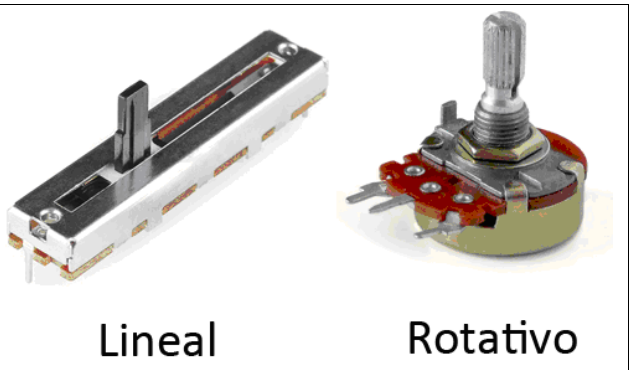
POTENCIOMETRO

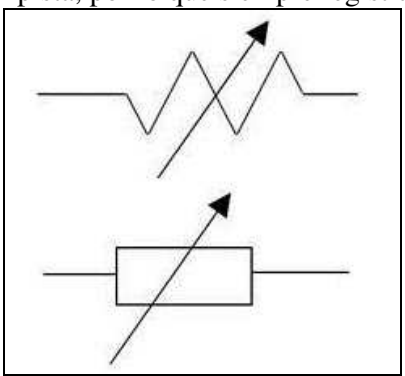
Un potenciómetro es un dispositivo que permite variar su resistencia de forma manual, entre un valor mínimo Rmin, (normalmente 0 ohminos) y un valor máximo Rmax. Valores habituales de Rmax son 5k, 10k o 20k ohmios.



Potenciómetros

Internamente un potenciómetro está constituido por un contacto móvil que se desplaza a lo largo de una pista resistiva. De esta forma, al mover el potenciómetro movemos el contacto a lo largo de la pista, y variando la longitud del tramo de pista con el que estamos en contacto, y por tanto variando su resistencia.

Normalmente un potenciómetro tiene tres terminales. Los dos extremos están unidos a ambos lados de la pista, por lo que siempre registrarán la resistencia máximo Rmax.


Simbología Usada

La terminal restante corresponde con el contacto móvil. Esta terminal varía su resistencia respecto a las otras dos terminales a medida que accionamos el potenciómetro, siendo la suma de la resistencia a las otras terminales igual a Rmax.

Comentario: Los Potenciómetros lineales presentan una proporcionalidad entre resistencia y desplazamiento, lo cuál significa un comportamiento más intuitivo.
Mientras, los exponenciales (Rotativos) permiten mayor precisión en valores de resistencia bajos, por lo que resultan adecuados cuando hace falta un ajuste fino en un amplio rango.

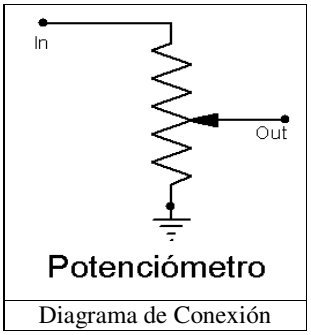
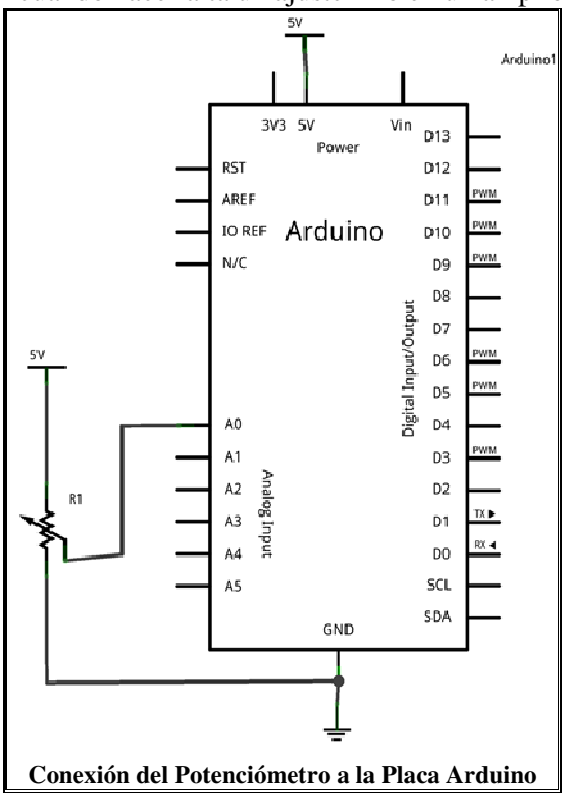


Diagrama de Conexión



Conexión del Potenciómetro a la Placa Arduino

Por si no queda claro como sebe conectarse el Potenciómetro a la placa Arduino, acá pueden visualizar apropiadamente, aunque no siempre será incluido este grafico en cada ejercicio.

FUNCIONES QUE UTILIZAREMOS EN NUESTRO PROGRAMA.

- **analogRead(pin)**: Esta función lee el valor desde el pin analógico especificado con una resolución de 10 bits. Esta función solo funciona en los pines analógicos (0-5). El valor resultante es un entero de 0 a 1023. Los pines analógicos, a diferencia de los digitales no necesitan declararse previamente como INPUT o OUTPUT.
- **map(ValorA_Transformar, ROrigenI, ROrigenF, RDestinoI, RDestinoF)**: La Función "**map( )**" trasforma un valor comprendido en un Rango o Intervalo Origen en otro numero equivalente pero que pertenece a otro un Rango o intervalo Destino. Por ejemplo, el numero 50, que pertenece al intervalo 1-100, a que numero equivaldría dentro

del rango 1-1000 ? : Si hacemos los cálculos necesario, encontraremos que el numero 50 es equivalente al 500 en el rango destino.

