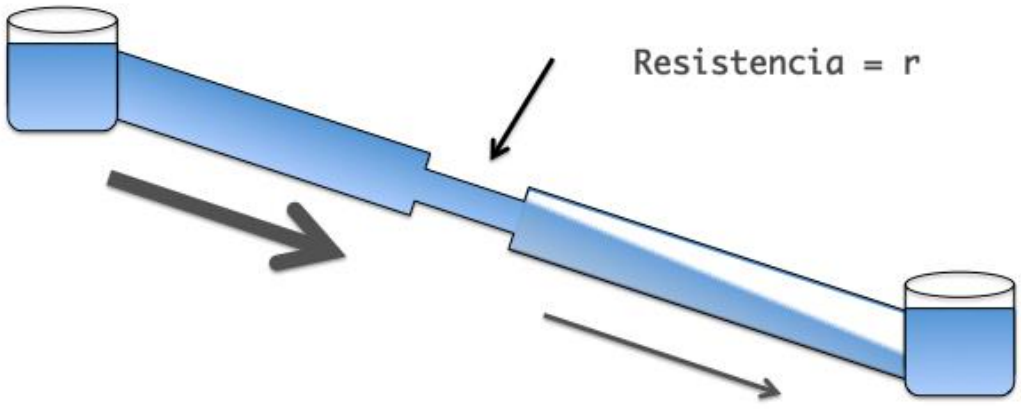
| | |
|---|---|
| **Tensión: v** | Su magnitud se mide en Volts. |

## ¿Qué es una resistencia o resistor?

La resistencia es una fuerza que se opone al paso de los electrones por un circuito. Se utiliza para delimitar y controlar la tensión y la corriente dentro de un circuito.

Si volvemos al ejemplo de los recipientes de agua, la resistencia es como una reducción del conducto que conecta los dos recipientes. El valor resistivo se mide en ohms y se lo denomina con la letra "r"

**Resistencia = r**.



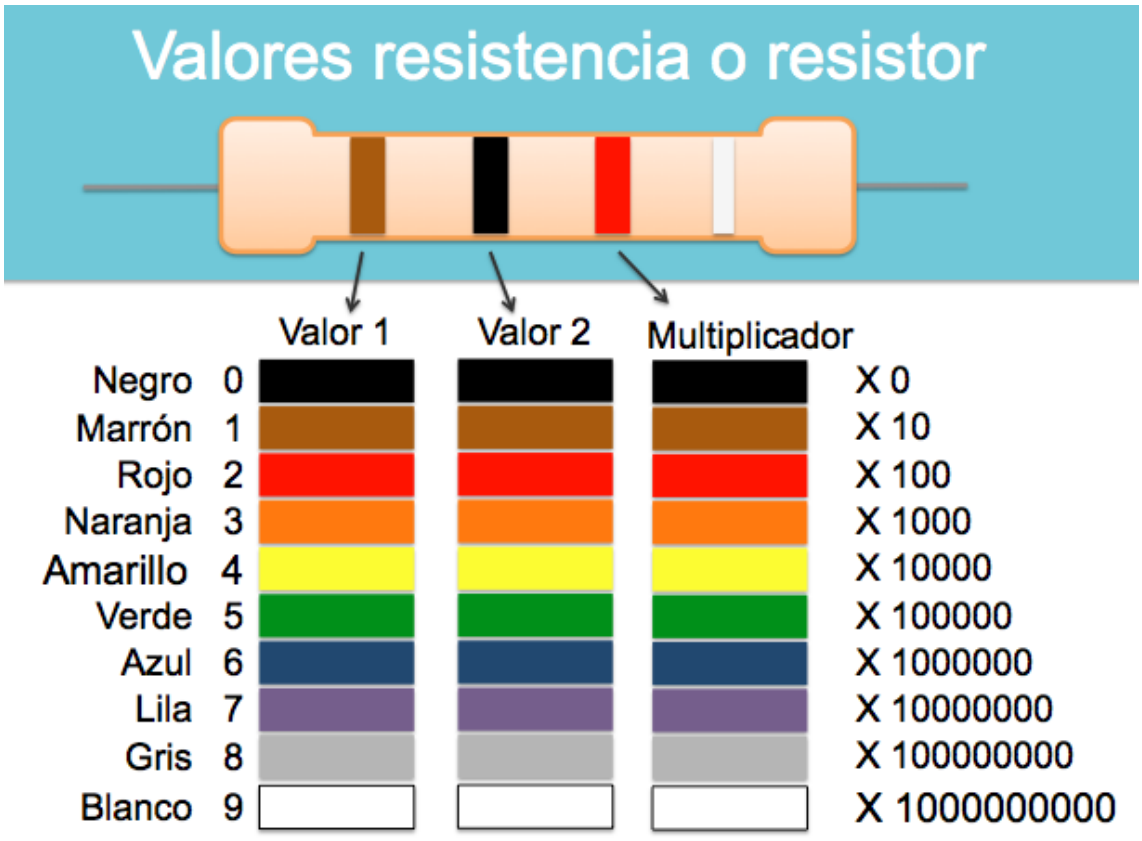
## Materiales resistivos y conductivos.

Un material con baja resistencia se dice que es conductivo: metales, agua, grafitos, soluciones salinas entre otros. Y los materiales que poseen alta resistencia se los denomina resistivos: plásticos, gomas, cerámicos entre otros. Utilizando resistencias podemos limitar y controlar la tensión y la corriente dentro de un circuito eléctrico.

## Valores de una resistencia

Los valores resistivos en ohms están indicados con banda de colores sobre la resistencia. Cada color corresponde a un número según el cuadro de colores. Los dos primeros corresponden a un número literalmente y el tercero a una cantidad de ceros. Existe una cuarta línea que indica un porcentaje de precisión del valor, pero no le daremos importancia. Era útil en tiempo pasado cuando la producción de las resistencias no era tan precisa como hoy en día.

## ¿Cómo se calculan las resistencias?



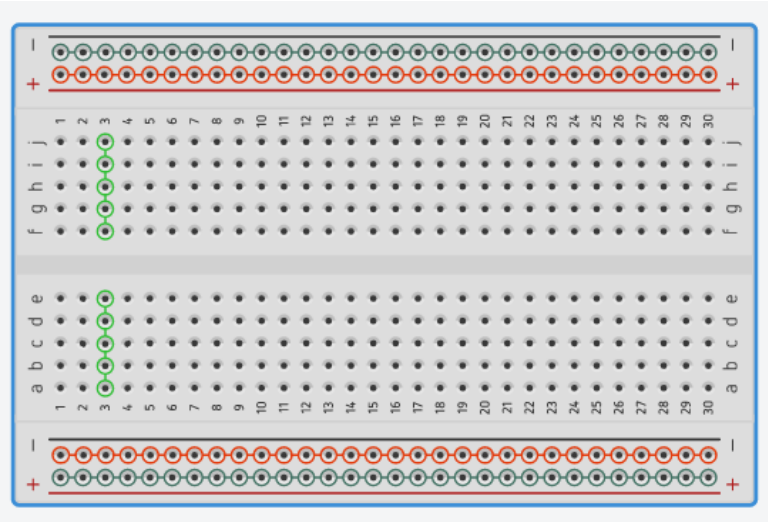
El objetivo en conectar un led al Arduino sin que este se rompa. Los leds (en general) son para 3 volts con un consumo de 20 mA. Entonces nuestro circuito no debe dejar llegar al led  a valores de voltaje superiores porque el led se quemará.

*La relación entre la tensión, corriente y resistencia, se encuentra en el ANEXO al final de este documento*

# 02 // Placa Arduino

## Protoboard

El protoboard es una herramienta para prototipar circuitos eléctricos. Se utiliza para probar los desarrollos antes de pasarlos a un circuito soldado. Su gran utilidad radica en que se puede armar y desarmar rápidamente.

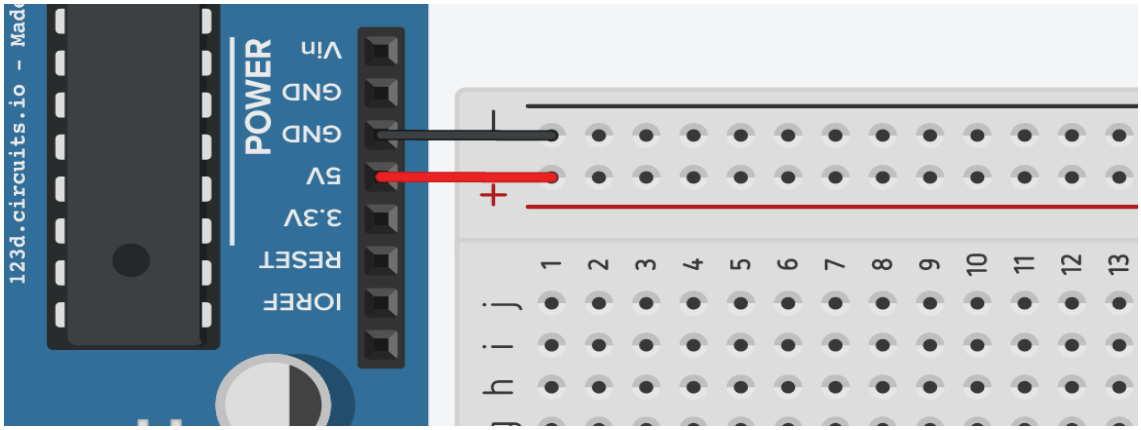


El protoboar contiene pistas de conexión que se deben conocer y no confundir. Las regiones exteriores poseen dos pistas de conexionado horizontales. Donde cada perforación esta conectada con las perforaciones horizontales vecinas. Las regiones internas (son dos divididas por un separación central) cada perforación esta conexionada con las perforaciones vecinas verticales. Por lo general son grupos de conexiones de 5 perforaciones.

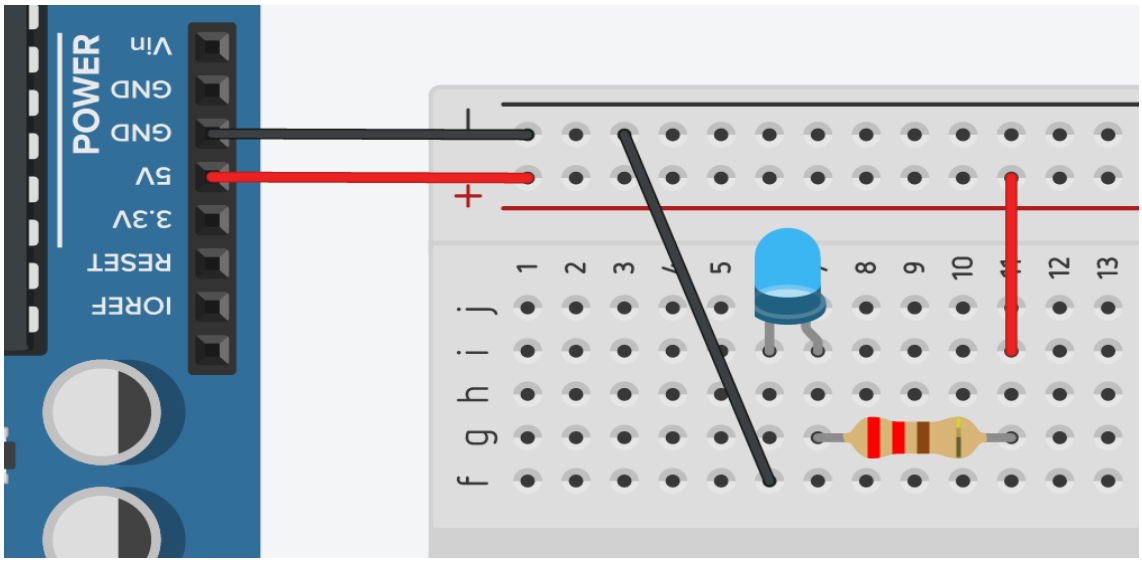
Para unir las perforaciones que no se encuentran conexionadas internamente se utilizan cables con pines en los extremos.

## Prender un led: Arduino como fuente.

El objetivo es encender un led utilizando al Arduino como generador de tensión. Conectar un cable del pin V5 (5 volt positivo) a una de las pistas exteriores del protoboard. Y conectar cualquiera de los pines que dicen “GND” del Arduino a otra línea exterior del protoboard. “GND” es la abreviación de la palabra “ground” (que en ingles significa “tierra”). Los tres pines “gnd” del Arduino son iguales: todos son negativos. No conectar en las mismas pistas los positivos con el negativo, eso provocará un cortocircuito y se corre riesgo de ruptura.