ValTransformado = map(ValorA_Transformar, ROrigenI, ROrigenF, RDestinoI, RDestinoF)

UTILIZANDO EL POTENCIÓMETRO CON ARDUINO

En la placa Arduino, tenemos Varios pines analógicos, desde A0 en adelante, y la cantidad dependerá del modelo de placa, y su uso común, es la lectura de datos de dispositivos analógicos, como es el caso del potenciómetro.
Tienen una resolución de 10 bits lo que implica que tenemos 1024 valores diferentes, es decir, podemos leer un rango de tensiones desde 0V hasta 5V detectando cambios de voltaje de 0.004V (5/1024). Por lo que los valores que obtendremos irán desde 0 hasta 1023.

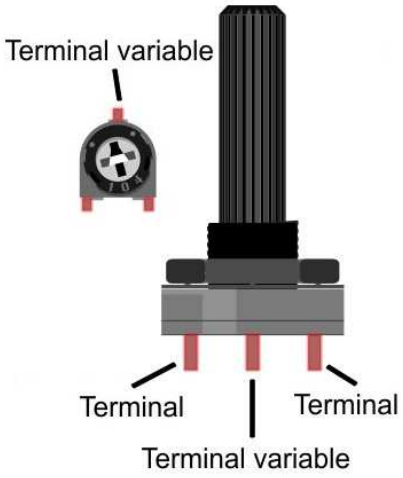
Y como la mejor manera de entender algo son los ejemplos, empezamos con uno que mediante el monitor serie podremos ir viendo que valores vamos obteniendo en un pin analógico según vayamos modificando la posición del potenciómetro.

1- Reconocimiento de un Potenciómetro - Lectura de Valores Arrojados.

Lecturas de los valores entregados por un potenciómetro y conversión a valor porcentual mediante la función "**map( )**". Visualización de Resultados en el monitor del Puerto Serie.

(Programa "003_Potenciometro_01")

```
1   int PinAnalogico = A0; // Conectaremos el Potenciómetro
-
2   double ValorOriginal,
3          ValorConvertido;
-
4   void setup( ) {
5     Serial.begin(9600);
6     pinMode( PinAnalogico, INPUT_PULLUP );
7   }
8   void loop( ) {
9     ValorOriginal = analogRead(PinAnalogico);
-
10  //   ValorConvertido = (ValorOriginal/1023) *100;
11    ValorConvertido = map( ValorOriginal, 0, 1023, 0, 100);
-
12    Serial.print("V. Original: ");
13    Serial.print(ValorOriginal);
14    Serial.print(" - V. Convertido: ");
15    Serial.println(ValorConvertido);
16    delay(500);
17  // Acá hacer lo que quiera, con el valor de posición medido
18  }
```

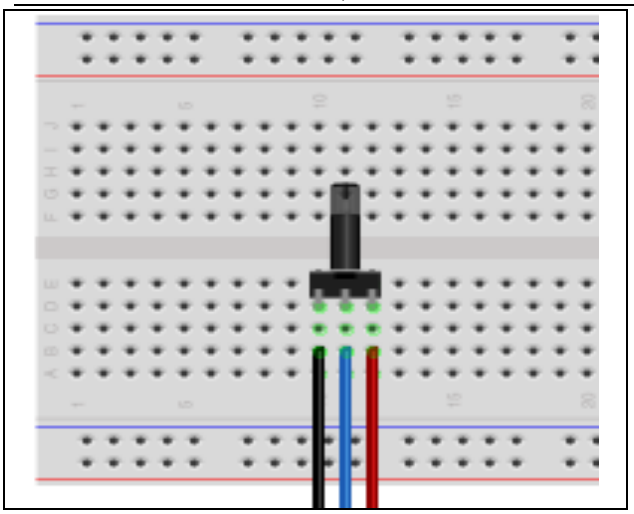


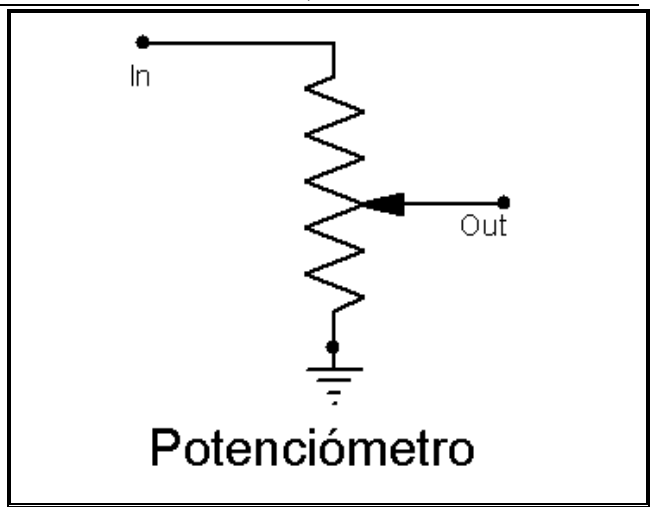
Terminal Variable representa la Entrada en Arduino, generalmente una entrada Analógica. Las Terminales (Izquierda o derecha) corresponden indistintamente a 5V y GND.