Con esas dos conexiones tenemos tensión en el protoboard. Ahora coloquemos el led en el protoboard y la resistencia necesaria para que no se queme el led. Tener en cuenta que el led posee polaridad, esto significa que hay una pata que va a positivo y otra a negativo. La pata más larga del led va a positivo y la corta a negativo. Copiar el conexionado del siguiente grafico.

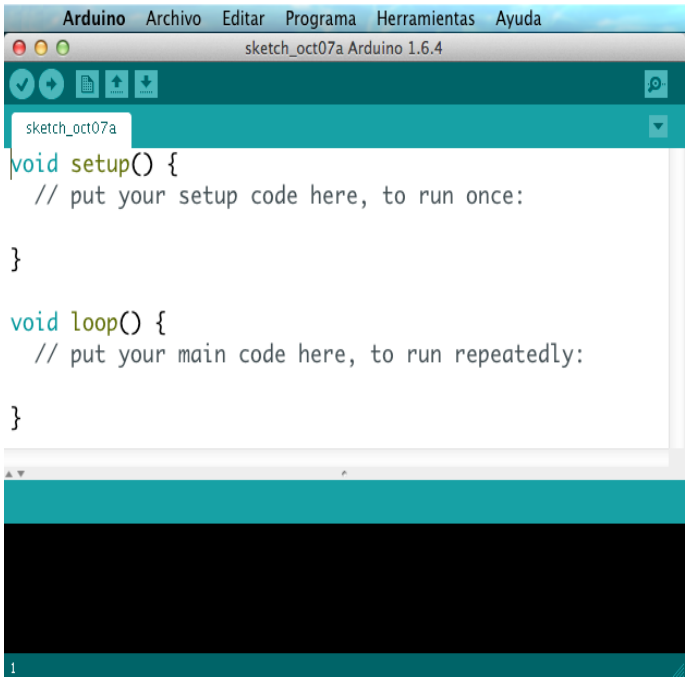


## Arduino – IDE Arduino

El IDE es una plataforma que permite escribir el código del programa, compilarlo y subirlo al Arduino. El IDE se descarga de: https://www.arduino.cc/en/Main/Software. Seleccionar la versión que corresponda a su computadora. Una vez instalada, abrir el programa.

El IDE posee varias secciones:
Un menú horizontal, botoneras de acción, el editor y la consola de mensajes:

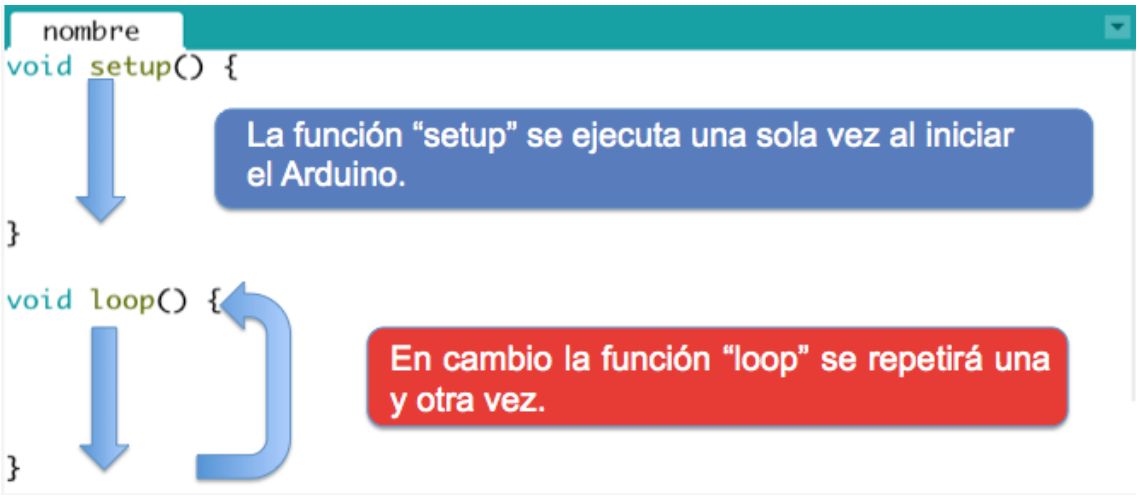
- El menú posee todas las opciones que el IDE ofrece.
- Con la botonera de acciones podemos guardar, leer, crear nuevos programas. También podemos cargar el programa realizado al Arduino.
- En el espacio de Edición es donde vamos a escribir muestro programa.
- Y por último en la consola de mensajes es donde el IDE nos mostrará si existen errores en el código o errores de comunicación con la tarjeta Arduino




## Conceptos Generales del Lenguaje Arduino

El lenguaje Arduino es un lenguaje muy sencillo de utilizar en relación a otros lenguajes. Lo primero que hay que saber es que los lenguajes poseen una estructura formal que debe respetarse. Si no es así el IDE de Arduino no comprenderá el programa que estamos escribiendo.

El lenguaje posee dos funciones fundamentales: la función **“setup”** y la función **“loop”**. La diferencia esencial entre ellas es que la función **“setup”** se ejecuta una sola vez al prenderse el Arduino. En cambio la función **“loop”** se repetirá una y otra vez.




En la función **“setup”** se definen todos las configuraciones de inicio de nuestro programa Arduino. Por ejemplo: si un pin será entrada o salida, si deseo que tal pin arranque encendido, o si deseo que en la memoria se guarde tal valor. En cambio en la función **“loop”** se colocan todas las acciones que quiero que realice Arduino.

## Comentarios:

El lenguaje Arduino permite escribir el comentario que no pertenece al lenguaje de programación. Es de mucha utilidad para poder escribir comentarios sobre lo que hace el programa. Los comentarios son resaltados en color gris.

```
void setup() {
  // comentario corto

  /*
    Cometarios
    Largos
  */

}
```

- Todo lo escrito en una línea luego de los símbolos **“//”** será un comentario hasta terminar la línea.
- Todo escrito después de los símbolos **“/*”** será un comentario hasta el cierre del comentario con los símbolos **“*/”**

# 03 // Programa 01 - BLINK

## Programa 01: Blink (parpadeo)

**código: Educar_Blink**

El objetivo de este programa es encender y apagar un led con un intervalo determinado. Lo primero es plantear lógicamente nuestro programa: identificar qué acciones se realizarán una solo vez y cuales se repetirán.

La acción que solo debemos ejecutar una vez es:

- Definir que pin utilizaremos

Las acciones que van a repetirse muchas veces son:

- Encender el pin
- Esperar un tiempo antes de apagarlo
- Apagar el pin
- Esperar un tiempo antes de prenderlo

Una vez planeado nuestro programa, solo falta escribirlo en el Lenguaje Arduino para que el mismo pueda interpretarlo.

Lo primero es establecer en que pin del Arduino conectaremos el Led. Elegiremos uno cualquiera, por ejemplo el "Pin 8". Esto debemos establecerlo en el código dentro de **"setup()"**, porque allí se definen las configuraciones de inicio del Arduino y solo necesitamos configurarlo una vez. Para hacerlo utilizamos la función **"pinMode()"** que lleva dos parámetros:

```
pinMode (número del pin, que modo);
```

Para nuestro ejemplo sería:

```
pinMode(8,OUTPUT);
```

Hay que respetar la formalidad del lenguaje, por lo tanto se tiene que escribir el código tal cual esta aquí respetando las mayúsculas y minúsculas. Notar que al terminar la función se agrega un **";"**.

El modo "OUTPUT" establece que el pin se usará como salida. Cuando se encienda correrá por una tensión de 5v y 40mA.

Para dar la orden de encendido debe utilizarse una nueva función que posee la siguiente formalidad:

```
digitalWrite (número de pin, estado del pin);
```

En nuestro ejemplo es:

```
digitalWrite(8,HIGH);
```

Esto se entiende como: escribir digitalmente en el "pin 8", esa escritura es un encendido (estado HIGH). Para apagar se usa la misma función utilizando el parámetro LOW (apagado).

```
digitalWrite(8,LOW);
```

Ahora solo nos falta un detalle más: entre el encendido y apagado, y nuevamente antes del encendido tenemos que definir el tiempo de espera entre acción y acción. Para ello utilizamos la siguiente función:

```
delay(milisegundos);
```

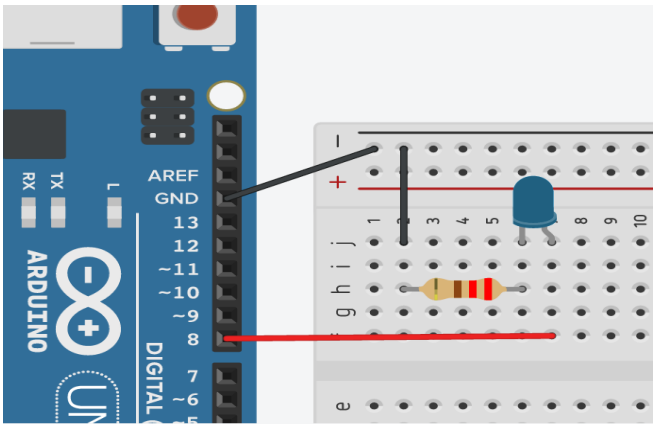
La función **"delay()"** detiene al Arduino en un tiempo establecido en milisegundo. Esta función requiere como parámetro cuantos milisegundos queremos que el Arduino no realice ninguna acción. En este caso usaremos una espera de un segundo: 1000 milisegundos.

```
delay(1000);
```

El código completo sería el siguiente: **código: Educar_Blink**

```
Educar_Blink §
void setup() {
 pinMode(8,OUTPUT);//pin 8 como salida
}

void loop() {
 digitalWrite(8,HIGH);//encender Pin 8
 delay(1000);//esperar 1000 milisegundos
 digitalWrite(8,LOW);//apagar Pin 8
 delay(1000);//esperar 1000 milisegundos
}
```

Por último falta realizar el conexionado. A diferencia del anterior, el led se conectará al negativito común del protoboar y el positivo será el “Pin 8” que programamos en el Arduino.




*Una advertencia importante: no conectar más de un led por pin. Los pines tienen una corriente máxima de 40mA. Un led consume 20 mA. Si conectáramos dos tendríamos un consumo de 40mA que sería lo máximo posible. Cruzar el limite de consumo causa roturas*

*parciales o totales del Arduino. Existe la posibilidad de conectar más led por pin, para ello hay que desarrollar un circuito de potencia que escapa a este primer taller.*

## Cargar Programa: Paso 1.

Una vez escrito el programa hay que cargarlo a la placa Arduino. Es aconsejable que antes de cargarlo, primero nos aseguraremos que el programa no tenga errores. Esto se realiza presionando el botón “Verificar”. Al verificar, Arduino realiza una compilación y marca los errores de escritura del código. No corrige aspectos lógicos del programa, Arduino desconoce nuestros objetivos.

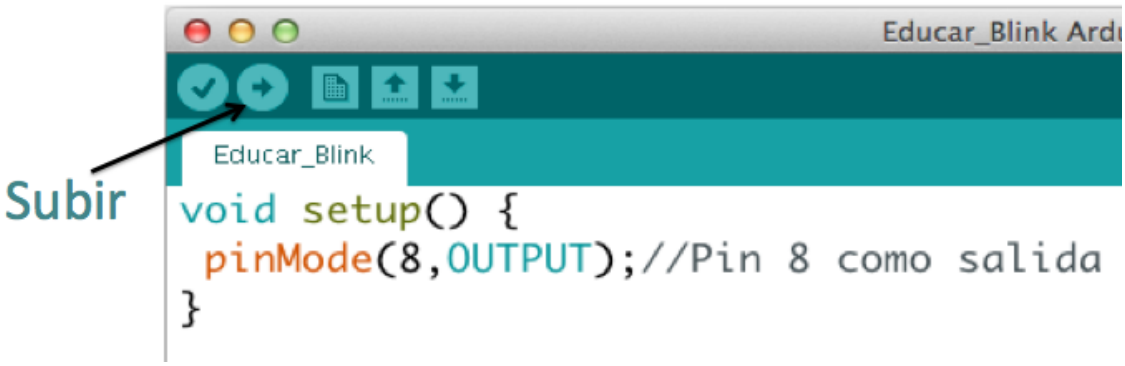

En caso que en la consola aparezcan errores, verificar el programa porque existe algún error. Error típico es la falta de los signos ”;”, la falta de llaves y la falta de algún paréntesis. La mayoría de las funciones o palabras pertenecientes al lenguaje Arduino son resaltadas con colores. Estar atentos si permanecen en color negro: esto significa que hay algo mal escrito. Hay que respetar las formalidades del lenguaje.

## Cargar Programa: Paso 2.

Antes de cargar hay que verificar que tengamos seleccionada la placa Arduino adecuada. Existen varios modelos, tenemos que seleccionar el correcto del menú. (Menú / herramientas / placas / Genuino Uno ).

## Cargar Programa: Paso 3.

Primero hay que conectar el Arduino al puerto USB de la computadora si es que no está hecho aún. Luego presionar el botón de “subir”.
El proceso llevará unos segundos. El Arduino prenderá y apagará varias luces durante la carga. Al terminar se iniciará el proceso: ejecutar el programa.




## Cargar Programa: error de puerto.

Puede surgir el siguiente error en la consola:

```
Avrdude: ser_open(): can´t open divice ”xxxxx”:
no such file or directory.
```

Esto significa que no está conectado el Arduino o está en un puerto equivocado. Para corregir el error probar en todos los puertos disponibles en el menú o ir al Administrador de Hardware del sistema operativo y verificar en que puerto está conectado el Arduino.

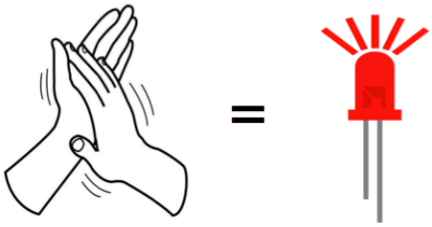
# 04 // Programa 02 - BLINK a RITMO

## Programa 02: Blink a ritmo

**código: Educar_Blink_RitmoA**

**Educar_Blink_RitmoB**

El objetivo de este programa es encender y apagar un led a ritmos sonoros propuestos por el docente. El docente propondrá ritmos musicales con aplausos. Y los programas deberán hacer que el led siga ese ritmo.



Dentro de la carpeta de los ejemplos, hay archivos de audio que ejemplifican esta actividad.

# 05 // Programa 03 - BLINK X3

## Programa 03: Blink X3

**código: Educar_BlinkX3_A**

**Educar_BlinkX3_B**

**Educar_BlinkX3_C**

**Educar_BlinkX3_D**

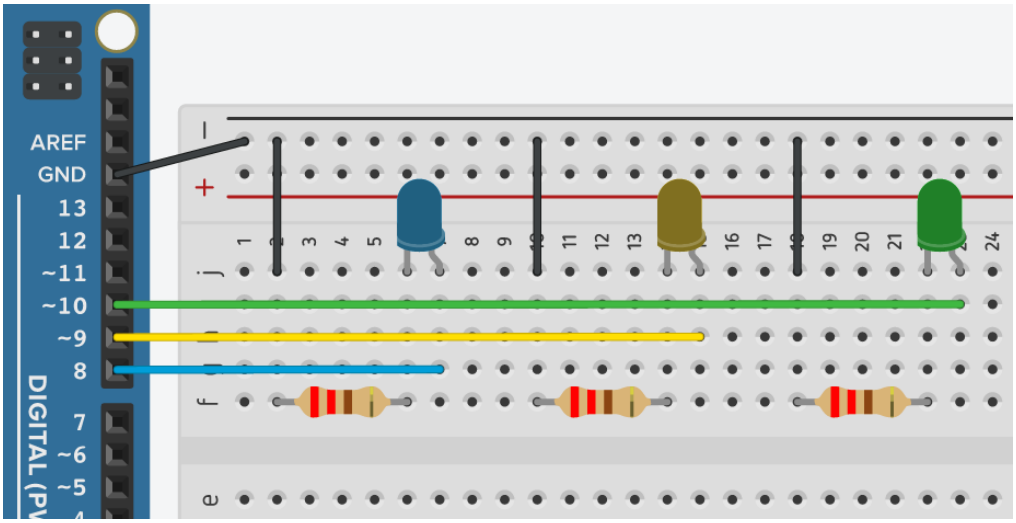
Funciones a utilizar:

**`pinMode`** (qué pin, qué modo);

**`digitalWrite`** (qué pin, qué estado);

**`delay`** (milisegundos);

El objetivo de este trabajo es encender tres leds en distintos orden. Primero realizaremos las conexiones seleccionando los pines que utilizaremos para los leds: pin 8, pin 9 y pin10.




Lo próximo es realizar el programa. Las funciones a utilizar son las mismas del primer programa, pero al tener tres led podemos realizar distintas secuencias

con el mismo conexionado y las mismas funciones de código. Lo que se cambia es el orden de las mismas para obtener distintos resultados.

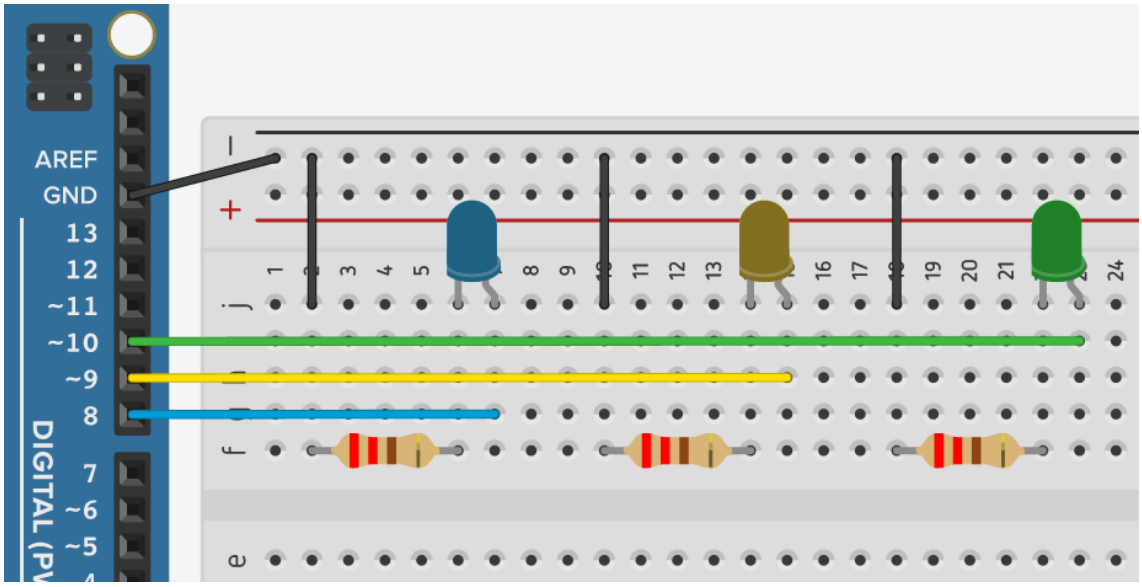
Educar_BlinkX3_A §

```
void setup() {
  pinMode(8,OUTPUT);//pin 8 como salida
  pinMode(9,OUTPUT);//pin 9 como salida
  pinMode(10,OUTPUT);//pin 10 como salida
}

void loop() {
  digitalWrite(8, HIGH);//encender Pin 8
  delay(500);//esperar 500 milisegundos
  digitalWrite(9, HIGH);//encender Pin 9
  delay(500);//esperar 500 milisegundos
  digitalWrite(10, HIGH);//encender Pin 10
  delay(500);//esperar 500 milisegundos
  digitalWrite(8, LOW);//apagar Pin 8
  digitalWrite(9, LOW);//apagar Pin 9
  digitalWrite(10, LOW);//apagar Pin 10
  delay(500);//esperar 500 milisegundos
}
```

## Ejercicio 2 : Teatro de sombras

Generar una historia para tres personajes, con diálogos incluidos. Colocar delante de los led figuras recortadas de estos personajes. Calcular el tiempo del dialogo de cada uno de ellos, teniendo en cuenta el orden.

Por último generar un programa que respete ese orden y el tiempo del led para el dialogo. Hacer la representación de la escena donde cada personaje solo hable cuando este iluminado.



# 06 // Programa Variables

## Programa 04: Variables

**código: Educar_Variables_BlinkX3**

Una variable es un espacio de la memoria en el que se puede asignar valores. Los valores pueden verse, cambiarse y realizar cálculos con ellos. Para acceder a ellos se los denomina bajo un nombre que el programador designe. Por lo general, se le designa un nombre que tenga relación con el contenido. En principio no parecen de mucha utilidad, pero es una de las herramientas más potentes de la programación. Con su utilización los programas se tornan más legibles y flexibles.

> ***Ejemplos de nombres:*** *si se va a guardar en la variable el número del pin donde conectamos un led azul, el nombre que se le asigna a la variable podría ser: pin_led_azul o pinAzul o pinLA, etc. Los nombres pueden contener caracteres, guión bajo y números. El nombre posee dos restricciones: el primer caracter del nombre no puede ser un número y el nombre no puede ser el de funciones o variables propias de Arduino. Por ejemplo no puede llamarse: 8pin_led porque empieza con un número y tampoco puede llamarse: OUTPUT o pinMode, etc.*

## Definición de variables

Para definir una variable debemos seguir la siguiente formalidad:

**Tipo nombre = contenido;**

El nombre es designado por el programador. El contenido, lo que el programador quiera que se ubique en memoria. Y el tipo de variable depende de lo que guardemos en ella.


Podemos entender las variables como un cajón donde guardamos algo. El nombre es un cartel que ponemos fuera de él para identificarlo. El contenido es lo que ponemos dentro y el tipo es el tamaño del cajón que necesitamos para guardar lo que deseamos.

## Tipo de variables

Existen distintos tipos de variables, los cuatro básicos son:

**`int`** (se utiliza para guardar números enteros entre -32,768 to 32,767)

**`float`** (se utiliza para guardar números decimales hasta 7 dígitos de precisión)

**`String`** (se utiliza para guardar textos)

**`Boolean`** (se utiliza para guardar un bit. Esta variables solo puede tener dos valores posibles true o false. Que es un equivalente a 0 y 1)

## Implementación de las variables

Definiremos una variable para guardar el número de pin que usamos al conectar un led.

```
int pinLedAzul = 8;
```

Aquí definimos y le asignamos el valor de 8 a la variable “pinLedAzul”. Y cada vez que mencionemos en el programa el nombre de la variable Arduino lo remplazará por el contenido de la misma. Por ejemplo:

```
pinMode(pinLedAzul,OUTPUT);
```

Con esta sentencia estamos configurando como salida el número de pin según el contenido de la variable “pinLedAzul”. En nuestro caso “8”.

## Otro ejemplo

```
int espera = 1000;
delay  (espera);
```

¿De cuánto será el tiempo de espera?

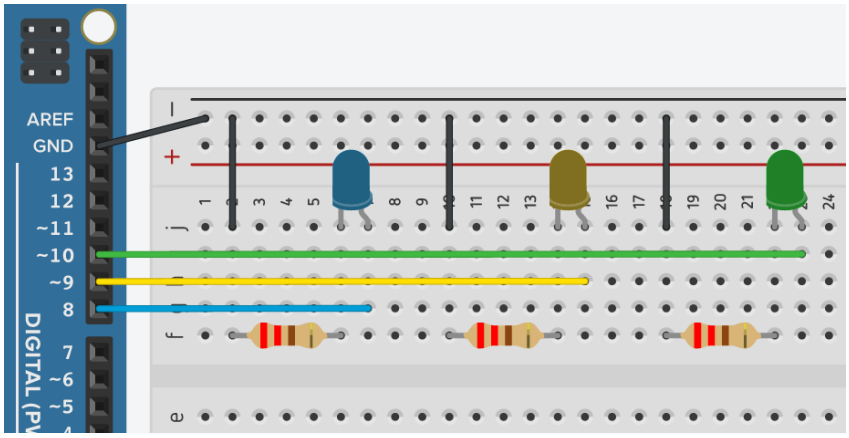
## Beneficios de uso de variables

**código: Educar_Variables_Blinkx3**

Tómenos un código que ya realizamos y rehagámoslo utilizando variables sin modificar el comportamiento: Educar_Blinkx3.

Educar_Variables_BlinkX3

```
int pinLedAzul = 8;
int pinLedAmarillo = 9;
int pinLedVerde = 10;
int espera = 500;

void setup() {
  pinMode(pinLedAzul,OUTPUT);//pin 8 como salida
  pinMode(pinLedAmarillo,OUTPUT);//pin 9 como salida
  pinMode(pinLedVerde,OUTPUT);//pin 10 como salida
}

void loop() {
  digitalWrite(pinLedAzul, HIGH);//encender pin 8
  delay(espera);//esperar 500 milisegundos
  digitalWrite(pinLedAmarillo, HIGH);//encender pin 9
  delay(espera);//Esperar 500 milisegundos
  digitalWrite(pinLedVerde, HIGH);//encender pin 10
  delay(espera);//esperar 500 milisegundos
  digitalWrite(pinLedAzul, LOW);//apagar Pin 8
  digitalWrite(pinLedAmarillo, LOW);//apagar pin 9
  digitalWrite(pinLedVerde, LOW);//apagar pin 10
  delay(espera);//esperar 500 milisegundos
}
```




Uno de los beneficios es que el código se vuelve más legible: ya no vemos en números de pines sino nombres de las variables que podemos identificar rápidamente con el led.