10 y 11	Estas dos líneas hacen exactamente lo mismo. Elegí dejar la 11, pero el resultado seria el mismo se dejara la línea 10.

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Potenciómetro – 3 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.





Los cables deberán ser conectactados: Cable Negro a GND, Rojo a 5V, y el cable Azul al Pin Analógico

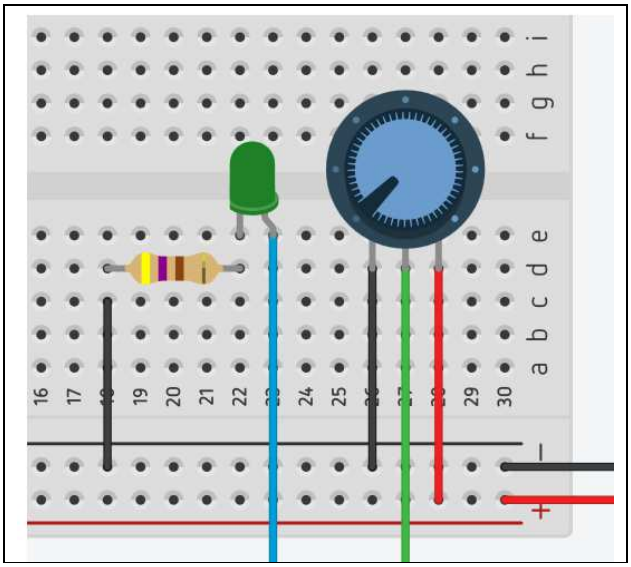
2- Desde un Potenciómetro, aumentar o disminuir la intensidad de un led. Tomar los valores entregados por un potenciómetro y usarlos para aumentar o disminuir la intensidad de un led. Recordar que los valores del potenciómetro oscilan entre 0 y 1023, y los soportados por un LED oscilan entre 0 y 255. Visualización de Resultados en el monitor del Puerto Serie.

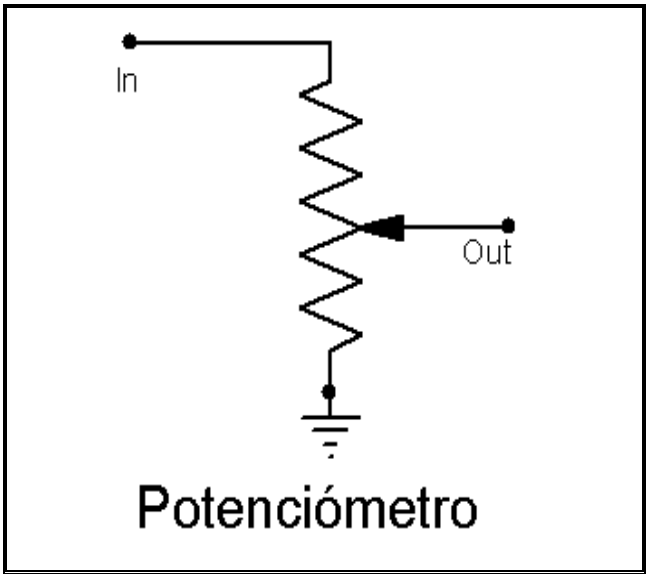
(Programa "003_Potenciometro_02_Intencidad_led")

```
1  int PinAnalogico = A0;  // Pin donde leeremos los valores del Potenciómetro
2  int PinLed       = 4;   // Pin donde conectaremos el LED
-
3  double ValorOriginal,
4         ValorConvertido;
-
5  void setup( ) {
6     Serial.begin(9600);
7     pinMode( PinAnalogico, INPUT_PULLUP );
8     pinMode( PinLed, OUTPUT );
9  }
-
10 void loop( ) {
11    ValorOriginal = analogRead(PinAnalogico); // Realiza lectura analógica
-
12 //  ValorConvertido = (ValorOriginal/1023) *255;          // convertir a porcentaje
13    ValorConvertido = map( ValorOriginal, 0, 1023, 0, 255); // convertir a porcentaje
-
14    analogWrite(PinLed, ValorConvertido);
15    VerPuertoSerie( );
16 }
-
17 void VerPuertoSerie(void){  //Envía al Monitor Puerto Serie, Valor Leído y Convertido
18    Serial.print("V Original: ");
19    Serial.print(ValorOriginal);
20    Serial.print(" - V Convertido: ");
21    Serial.println(ValorConvertido);
22    delay(500);
23 }
```

CIRCUITO PARA NUESTRO PROYECTO

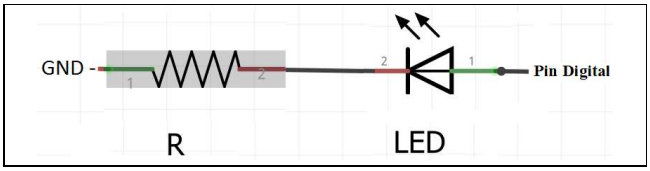
Lista de Materiales: 1 Led, 1 Resistencia de 470Ω (Para el Led), 1 Potenciómetro – 7 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.





Conexión de cables

DERECHA, de arriba hasta abajo: Cable Negro a GND. Cable Rojo a 5V. **Abajo, de izquierda a derecha:** Cable Azul al pin digital destinado al LED que se haya configurado en el programa y cable Verde, Al pin Analógico destinado al Potenciometro.



Cuando termine con este ejercicio, no desarme el circuito, lo usaremos en el próximo proyecto.