Y el segundo beneficio es la flexibilidad: si cambiamos el contenido de las variables puede cambiar el comportamiento de todo el programa. Por ejemplo si cambiamos el contenido a una variable vinculada a los "delays" cambiará la velocidad de toda la secuencia.

## Cálculos entre variables

El Lenguaje Arduino permite realizar cálculos entre variables y asignarles resultado de cálculos. Podemos hacer sumas, resta, divisiones, multiplicaciones entre otras operaciones. Formalidad:

**Variable = variables + variable;**

Luego del signo de asignación **"="** podemos realizar cálculos entre variables, entre variables y un números o entre números. En este cálculo puede también intervenir la variable a la que le asignamos el valor.

Por último hay que tener presente que la variable a la que le vamos asignar el resultado del calculo sea una variable del mismo tipo del resultado. Por ejemplo: si el resultado es con decimales es conveniente se guarde el resultado en una variable "float" (decimal) y no int (entera). Si no es así, puede perderse información.

# 07 // Programa Números al Azar

## Programa 05, 06 y 07: Random

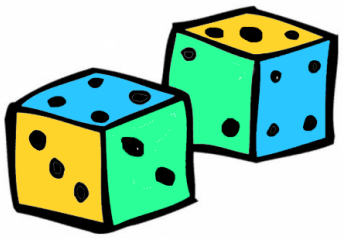
**código: Educar_RandomA**

**Educar_RandomB**

**Educar_RandomC**

En muchas ocasiones queremos utilizar el azar como parte de nuestro programa. Para ello Arduino nos ofrece la función “random()”. Esta función nos devuelve un número al azar entre dos números que nosotros designemos, Por ejemplo:

```
int número = random(0,10);
```



A la variable número le asignamos un valor al azar que este entre 0 al 9,9999. El número 10 queda excluido. Otro ejemplo:

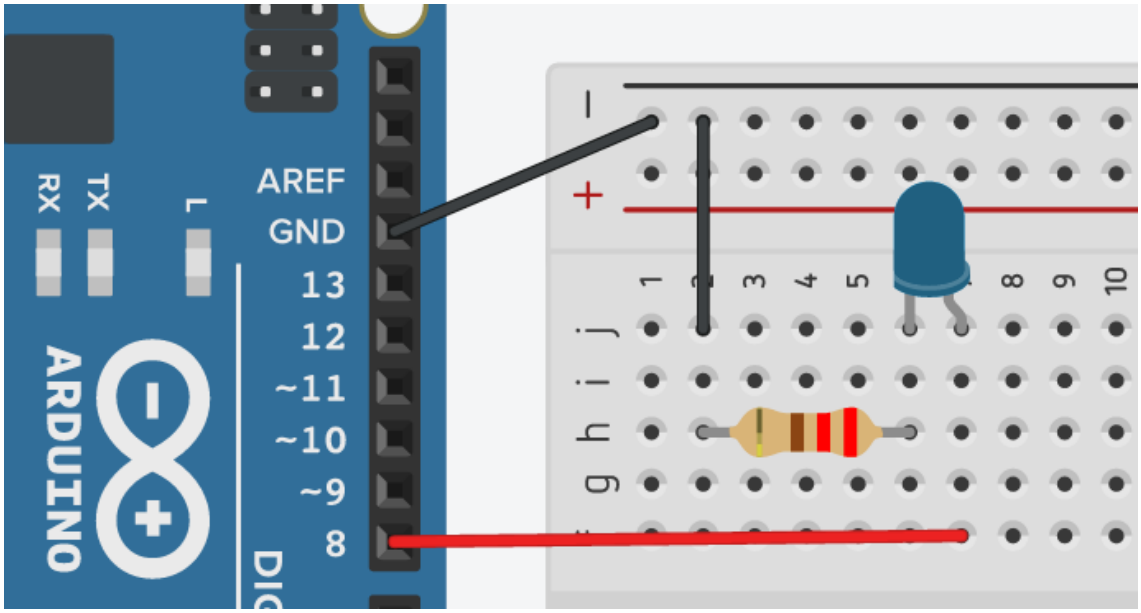
```
int número = random(5, 21);
```

En este caso la variable número tomará un valor de 5 a 20,9999. El número 21 queda excluido.

## Programa 05 : Números al Azar

**Código: Educar_RandomA**

Tomemos como objetivo hacer un programa donde un led se encienda y se apague con tiempos de encendido y apagados al azar.




Educar_RandomA §

```
int pinLed = 8;//pin del led
int espera = 0;//número al azar

void setup() {
  pinMode(pinLed , OUTPUT);//configuración como salida
}

void loop() {
  digitalWrite(pinLed,HIGH);//prende led
  espera = random(10,200);//número al azar
  delay(espera);//delay al azar

  digitalWrite(pinLed,LOW);//apaga led
  espera = random(10,200);//número al azar
  delay(espera);//delay al azar
}
```

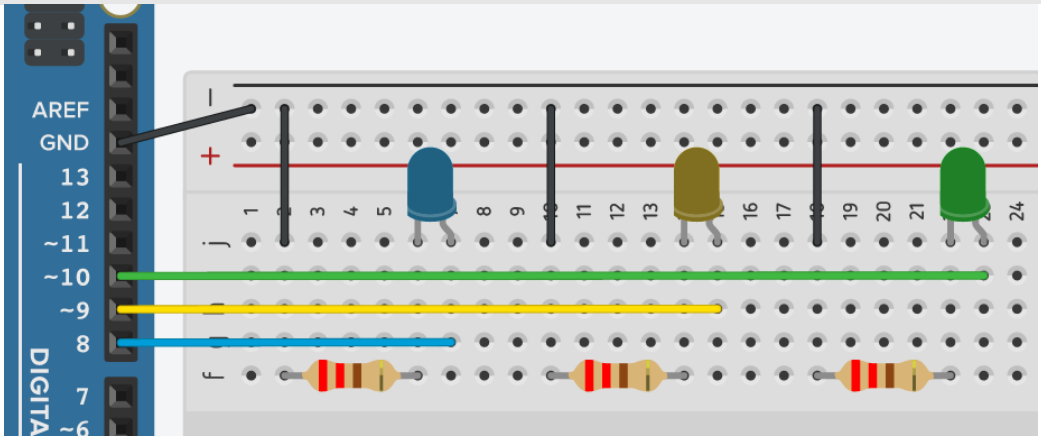
## Programa 06 : Números al Azar

**Código: Educar_RandomB**

Como segundo ejercicio podemos hacer un programa que encienda o no tres led al azar. Para lograrlo usaremos el siguiente truco: ya sabemos que HIGH enciende un pin y LOW lo apaga. HIGH es equivalente a un número 1 y LOW a 0. Por lo tanto si realizo un “random (0,2)” (que solo me dará de retorno un valor de 0 o 1) y el resultado lo utilizo con variables de estado, puedo prender y apagar un led al azar.

Educar_RandomB §

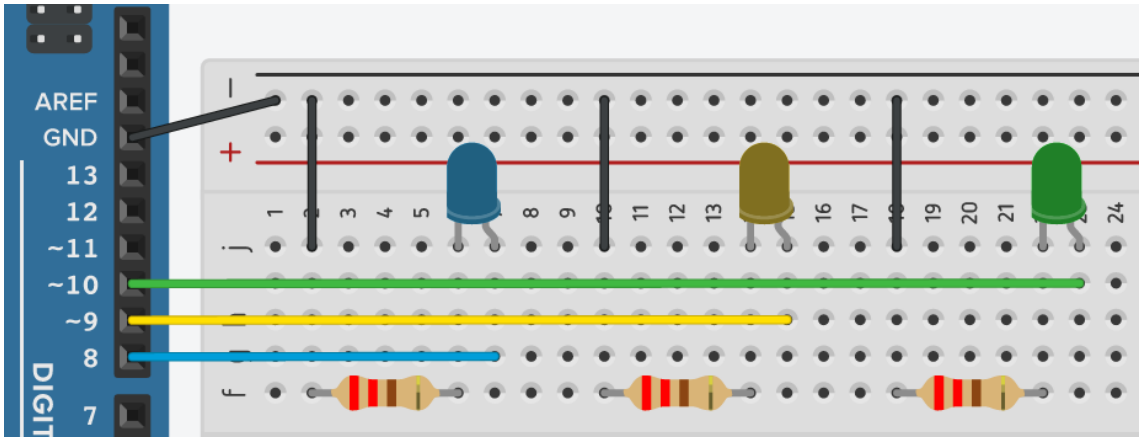
```
int pinLedAzul = 8;//pin del led Azul
int pinLedAmarillo = 9;//pin del led Amarillo
int pinLedVerde = 10;//pin del led Verde

int numero = 0;//número al azar
int espera = 0;

void setup() {
  pinMode(pinLedAzul , OUTPUT);//configuración como salida
  pinMode(pinLedAmarillo , OUTPUT);//configuración como salida
  pinMode(pinLedVerde , OUTPUT);//configuración como salida
}

void loop() {
  numero= random(0,2);//número 0 o 1. 0 = apaga, 1 = prende
  digitalWrite(pinLedAzul,numero);//prende o o apaga led azul

  numero= random(0,2);//número 0 o 1. 0 = apaga, 1 = prende
  digitalWrite(pinLedAmarillo,numero);//prende o  apaga led azul

  numero= random(0,2);//número 0 o 1. 0 = apaga, 1 = prende
  digitalWrite(pinLedVerde,numero);//prende o  apaga led azul

  espera = random(200,1000); // espera al azar
  delay(espera);//delay al azar
}
```



## Programa 07 : Teatro de improvisaciones

**Código: Educar_RandomC**

El objetivo de este ejercicio es generar una obra de teatro con textos improvisados, donde cada personaje hable si esta iluminado. Colocar delante de los led figuras recortadas de estos personajes. Generar un programa que encienda los led con un orden al azar. El tiempo del encendido también puede ser al azar o un tiempo fijo. El alumno debe improvisar el texto de los personajes.



Educar_RandomC

```
int pinLedAzul = 8;//pin del led Azul
int pinLedAmarillo = 9;//pin del led Amarillo
int pinLedVerde = 10;//pin del led Verde

int numero = 0;//número al azar
int espera = 0;

void setup() {
  pinMode(pinLedAzul , OUTPUT);//configuración como salida
  pinMode(pinLedAmarillo , OUTPUT);//configuración como salida
  pinMode(pinLedVerde , OUTPUT);//configuración como salida
}

void loop() {
  numero= random(0,2);//número 0 o 1. 0 = apaga, 1 = prende
  digitalWrite(pinLedAzul,numero);//prende o o apaga led azul

  numero= random(0,2);//número 0 o 1. 0 = apaga, 1 = prende
  digitalWrite(pinLedAmarillo,numero);//prende o  apaga led azul

  numero= random(0,2);//número 0 o 1. 0 = apaga, 1 = prende
  digitalWrite(pinLedVerde,numero);//prende  o apaga led azul

  espera = random(5000,10000);
  delay(espera);//delay al azar
}
```

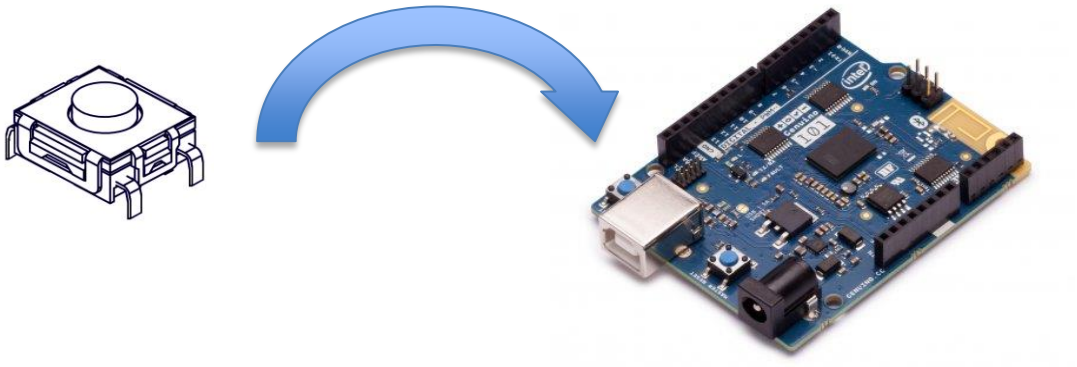
# 08 // Entradas Digitales

## Programa 08 y 09: Botones

**código: Educar_Boton**

**Educar_BotonX2**

Los microcontroladores pueden, no solo encender algunas de su pines, sino también pueden recibir información del exterior en ellos y analizarla. En principio utilizaremos botones y tendremos como objetivo encender y apagar un led en relación al estado de un botón. Realizar el conexionado.