3- Controlar la Frecuencia de parpadeo de un LED con un potenciómetro.

Con un potenciómetro controlar el tiempo de espera en prender y apagar. Recordar que los valores del potenciómetro oscilan entre 0 y 1023, y los soportados por un LED oscilan entre 0 y 255. Visualización de Resultados en el monitor del Puerto Serie.

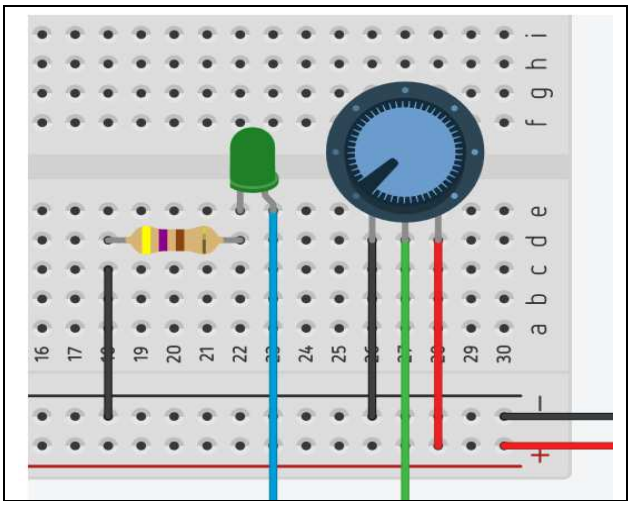
(Programa "003_Potenciometro_03_Frecuencia_Parpadeo_Led")

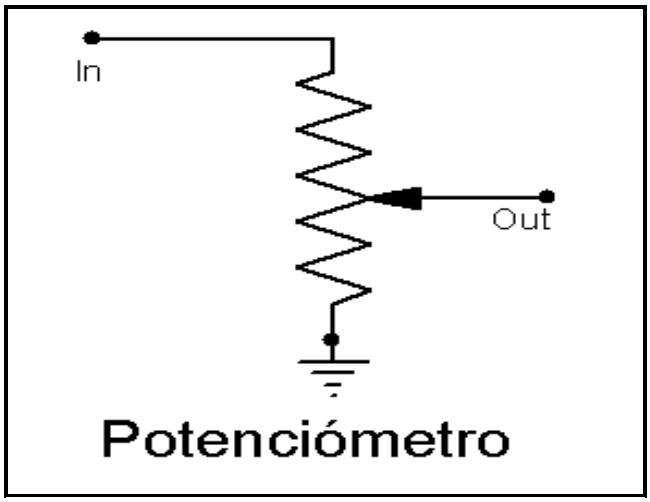
```
1   int PinAnalogico = A0;   // Pin potenciómetro
2   int PinLed       = 4;    // Pin para el LED
-
3   double TiempoInicial,
4          TiempoEspera;
5   int    EstadoLed;
-
6   void setup( ) {
7     pinMode( PinAnalogico, INPUT_PULLUP );
8     pinMode(PinLed, OUTPUT);
9     TiempoInicial = millis( );
10    EstadoLed = HIGH;
11  }
12  void loop( ) {
13    TiempoEspera = analogRead(PinAnalogico);  // Leo Potenciómetro
14    if(millis( ) < TiempoInicial + TiempoEspera ){
15      digitalWrite(PinLed, EstadoLed);
16    }else{
```

```
17      EstadoLed = 1 - EstadoLed;   // Invierto estado Actual del LED (Prendido / Apagado)
18      TiempoInicial = millis( );
19    }
20  }
```

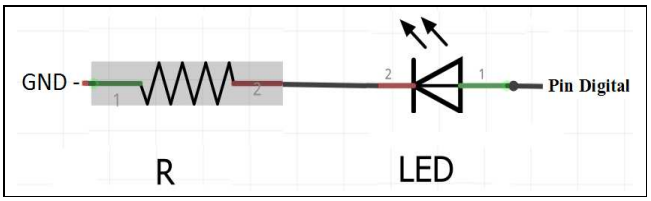
CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Led, 1 Resistencia de 470Ω (Para el Led), 1 Potenciómetro – 7 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.





Conexión de cables
DERECHA, de arriba hasta abajo: Cable Negro a GND. Cable Rojo a 5V. **Abajo, de izquierda a derecha:** Cable Azul al pin digital destinado al LED que se haya configurado en el programa y cable Verde, Al pin Analógico destinado al Potenciometro.



4- Medir Potencia que arroja el Potenciómetro, en forma visual (Vúmetro usando 6 Leds).

(Un vúmetro es un indicador de volumen. El VU es la unidad de volumen)
Medir Potencia liberada por un potenciómetro, con una escala de 6 LEDs Recordar que los valores del potenciómetro oscilan entre 0 y 1023. Incluir Visualización de Resultados en el monitor del Puerto Serie.
Ahora, según el potenciómetro entregue valores distintos, podremos encender un LED u otro. Esto lo vamos a programar a través de varios "**if**" anidados.

a. Desde 0 en adelante se encenderá LED 1
b. Desde 170 en adelante se encenderá LED 2
c. Desde 340 en adelante se encenderá LED 3
d. Desde 510 en adelante se encenderá LED 4
e. Desde 680 en adelante se encenderá LED 5
f. Desde 850 en adelante se encenderá LED 6

(Programa "003_Potenciometro_04_Intensidad_6_Led")

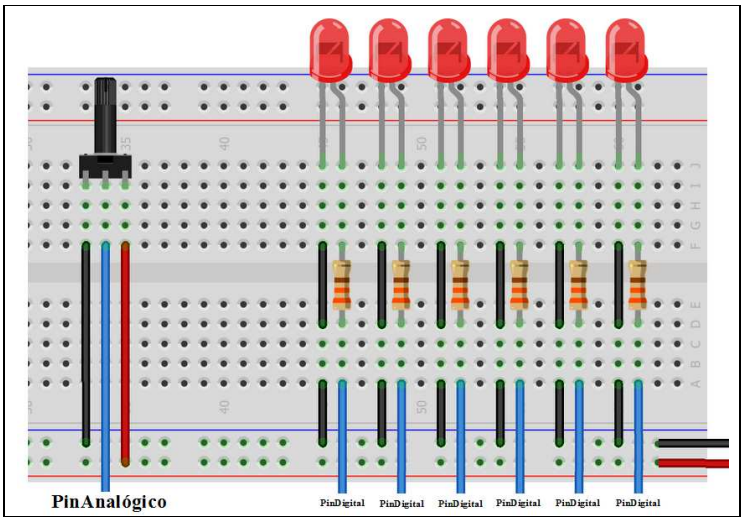
```
1  int PinPotenciometro = A0;
2  int PinLed_01 = 12;
3  int PinLed_02 = 10;
4  int PinLed_03 = 8;
5  int PinLed_04 = 6;
6  int PinLed_05 = 4;
```

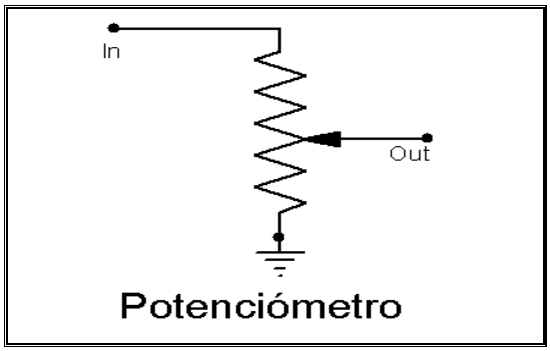
```
7   int PinLed_06 = 2;
-
8   void setup( ){
9     Serial.begin(9600);
10    pinMode( PinPotenciometro, INPUT_PULLUP );
11    pinMode(PinLed_01, OUTPUT);
12    pinMode(PinLed_02, OUTPUT);
13    pinMode(PinLed_03, OUTPUT);
14    pinMode(PinLed_04, OUTPUT);
15    pinMode(PinLed_05, OUTPUT);
16    pinMode(PinLed_06, OUTPUT);
17  }
-
18  void loop( ){
19    int valorPotenciometro = analogRead(PinPotenciometro);
-
20    if(valorPotenciometro > 0){
21      digitalWrite(PinLed_01, HIGH);
22    }else{
23      digitalWrite(PinLed_01, LOW);
24    }
25    if(valorPotenciometro > 170){
26      digitalWrite(PinLed_02, HIGH);
27    }else{
28      digitalWrite(PinLed_02, LOW);
29    }
30    if(valorPotenciometro > 340){
31      digitalWrite(PinLed_03, HIGH);
32    }else{
33      digitalWrite(PinLed_03, LOW);
34    }
35    if(valorPotenciometro > 510){
36      digitalWrite(PinLed_04, HIGH);
37    }else{
38      digitalWrite(PinLed_04, LOW);
39    }
40    if(valorPotenciometro > 680){
41      digitalWrite(PinLed_05, HIGH);
42    }else{
43      digitalWrite(PinLed_05, LOW);
44    }
45    if(valorPotenciometro > 850){
46      digitalWrite(PinLed_06, HIGH);
47    }else{
48      digitalWrite(PinLed_06, LOW);
49    }
50    PorMonitorSerie(valorPotenciometro);
51  }
-
52  void PorMonitorSerie(int Valor){
53  // Esta Función puede eliminarse, solo está para visualizar y comparar lo que se ve
54  // al Prender/Apagar sucesivamente los LEDs.
55    Serial.print("Valor Potenciómetro: ");
56    Serial.println(Valor);
```