## Entradas Digitales : Botón

**Código: Educar_Boton**

La lectura que haremos del botón será digital, es decir que tendrá dos estados posibles: encendido o apagado (true o false). Para realizar esta acción utilizaremos algunas intrusiones nuevas. Lo primero es definir un pin para conectar el botón: en nuestro caso el “pin 3”. El cual configuramos como entrada (INPUT) a diferencia del led que es una salida (OUTPUT):

```
pinMode(pinBoton, INPUT);
```

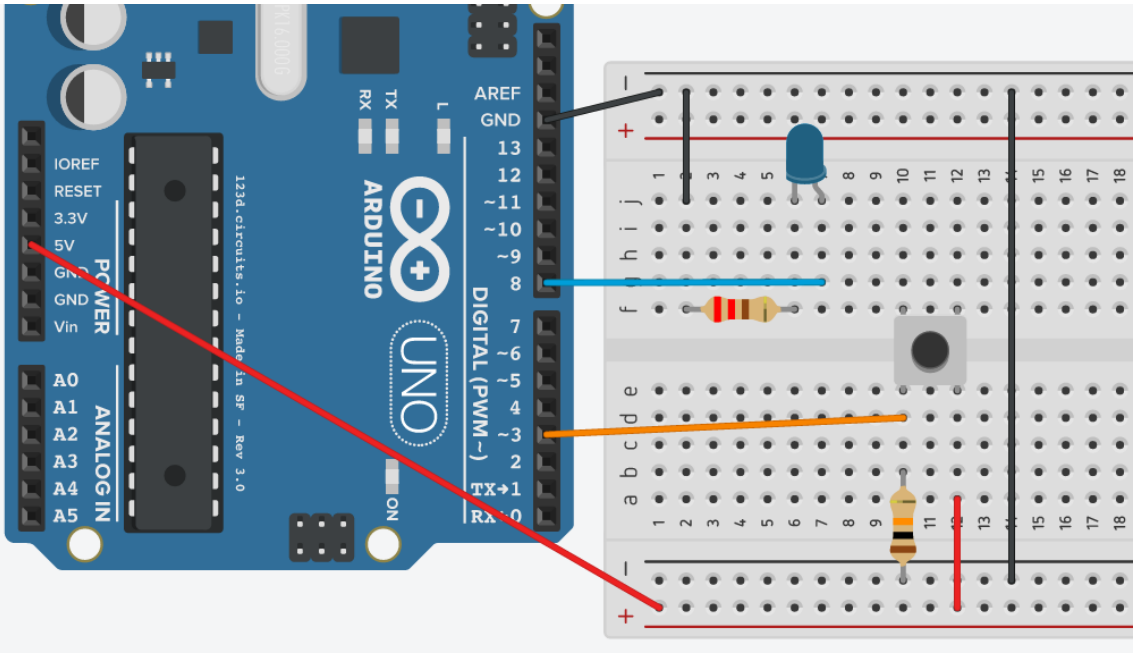
Ya definimos el pin como una entrada, ahora leeremos ese estado. Para leer el estado y no perderlo lo guardaremos en una variable que llamaremos

**“estado”**. Esta variable será del tipo **“boolean”** ya que solo guardará dos estados posibles.

```
estado = digitalRead(pinBoton);
```

Por último relacionaremos el led con el botón de la siguiente manera.

```
digitalWrite(pinLed,estado);
```

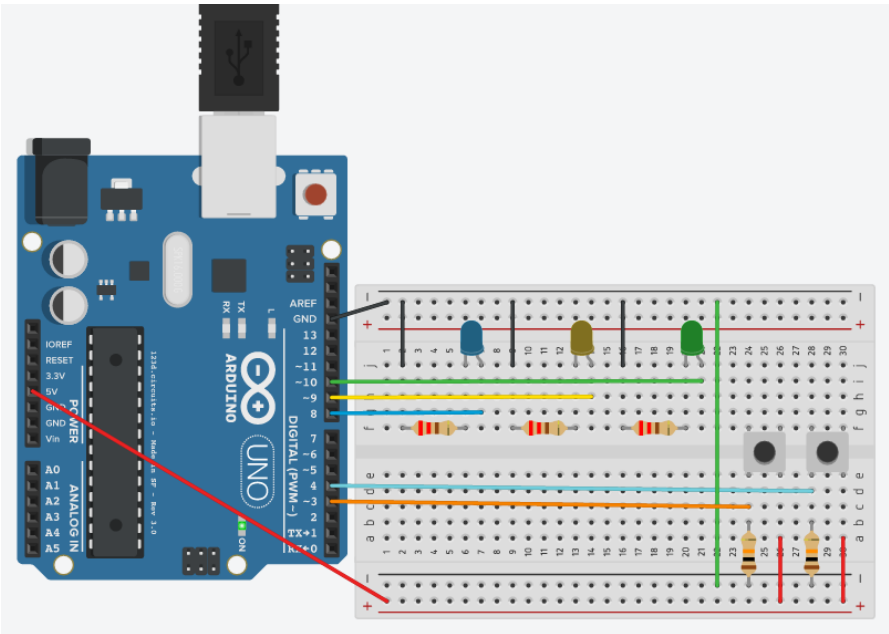



```
Educar_Boton §
int pinLed = 8;//pin del led 8
int pinBoton = 3;//pin del botón con una R de 10k a Gnd
boolean estadoBoton = false;//estado del botón

void setup() {
  pinMode(pinBoton, INPUT);//configuración como entrada
  pinMode(pinLed , OUTPUT);//configuración como salida
}

void loop() {
  estadoBoton = digitalRead(pinBoton);//lee el estado del botón
  digitalWrite(pinLed, estadoBoton);//prende o apaga el led en relación al botón
}
```

## Entradas Digitales : dos botones

**Código: Educar_BotonX2**

Puede utilizarse más de un botón,  por ejemplo dos. En este próximo ejemplo aplicamos el truco de utilizar las valores “true” y “false” que entrega las lectura de los botones, como números (1 y 0). Con ellos podemos realizar operaciones matemáticas. En este ejemplo usamos dos botones, cada uno enciende un led. Pero si presionamos ambos se enciende un tercer led.



Educar_BotonX2 §

```
int pinLedAzul = 8;//pin del led
int pinLedAmarillo = 9;//pin del led
int pinLedVerde = 10;//pin del led

int pinBoton0 = 3;//pin del botón 0 con una R de 10k a Gnd
int pinBoton1 = 4;//pin del botón 1 con una R de 10k a Gnd

boolean estadoBoton0 = false;//estado del botón0
boolean estadoBoton1 = false;//estado del botón1

void setup() {
  pinMode(pinBoton0, INPUT);//configuración como entrada
  pinMode(pinBoton1, INPUT);//configuración como entrada
  pinMode(pinLedAzul , OUTPUT);//configuración como salida
  pinMode(pinLedAmarillo , OUTPUT);//configuración como salida
  pinMode(pinLedVerde , OUTPUT);//configuración como salida
}

void loop() {
  estadoBoton0 = digitalRead(pinBoton0);//lee el estado del botón0
  digitalWrite(pinLedAzul, estadoBoton0);//prende o apaga led Azul en relación al botón0

  estadoBoton1 = digitalRead(pinBoton1);//lee el estado del botón1
  digitalWrite(pinLedVerde, estadoBoton1);//prende o apaga led Verde en relación al botón1

  digitalWrite(pinLedAmarillo, estadoBoton0 * estadoBoton1);//prende el led Amarillo si los dos botones están apretados
}
```

# 09 // Condicionales

## Programa 10, 11 y 12: Condicionales

**código: Educar_if_A**

**Educar_if_B**

Una de las grandes herramientas que proponen los microcontroladres es poder tomar decisiones. Estas decisiones son en base a la lógica que establezca el programador. Todos los lenguajes poseen estas herramientas de una forma u otra. En principio la fusión más básica que utilizaremos es la función “if()” que posee la siguiente formalidad:

```
if(condición){          Si la condición es positiva

} else {                Si la condición es negativa

}
```

Todos los programas que realizamos hasta aquí fueron lineales donde en Arduino realizaba una acción detrás de la otra. A partir de ahora eso cambia, ya que vamos a poder elegir que ejecute distintas acciones según ciertas condiciones: si la condición es positiva: se ejecutarán todas las instrucciones que se encuentren entre las dos primeras llaves. En caso que la condición sea negativa se ejecutarán las instrucciones que se encuentran entre las segundas llaves (luego del **“else”**). El **“else”** y las segundas llaves son opcionales: pueden estar o no.

## Condicionales simples

Los condicionales simples son:

== igual que

< menor que

<= menor o igual que

> mayor que

> mayor o igual que

¡= es distintos que

Los valores dentro de un condicional pueden se literales o variables.

## Programa 10 : Condicionales

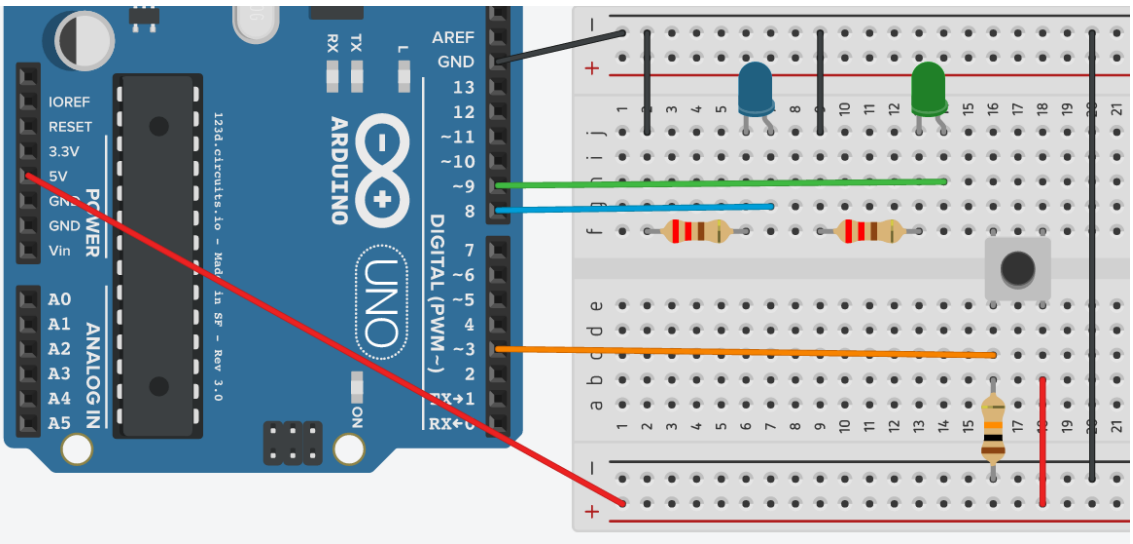
**Código: Educar_if_A**

Hacer un programa que analice el estado de un botón:

- Si esta apretado encienda un led Azul
- Si no está apretado enciende un led Verde

El planteo lógico sería:

- Leer el estado de un botón
- Si esta apretado encender el led azul y apago el verde
- Si no está apretado encender led verde y apago el azul