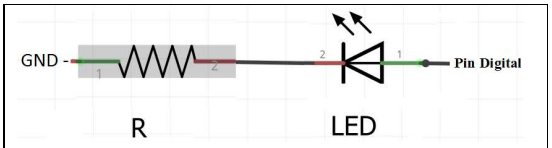
```
   delay(300);
}
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 6 Leds, 6 Resistencia de 470Ω (Para Leds), 1 Potenciómetro – 17 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.







Repetir este circuito, para cada uno de los LEDs. Y todos conectar al mismo GND, tal como se muestra en la imagen del circuito

Conexión de cables

Abajo, de Izquierda a Derecha: Primer Cable Azul: PIN Analógico configurado a tal efecto, los siguientes cables azules, se conectan a los pines digitales configurados para leds en el programa. **A la Derecha:** Salen dos cables, el Negro conectar a **GND**, el Rojo a **5V**.

5- Leer Dos Potenciómetros Simultáneamente y Mostrar ambos Valores por el Puerto Serie.

La idea de este ejercicio es mostrar como leer dos sensores o instrumentos de control simultáneamente y procesarlos separadamente (haciendo dos tareas distintas).

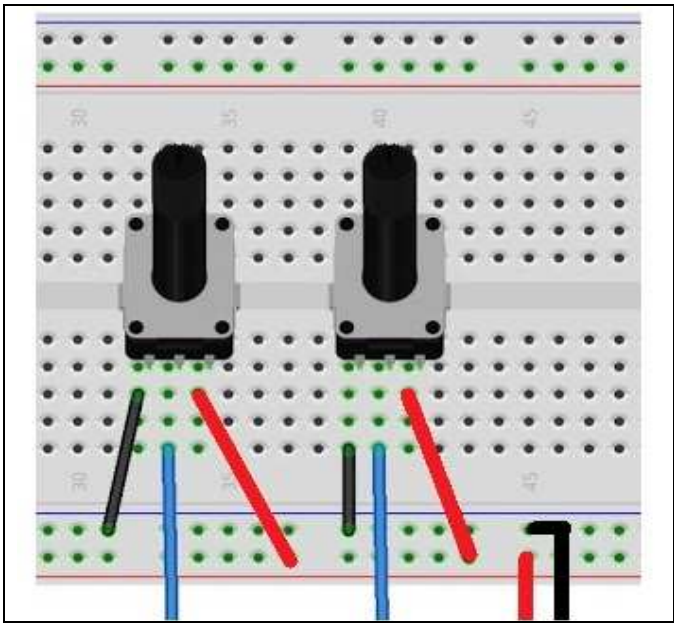
(Programa "003_Potenciometro_11_Conecta__dos_Potenciometros")

```
1   const int PinPot_01 = A0;
2   const int PinPot_02 = A1;
3   int LectutaPot_01 = 0;
4   int LectutaPot_02 = 0;
-
5   void setup( ) {
6     Serial.begin(9600);
7     pinMode(PinPot_01, INPUT);
8     pinMode(PinPot_02, INPUT);
9   }
10  void loop( ) {
11    LectutaPot_01 = analogRead(PinPot_01);
12    LectutaPot_02 = analogRead(PinPot_02);
-
13  //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
14  // Acciones comandadas Potenciometro 01
15  //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
16  // Acciones comandadas Potenciometro 02
17  //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
-
18    Serial.print(LectutaPot_01);
19    Serial.print(",");
20    Serial.println(LectutaPot_02);
21    delay(20);
22  }
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 2 Potenciómetros – 8 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.



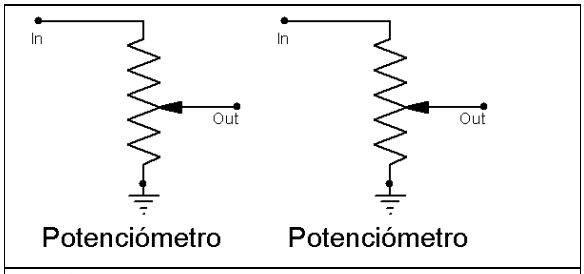


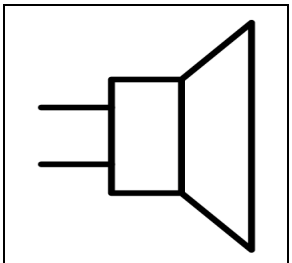
Diagrama de Conexión – GND Común a Todos.

<u>Conexión de Cables:</u> Abajo, de Izquierda a Derecha: Cables Azules, a los pines analógicos configurados para cada Potenciómetro. Cable Rojo a 5V. Cable Negro conectar a GND.

Se siente Capaz de Controlar un LED (la intensidad) Con el Potenciómetro 1 y Generar sonido con el potenciómetro 2? Las dos cosas simultáneamente.

EL PARLANTE, BUZZER o ZUMBADOR con ARDUINO

Un zumbador (Comúnmente nombrado como Speaker) es un elemento parecido a un altavoz pero sólo emite zumbidos (típico sonido que emiten los electrodomésticos). Son muy económicos, y en nuestros proyectos podemos reemplazarlos por un parlantito viejo, de algún electrodoméstico en desuso, incluso uno de esos parlante para PC que se nos ha roto.



Simbología Usada

Un zumbador (Comúnmente nombrado como Speaker) es un elemento parecido a un altavoz pero sólo emite zumbidos (típico sonido que emiten los electrodomésticos). Son muy económicos, y en nuestros proyectos podemos reemplazarlos por un parlantito viejo, de algún electrodoméstico en desuso, incluso uno de esos parlante para PC que se nos ha roto. Y cual seria la diferencia entre un Parlante (Speaker) y el Zumbador? **<u>Veamos:</u>**

Zumbadores o Buzzer: Si variamos la frecuencia a penas varía el tono o no suena. La causa de esto es que, un zumbador tiene poca capacidad para reproducir sonidos fielmente, pero es de un tamaño adecuado y es fabricado para este fin. Hay que tener en cuenta que la frecuencia influye tan poco que incluso sonaría conectándolo entre los pines 5V y GND. Para frecuencias demasiado altas el zumbador no responderá.

Parlante o Mini Speakers: Acá siempre podremos hacer lo mismo que con los Zumbadores, pero además también podremos reproducir "Musiquita" y son más resistentes, pero tendremos más trabajo al momento de adaptarlos a la carcaza de nuestro proyecto, y deberemos improvisar la conexión a nuestra placa Arduino.