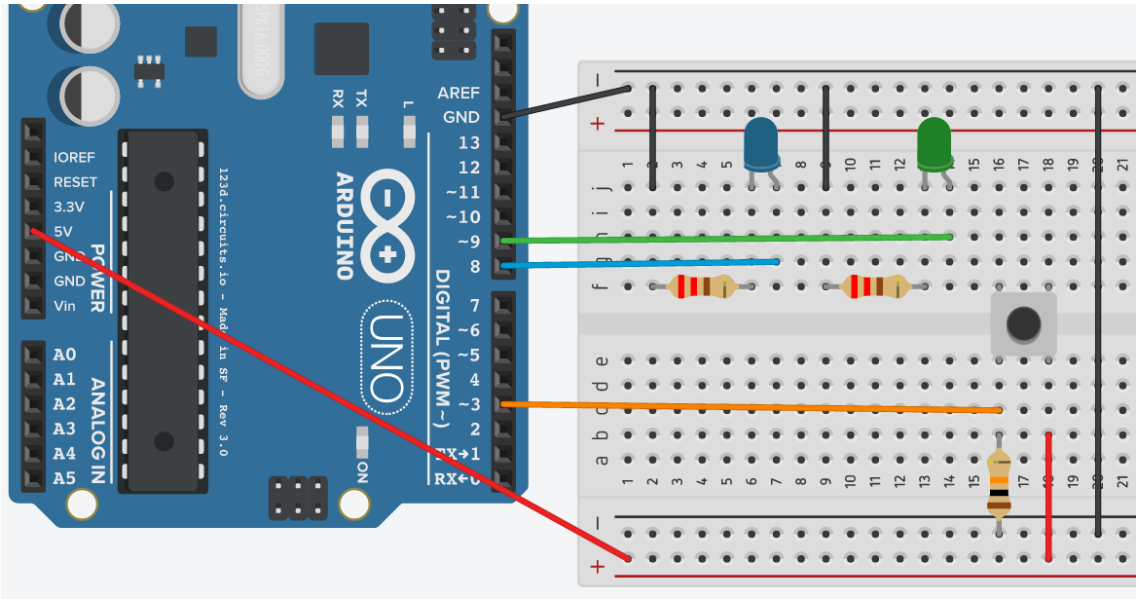

Educar_Boton_if_A §

```
int pinLedAzul = 8;//pin del led Azul
int pinLedVerde = 9;//pin del led Verde

int pinBoton = 3;//pin del botón con una R de 10k a Gnd
boolean estadoBoton = false;//estado del botón

void setup() {
  pinMode(pinBoton, INPUT);//configuración como entrada
  pinMode(pinLedAzul , OUTPUT);//configuración como salida
  pinMode(pinLedVerde , OUTPUT);//configuración como salida
}

void loop() {
  estadoBoton = digitalRead(pinBoton);//lee el estado del botón
  if (estadoBoton == true) {//si el botón está pulsado
    digitalWrite(pinLedAzul, HIGH);//prende led Azul
    digitalWrite(pinLedVerde, LOW);//apaga led Verde
  } else {
    digitalWrite(pinLedAzul, LOW);//apaga led Azul
    digitalWrite(pinLedVerde, HIGH);//prende led Verde
  }
}
```

## Programa 11: Condicionales

**Código:Educar_if_B**

Hacer un programa que analice un botón. Que encienda y apague un led en “blink” a un velocidad fija. Pero cuando apretemos el botón cambiará en velocidad del “blink” a velocidad variable al azar.




Educar_Boton_if_B

```
int pinLedAzul = 8;//pin del led Azul
int pinBoton = 3;//pin del botón con una R de 10k a Gnd
boolean estadoBoton = false;//estado del botón
int espera = 200;
void setup() {
  pinMode(pinBoton, INPUT);//configuración como salida
  pinMode(pinLedAzul , OUTPUT);//configuración como entrada
}

void loop() {
  estadoBoton = digitalRead(pinBoton);//lee el estado del botón
  if (estadoBoton == true) {//si el botón está pulsado
    espera = random(100,500);
  }
  delay(espera);
  digitalWrite(pinLedAzul, HIGH);//prende led Azul
  delay(espera);
  digitalWrite(pinLedAzul, LOW);//apaga led Azul
}
```

## Programa 12: El Adivinador (actividad en el aula)

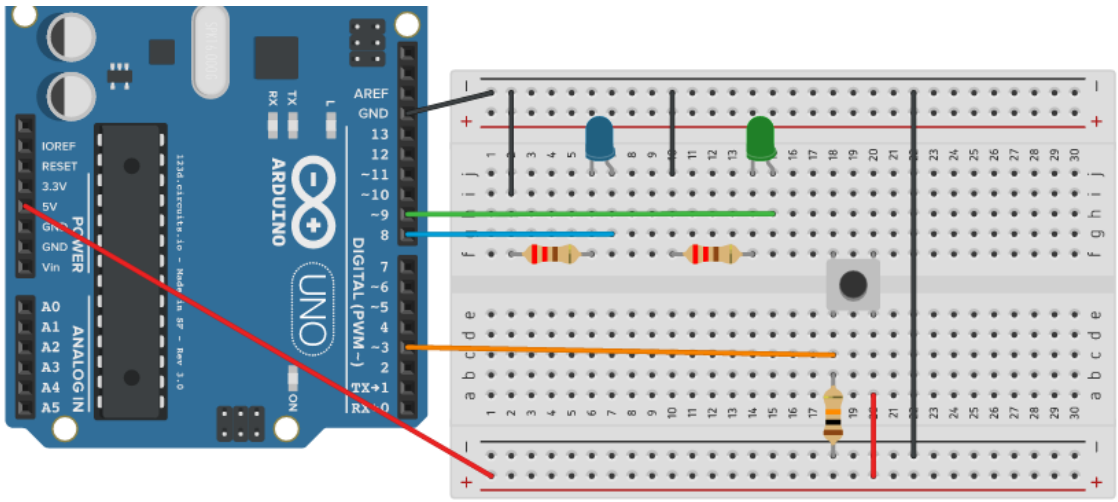
**Código:Educar_adivinado**

Dinámica del juego: el participante hace una pregunta, pulsa el botón y cuando lo suelta el Arduino le contesta por si o por no.

Primero, hacer tres dibujos:

- Un adivinador
- Una leyenda “si”
- Una leyenda “no”

Luego realizar un circuito con un botón y dos led. Colocarle a los led las leyendas de “si” y “no”. Por detrás colocar el “adivinador”.




Por último realizar un programa: que analice el estado de un botón.

- Cuando esté apretado: encienda un led al azar.
- Cuando no esté apretado solo deje encendido el led recientemente seleccionado.

Educar_Adivinador §

```
int pinLedAzul = 8;//pin del led Azul
int pinLedVerde = 9;//pin del led Verde

int pinBoton = 3;//pin del botón con una R de 10k a Gnd
boolean estadoBoton = false;//estado del botón

int numero = 0 ;// si es 0 prendo el led azul, si es 1 prendo led verde

void setup() {
  pinMode(pinBoton, INPUT);//configuración como entrada
  pinMode(pinLedAzul , OUTPUT);//configuración como salida
  pinMode(pinLedVerde , OUTPUT);//configuración como salida
}
void loop() {
  estadoBoton = digitalRead(pinBoton);//lee el estado del botón
  if (estadoBoton == true) {//si el botón está pulsado
    numero = random (0, 2); //resultados posibles 0 o 1
  }
  if (numero == 0) { //si es 0 prendo el led azul
    digitalWrite(pinLedAzul, HIGH); //prende led Azul
    digitalWrite(pinLedVerde, LOW);//apaga led Verde
  } else { // si no es 0 prendo el led verde
    digitalWrite(pinLedAzul, LOW); //apaga led Azul
    digitalWrite(pinLedVerde, HIGH);//prende led Verde
  }
}
```

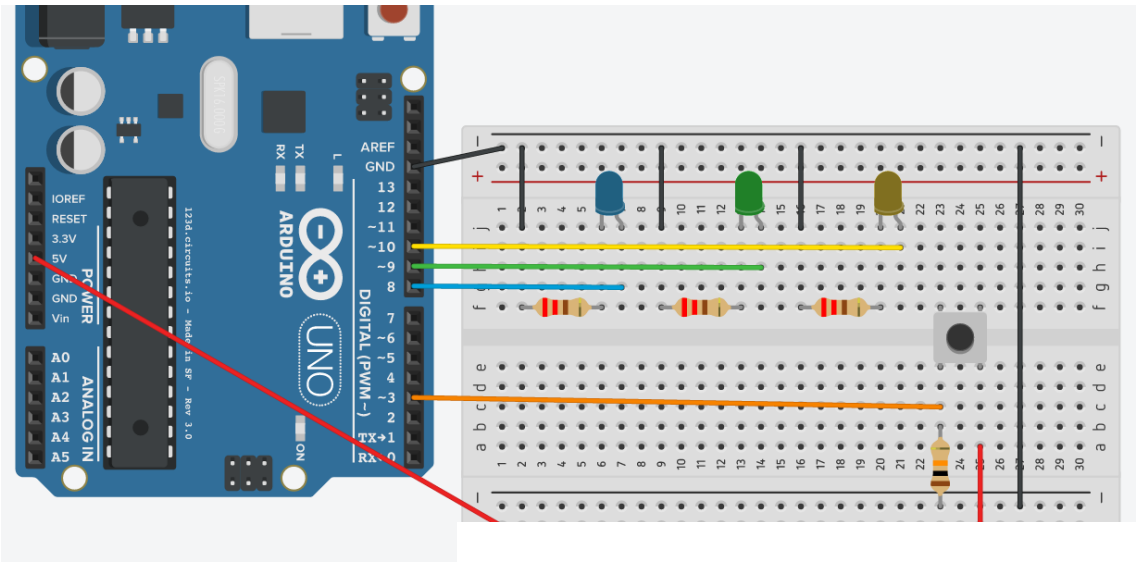
## Programa 13: Piedra, papel o tijera (actividad en el aula)

**Código: Educar_Piedra_Papel_Tijera**

Dinámica del juego: Es para dos jugadores. Un jugador opera el Arduino y el otro juega en forma tradicional. Al iniciar la ronda el operador presiona el botón y al momento de decidir la acción lo suelta. El Arduino seleccionará una de las tres opciones y el otro jugador también.

Primero, hacer tres dibujos:

- Una piedra, un papel y una tijera.
- Condicionales (actividad en el aula)
- Objetivo: Piedra, papel o tijera – Arduino vs Arduino




Educar_Piedra_Papel_Tijera §

```
int pinLedAzul = 8;//pin del led Azul
int pinLedVerde = 9;//pin del led Verde
int pinLedAmarillo = 10;//pin del led Amarillo

int pinBoton = 3;//pin del botón con una R de 10k a Gnd
boolean estadoBoton = false;//estado del botón

int numero = 0 ;// si es 0 prendo el led azul, si es 1 prendo led verde y 3 el led amari

void setup() {
  pinMode(pinBoton, INPUT);//configuración como entrada
  pinMode(pinLedAzul , OUTPUT);//configuración como salida
  pinMode(pinLedVerde , OUTPUT);//configuración como salida
  pinMode(pinLedAmarillo , OUTPUT);//configuración como salida
}

void loop() {
  estadoBoton = digitalRead(pinBoton);//lee el estado del botón
  if (estadoBoton == true) {//si el botón está pulsado
    numero = random (0, 3); //resultados posibles 0,1 o 2
  }

  if (numero == 0) { //si es 0 prendo el led azul
    digitalWrite(pinLedAzul, HIGH); //prende led Azul
    digitalWrite(pinLedVerde, LOW);//apaga led Verde
    digitalWrite(pinLedAmarillo, LOW);//apaga led Amarillo
  }

  if (numero == 1) { // si es 1 prendo el led verde
    digitalWrite(pinLedAzul, LOW); //apaga led Azul
    digitalWrite(pinLedVerde, HIGH);//prende led Verde
    digitalWrite(pinLedAmarillo, LOW);//apaga led Amarillo
  }

  if (numero == 2) { // si no es 0 prendo el led verde
    digitalWrite(pinLedAzul, LOW); //apaga led Azul
    digitalWrite(pinLedVerde, LOW);//apaga led Verde
    digitalWrite(pinLedAmarillo, HIGH);//prende led Amarillo
  }
}
```

## Programa 14: Piedra, papel o tijera

## Arduino vs Arduino (actividad en el aula)

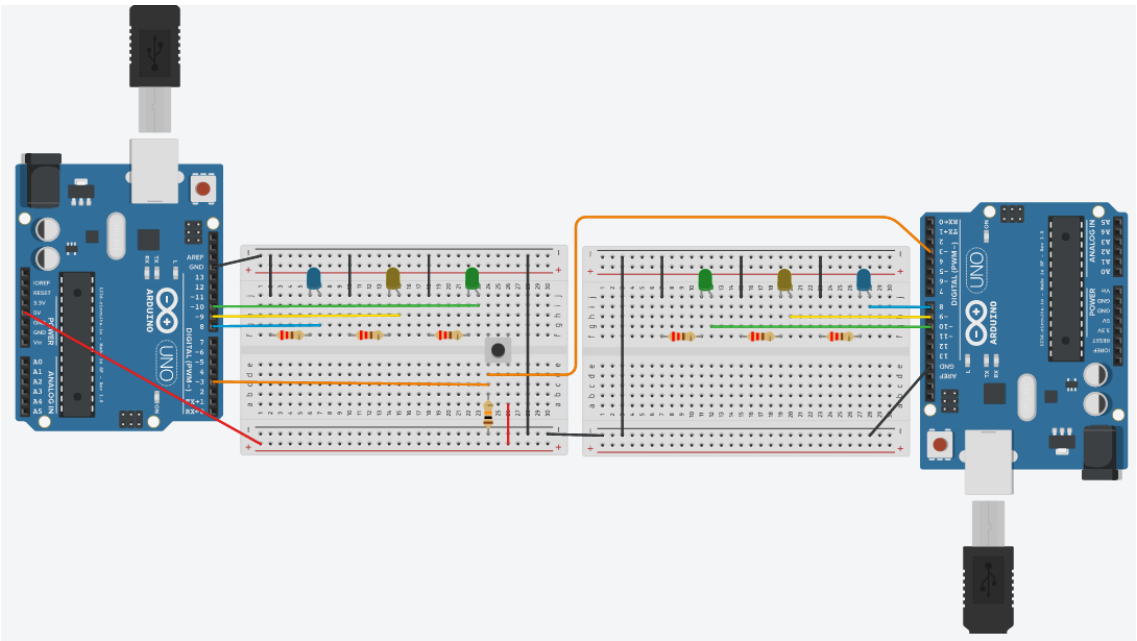
**Código: Educar_Piedra_Papel_Tijera**

Dinámica del juego: Es para un jugador. Un jugador opera con el mismo botón los dos Arduinos. Al presionar y soltar el botón los dos Arduinos seleccionarán su jugada.

Primero, hacer seis dibujos:

- Dos piedra
- Dos papel
- Dos tijera.

Tener en cuenta que el programa es el mismo para los dos Arduinos. Para que un botón sea leído en forma correcta, hay que unir los negativos de ambas placas. Es decir hacer una conexión entre los negativos de los dos Arduinos. Solamente los negativos, no unir los positivos.



Educar_Piedra_Papel_Tijera §

```
int pinLedAzul = 8;//pin del led Azul
int pinLedVerde = 9;//pin del led Verde
int pinLedAmarillo = 10;//pin del led Amarillo

int pinBoton = 3;//pin del botón con una R de 10k a Gnd
boolean estadoBoton = false;//estado del botón

int numero = 0 ;// si es 0 prendo el led azul, si es 1 prendo led verde y 3 el led amaril

void setup() {
  pinMode(pinBoton, INPUT);//configuración como entrada
  pinMode(pinLedAzul , OUTPUT);//configuración como salida
  pinMode(pinLedVerde , OUTPUT);//configuración como salida
  pinMode(pinLedAmarillo , OUTPUT);//configuración como salida
}

void loop() {
  estadoBoton = digitalRead(pinBoton);//lee el estado del botón
  if (estadoBoton == true) {//si el botón está pulsado
    numero = random (0, 3); //resultados posibles 0,1 o 2
  }

  if (numero == 0) { //si es 0 prendo el led azul
    digitalWrite(pinLedAzul, HIGH); //prende led Azul
    digitalWrite(pinLedVerde, LOW);//apaga led Verde
    digitalWrite(pinLedAmarillo, LOW);//apaga led Amarillo
  }

  if (numero == 1) { // si es 1 prendo el led verde
    digitalWrite(pinLedAzul, LOW); //apaga led Azul
    digitalWrite(pinLedVerde, HIGH);//prende led Verde
    digitalWrite(pinLedAmarillo, LOW);//apaga led Amarillo
  }

  if (numero == 2) { // si no es 0 prendo el led verde
    digitalWrite(pinLedAzul, LOW); //apaga led Azul
    digitalWrite(pinLedVerde, LOW);//apaga led Verde
    digitalWrite(pinLedAmarillo, HIGH);//prende led Amarillo
  }
}
```

# 10 // Servo Motores

## Programa 15, 16, 17 y 18:

**código: Educar_ServoA**

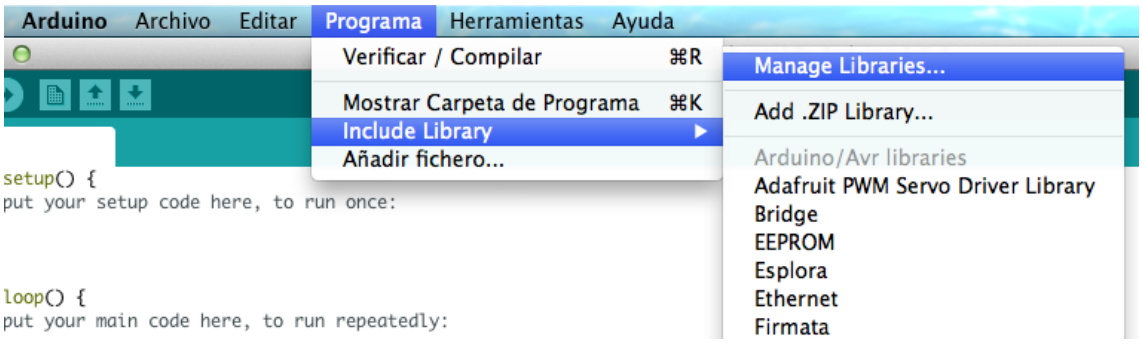
**Educar_ServoB**

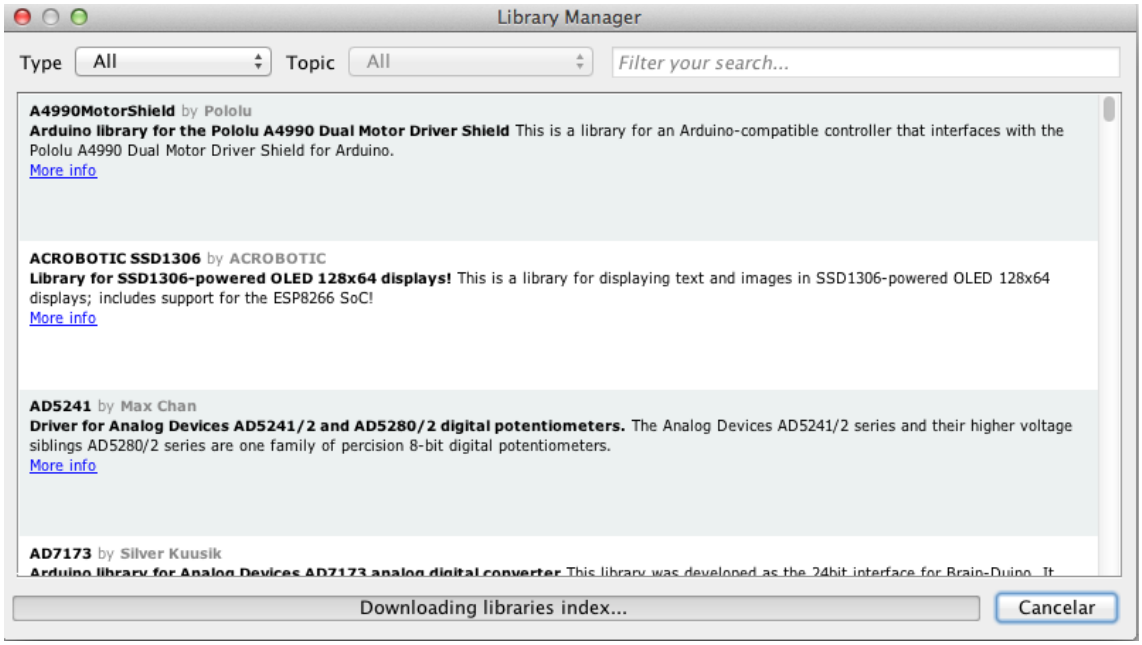
El servomotor es un motor eléctrico de corriente continua con un juego de engranajes que aumenta su torque y disminuye su velocidad. Y también posee electrónica interna que nos permite controlar su posición dentro de su rango de operación. El servomotor posee tres cables de conexión: uno positivo, uno negativo y uno de control.

Las librerías son códigos escritos por algún programador que podemos incluir en el nuestro. Al incluirlo se agrega en forma invisible al nuestro. Para utilizar las librerías hay que realizar dos pasos:

- Instalar las librerías en la computadora.
- Incluir la librerías en nuestro código.

Para instalar las librerías Arduino nos ofrece un instalador que permite buscar e instalar librerías online.






Las librerías **"Servo"** ya están incluidas en Arduino, pero si debemos incluirlas, hay que conocerlas para utilizarlas. Para incluir la librería en nuestro programa tenemos que hacerlo al inicio del mismo:

```
#include <Servo.h>
```

Notar que al final de la instrucción no lleva el símbolo **";"**. Con solo esa sentencia ya se incluye el código necesario para controlar el motor. Esta librería es capaz de controlar varios motores. Con nuestro Arduino solo podemos controlar seis a la vez ya que posee solo 6 pines especiales que pueden hacerlo (salidas con pwm). Son los pines: 3,5,6,9,10 y 11 únicamente.

El próximo paso es crear el objeto de la librería **"Servo"**. En pocas palabras: hay que decirle al Arduino cuantos motores usaremos y que nombre tienen. Esta formalidad va debajo de la inclusión de la librería:

`Servo miServo;//` **"miServo"** será el nombre del motor dentro de nuestro programa

Para terminar con la configuración del objeto **“Servo”**, y ya dentro del **“setup”**, hay que establecer que pin usaremos para el control: solamente pueden ser los pines especiales 3,5,6,9,10 u 11. En nuestro ejemplo usaremos el pin: 9.

```
miServo.attach(9);
```

Ya esta todo listo para poder controlar nuestro servomotor. Solamente nos falta indicarle el ángulo de giro donde queremos que se coloque:

```
miServo.write(ángulo);
```

Este tipo servomotor posee un rango de operación de 180 grados, de 0 a 179 como máximo. La velocidad es constante. Lo que hay que tener en cuenta es que es un objeto mecánico que demora en sus movimientos. Si por ejemplo queremos que se sitúe en un ángulo y luego en otro, entre instrucción e instrucción hay que darle un tiempo de respuesta mecánica (un delay). Si no lo prevemos solo veremos que se posiciona en el último ángulo designado.

### Ejemplo:

```
miServo.write(0);
delay(2000);//espera para observar en cambio de posición
miServo.write(90);
delay(2000);//espera para observar en cambio de posición
miServo.write(0);
```

### Ejemplo:

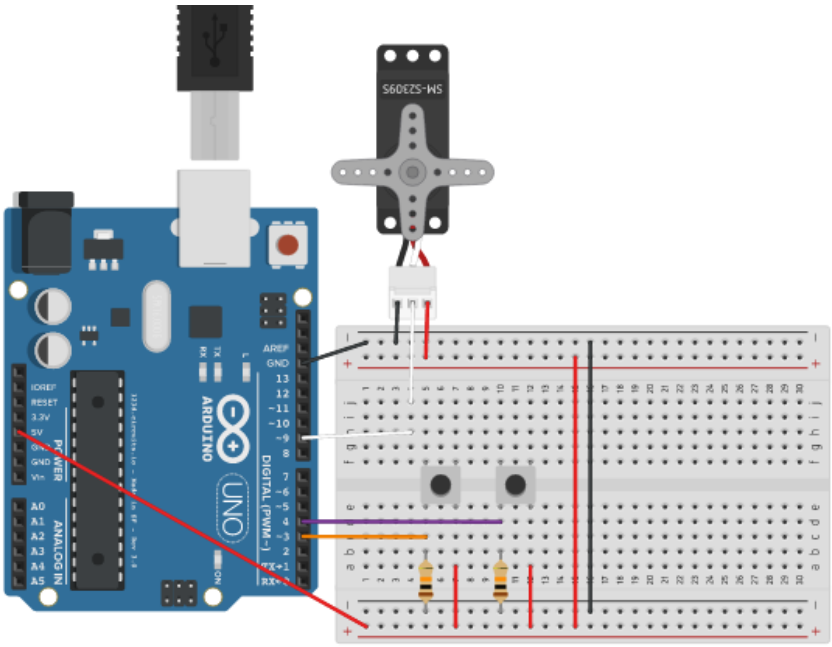
```
miServo.write(0);
miServo.write(90);
miServo.write(0);//como no hay espera el servo no llegará
                 al grado 90.
```

## Programa 15: Servo motor

**Código: Educar_ServoA**

Hacer sistema con dos botones y un servomotor. Establecer cuatro posiciones para el motor y que varíe entre ellas dependiendo si los botones están pulsados o no.

| Tabla de posiciones | | | | |
|---|---|---|---|---|
| Botón 0 | | Botón 1 | | Angulo |
| No | Si | No | Si | |
| X | | X | | 0 |
| | X | X | | 60 |
| X | | | X | 120 |
| | X | | X | 179 |