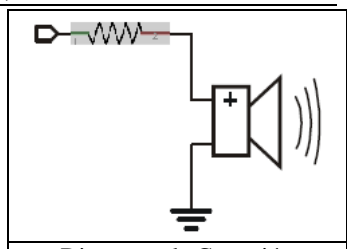
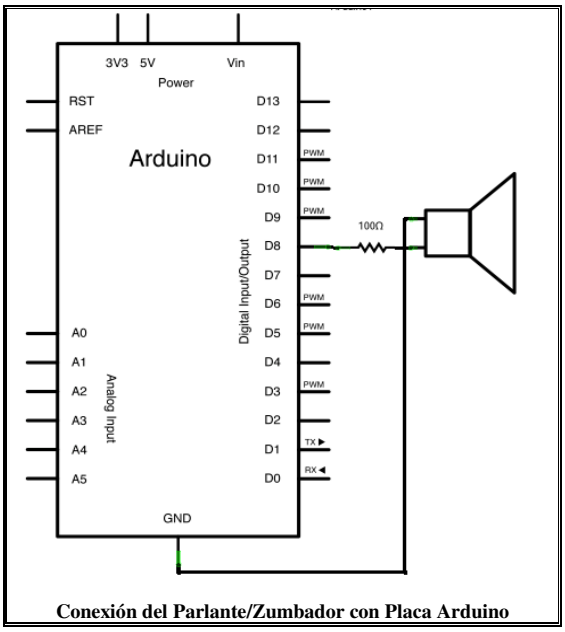


Diagrama de Conexión

La Conexión, para cualquiera de las dos opciones (Speakers o Zumbadores), es muy simple, El Cable rojo o comúnmente llamado positivo, se conecta al PIN Configurado desde el programa como "OUTPUT" y el cable negro o Negativo a GND. Por seguridad en los Zumbadores, se suele colocar una resistencia (Se visualiza en el diagrama de Conexión), Que puede oscilar entre 200 a 470 Ω (ohmios). En los mini parlantes reciclados, más resistentes, no es necesaria la resistencia.



Conexión del Parlante/Zumbador con Placa Arduino

Por si no queda claro como sebe conectarse el Parlante/Zumbador a la placa Arduino, acá pueden visualizar apropiadamente, aunque no siempre será incluido este grafico en cada ejercicio.

FUNCIONES QUE UTILIZAREMOS EN NUESTRO PROGRAMA.

Hay dos formas de emitir sonidos desde nuestra Arduino.

A- Usando las funciones "**tone( )**" y "**noTone( )**".
- tone( )
- noTone( )

B- Con la función "**analogWrite( )**".

A - GENERACION DE SONIDO USANDO LAS FUNCIONES "tone( )" y "noTone( )"

EJECUTAR UN TONO: Esta Función, en sus dos variantes, genera en un Pin una onda cuadrada de la frecuencia especificada.
- tone(Pin, frecuencia)
- tone(Pin, frecuencia, duración)

DETENER EL SONIDO:
- npTone(Pin)

DESCRIPCIÓN DE LOS PARÁMETROS:
Pin: el pin en el que se quiere generar el tono
frecuencia: la frecuencia del tono en hertzios – El tipo del dato debe ser "**int**" sin signo
duración: la duración del tono en milisegundos (opcional) – El tipo del dato debe ser "**long**" sin signo

IMPORTANTE: Sólo un tono puede ser generado a la vez. Si un tono ya se está reproduciendo en un Pin diferente, la llamada a "**tono( )**" no tendrá ningún efecto.
Si desea reproducir tonos diferentes en varios pines, es necesario llamar "**noTone( )**" en el Pin que se este ejecutando antes de llamar a "**tone ( )**" en el siguiente Pin.

B - GENERACION DE SONIDO USANDO LAS FUNCIONES "analogWrite( )"

Esta función se usara de la misma forma que la usamos para prender gradualmente un LED, pero conectando los cables del Speaker/Zumbador/Buzzer al Pin configurado y GND de la placa Arduino, tal como lo hiciéramos con el LED.

```
analogWrite(pin, valor); // escribe 'valor' en el 'Pin' definido como analógico.
```

Para Mostrar ambas formas de generar sonido, desarrollaremos dos programas, lo más similares posibles, y solo se los diferenciara, por las funciones que usaremos para emitir el sonido.

Para los próximos ejemplos, usaremos el mismo circuito, así que no lo desarme.

6- Emitir Beep-Beep por el Speaker/Zumbador - (Versión A) - Usando "delay( )"

Este programa emite Beep-Beep Intermitente por el Speaker/Zumbador conectada a la placa Arduino. Veremos a continuación, las dos formas de hacerlo, primero usando las Funciones "**tone( )**" y "**noTone( )**". Luego con la ya conocida instrucción "**analogWrite( )**". Si Bien a primera vista parecen lo mismo, luego, a la hora de hacer "música", se establece una diferencia marcada entre los métodos.