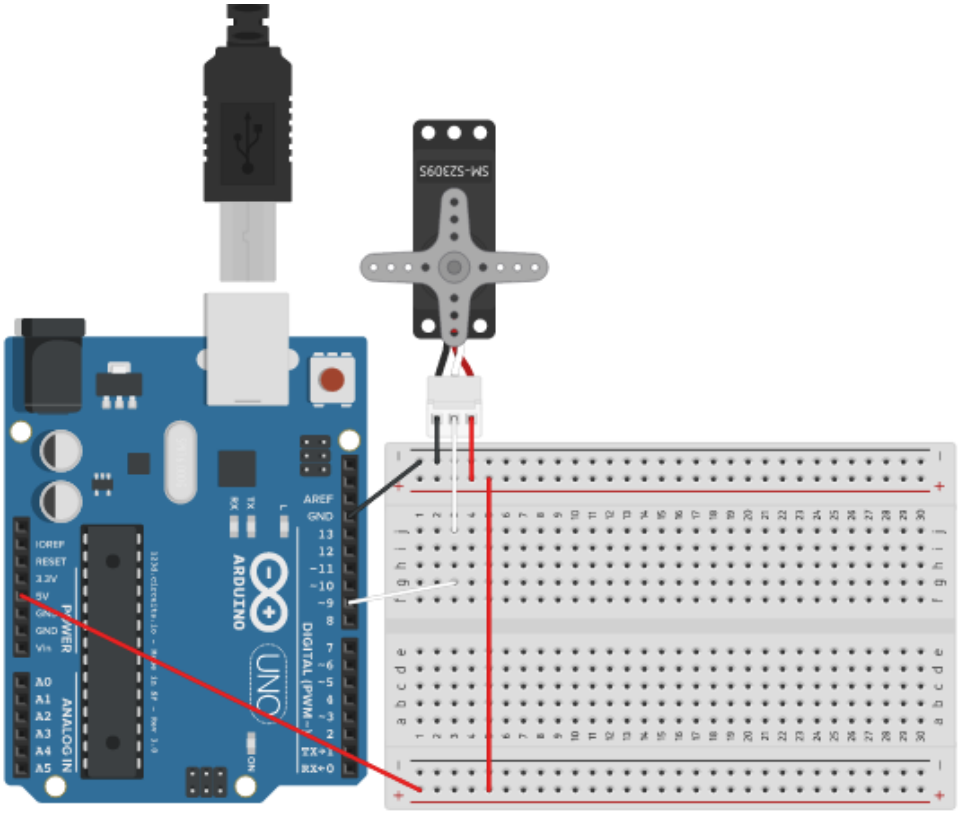
Educar_ServoA §

```
#include <Servo.h>//Incluye librería controladora del servo
Servo miServo;  // Crea el objeto Servo

int pinServo = 9;
int pinBoton0 = 3;
int pinBoton1 = 4;

boolean estadoBoton0 = false;
boolean estadoBoton1 = false;
int posicion = 0;

void setup()
{
  miServo.attach(pinServo); //Defino el pin de información para el motor
  pinMode(pinBoton0, INPUT);//configuración como entrada
  pinMode(pinBoton1, INPUT);//configuración como entrada
}

void setup()
{
  miServo.attach(pinServo); //Defino el pin de información para el motor
  pinMode(pinBoton0, INPUT);//configuración como entrada
  pinMode(pinBoton1, INPUT);//configuración como entrada
}

void loop()
{
  estadoBoton0 = digitalRead(pinBoton0);
  estadoBoton1 = digitalRead(pinBoton1);
  posicion = 0; //posición inicial del motor
  if (estadoBoton0 == true && estadoBoton1 == true) { //Si los dos botones estan presionados. && = Y
    posicion = 179;
  } else {
    if (estadoBoton0 == true) {
      posicion = 60;
    }
    if (estadoBoton1 == true) {
      posicion = 120;
    }
  }
  miServo.write(posicion);//cambia posición del motor
  delay(15);
}
```

## Programa 16: Servo motor

**Código: Educar_ServoB**

Hacer un programa que controle un servomotor. Que lleve al motor por todos los ángulos: de 0 a 179 y nuevamente a 0.



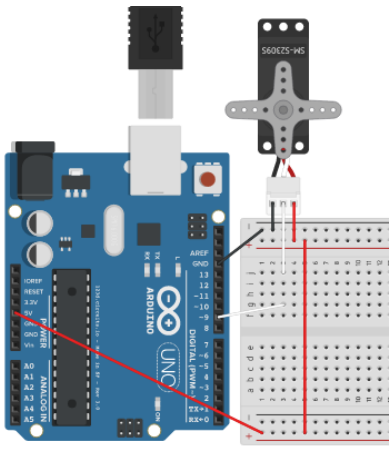
Educar_SerialB §

```
int pinLedAzul = 8;
int pinBoton = 3;
boolean estadoBoton = false;
int variable = 0;
int espera = 1000;

void setup() {
  Serial.begin(9600);
  pinMode(pinBoton, INPUT);//configuración como entrada
  pinMode(pinLedAzul, OUTPUT); ////configuración como salida
}

void loop() {
  variable = variable - 1;//decrece el valor de la variable
  estadoBoton = digitalRead(pinBoton);//lee el estado del botón

  if (estadoBoton == true) {//si el botón está pulsado
    if (variable > 50) {//si la variable no paso los 50, el jugador perdió
      Serial.print("Perdio: Muy pronto: ");//muestra leyenda en el puerto
      Serial.println(variable);
      variable = random(100, 200); //reinicio el valor para otra jugada
      delay(espera);
      digitalWrite(pinLedAzul, HIGH);//prende led Azul
      delay(espera);
      digitalWrite(pinLedAzul, LOW);//apaga led Azul
    }else{//si no es menor que 50, el jugador ha ganado
      variable = random(100, 200);//reinicio el valor para otra jugada
      Serial.print("Gana. Otro intento: ");//muestra leyenda en el puerto
      delay(espera);
    }
  }

  if (variable <= 0) {//si llega a 0, el jugador perdió
    Serial.println("Perdio: Llego a 0");//muestra leyenda en el puerto
    variable = random(100, 200); //reinicio el valor para otra jugada
    delay(espera);
    digitalWrite(pinLedAzul, HIGH);//prende led Azul
    delay(espera);
    digitalWrite(pinLedAzul, LOW);//apaga led Azul
  }

  Serial.println(variable);//imprimo en el puerto serial el contenido de variable
}
```

## Programa 17: Servo motor

**Código: Educar_ServoC**

Hacer un programa que controle un servomotor. Que lleve al motor por todos los ángulos: de 0 a 179 y nuevamente a 0 a distintas velocidades.



Educar_ServoC §

```
#include <Servo.h>//incluye librería controladora del servo
Servo miServo;  // crea el objeto Servo
int pinServo = 9;///pin de control para el servo
int posicion = 0;
int limite_superior = 179;
int limite_inferior = 0;
int incremento  = 1;
int espera = 15;
void setup()
{
  miServo.attach(pinServo); //defino el pin de información para el motor
}
void loop()

{
  posicion = posicion + incremento;//cambia la posición según la variable incremento

  if (posicion > limite_superior) {//límite superior
    posicion = limite_superior;
    incremento = incremento * -1;// pasa de valor positivo a negativo
    espera = random(5,20);
  }
  if (posicion < limite_inferior) {//límite inferior
    posicion = limite_inferior;
    incremento = incremento * -1;//pasa de valor negativo a positivo
    espera = random(5,20);
  }

  miServo.write(posicion);//cambia posición del motor
  delay(espera);
}
```

## Programa 18: Piedra, papel o tijera (actividad en el aula)

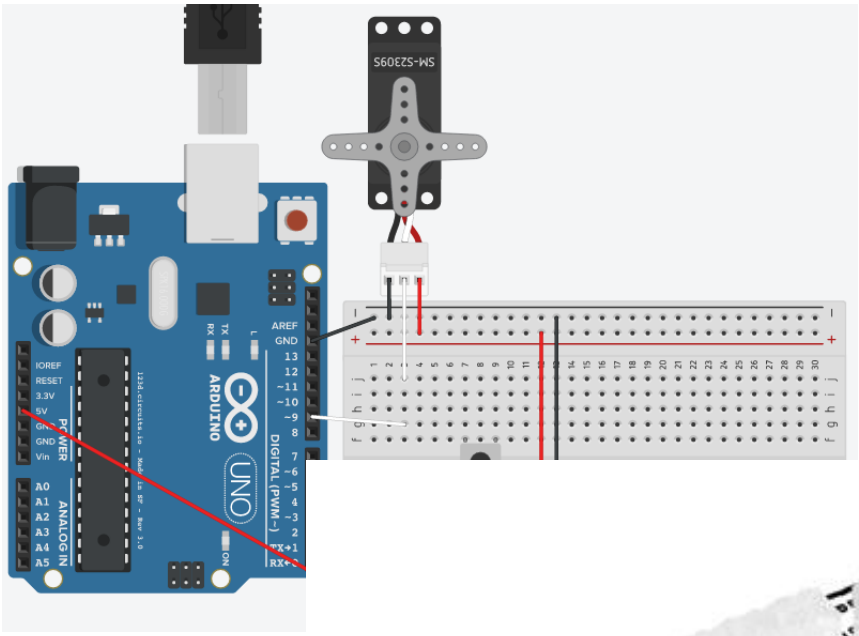
**Código:_ Educar_Servo_Piedra_Papel_Tijera**

Dinámica del juego: el juego es para dos jugadores. Uno de ellos presiona el botón mientras el otro realiza el piedra, papel o tijera. Llegado el momento de seleccionar la figura, el jugador con Arduino suelta el botón. Para mostrar la jugada realizada, Arduino mueve el servomotor con una flecha, señalando la jugada

Primero, hacer cuatro dibujos:

- Una piedra
- Un papel
- Una tijera
- Una flecha.

Colocar la flecha sobre el motor. Designar cuales son los ángulos de movimiento del motor para cada jugada. Luego colocar los dibujos de modo que cuando el motor se posiciona en uno de los ángulos seleccionados, la flecha apuntará a uno de los dibujos. Así el Arduino mostrará la jugada seleccionada.




Educar_Servo_Piedra_Papel_Tijera §

```
#include <Servo.h>//incluye librería controladora del servo
Servo miServo;  // crea el objeto Servo

int pinServo = 9;///pin de control para el servo
int posicion = 0;

int pinBoton = 3;//pin del botón con una R de 10k a Gnd
boolean estadoBoton = false;//estado del botón

int numero = 0 ;// si es 0 prendo el led azul, si es 1 prendo led verde y 2 el led amarillo

void setup() {
  pinMode(pinBoton, INPUT);//configuración como entrada
  miServo.attach(pinServo); //defino el pin de información para el motor
}

void loop() {
  estadoBoton = digitalRead(pinBoton);//lee el estado del botón
  if (estadoBoton == true) {//si el botón esta pulsado
    numero = random (0, 3); //resultados posibles 0, 1 o 2.
    posicion = 0;//posición inicial
  } else {//si el botón no esta presionado

    if (numero == 0) { //si es 0 prendo el verde azul
      posicion = 60;//posición Piedra
    }

    if (numero == 1) { // si no es 0 prendo el led verde
      posicion = 120;//posición Papel
    }

    if (numero == 2) { // si no es 0 prendo el led verde
      posicion = 179;//posición Tijera
    }
  }

  miServo.write(posicion);//cambia posición del motor
  delay(15);
}
```

# ANEXOS

## Ley de ohms:

La relación existente entre la tensión, la corriente y la resistencia es descripta por la ley de ohms.
Donde establece que:

**i = v / r  (corriente = voltaje / resistencia)**
**r = v / i  (resistencia = voltaje / resistencia)**
**v = i * r  (voltaje = corriente * resistencia)**

Para calcular la resistencia utilizaremos la siguiente relación

**r = v / i  (resistencia = voltaje / resistencia)**

Se aplica  de la siguiente forma:

$$R = \frac{\text{(voltaje de la fuente – voltaje del led)}}{\text{corriente necesaria para el led}}$$

Los datos del led pueden variar: son aportados por el fabricante del mismo. Pero utilizamos datos típicos de un led.

**R = (5v – 3V) / 0,020 A**
**R=  2 / 0,020 A**
**R = 100 ohms**

La resistencia para que el led encienda a 5v en forma correcta es de 100 ohms. Una resistencia menor tenderá a sobrecalentar el led hasta quemarlo. Y una resistencia mayor encenderá al led con menor intensidad.