(Programa "004_Parlante_01a")

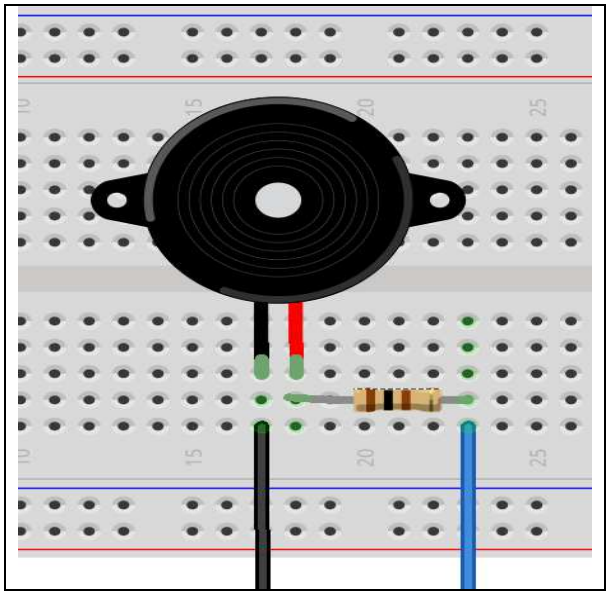
```
1  const int SpeakerPin = 10;
2  double TiempoEsperaSpeaker = 500;
3  int Frecuencia =1500;
-
4  void setup( ) {
5    pinMode(SpeakerPin, OUTPUT);
6  }
7  void loop( ) {
8    tone(SpeakerPin, Frecuencia);
9    delay(TiempoEsperaSpeaker);
10   noTone(SpeakerPin); //Libero SpeakerPin
11   delay(TiempoEsperaSpeaker);
12 }
```

(Programa "004_Parlante_01b")

```
int speakerOut = 10;
double TiempoEsperaSpeaker = 500;
int Volumen = 500;

void setup( ){
  pinMode(speakerOut, OUTPUT);
}
void loop( ) {
   analogWrite(speakerOut, Volumen);
   delay(TiempoEsperaSpeaker);
   analogWrite(speakerOut, 0);
   delay(TiempoEsperaSpeaker);
}
```

Estos dos programas, hacen que parezca que ambas funciones hacen lo mismo, pero no se engañe, existe todo un estudio y librerías con sonidos pre concebidos para las funciones "tone( )" y "noTone( )". Ya lo descubriremos más adelante en próximos ejercicios.

CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 Resistencia de 470Ω, 2 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

-o-

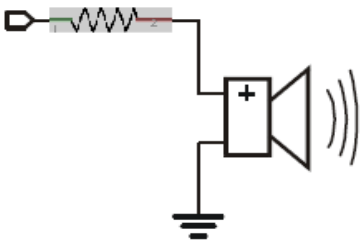


Diagrama de Conexión

Conexión de cables

Abajo, de Izquierda a Derecha: Cable Negro conectar a GND, y cable Azul, conectar al PIN configurado en el programa a través de la constante "SpeakerPin".

7- Emitir Beep-Beep por el Speaker/Zumbador- (Versión B) – Programación Multitarea, sin usar delay( ).

Estos programas, al igual que los dos anteriores, emiten Beep-Beep Intermitente por el Speaker/Zumbador conectada a la placa Arduino, pero sin usar la función "delay( )". Primero usando las Funciones "**tone( )**" y "**noTone( )**". Luego con la ya conocida instrucción "**analogWrite( )**".

(Programa "004_Parlante_03a")

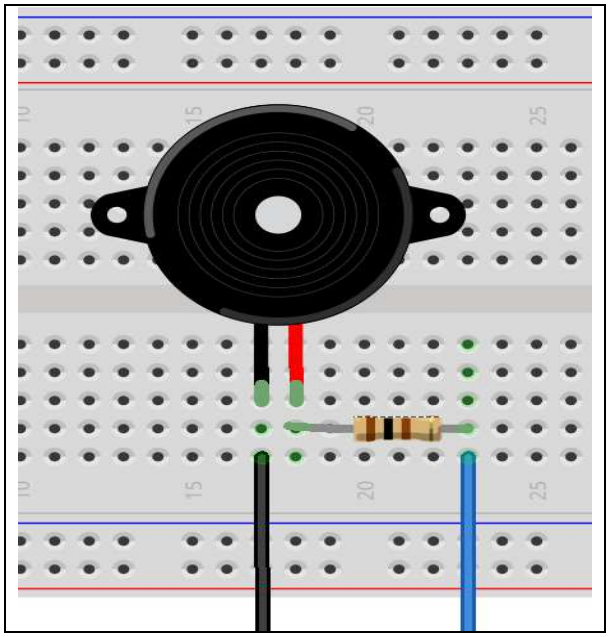
```
const int SpeakerPin = 10;

double TiempoInicialSpeaker,
       TiempoEsperaSpeaker;
int    EstadoBeepSpeaker,
       Frecuencia;

void setup( ) {
  pinMode(SpeakerPin, OUTPUT);
  TiempoInicialSpeaker = millis( );
  TiempoEsperaSpeaker = 500;
  EstadoBeepSpeaker = 0;
  Frecuencia =1500;
}

void HaceBeep(void){

  if(millis( ) < TiempoInicialSpeaker +
                 TiempoEsperaSpeaker ){

    if(EstadoBeepSpeaker == 1){
      tone(SpeakerPin, Frecuencia);
    }else{
      noTone(SpeakerPin); // Libero SpeakerPin
    }
  }else{
    EstadoBeepSpeaker = 1 - EstadoBeepSpeaker;
    TiempoInicialSpeaker = millis( );
  }

}

void loop( ) {
  HaceBeep( );
  // Acá otras Tareas
  // Acá otras Tareas
}
```

(Programa "004_Parlante_03b")

```
const int SpeakerPin = 10;

double TiempoInicialSpeaker,
       TiempoEsperaSpeaker;
int    EstadoBeepSpeaker,
       Volumen;

void setup( ) {
  pinMode(SpeakerPin,OUTPUT);
  TiempoInicialSpeaker = millis( );
  TiempoEsperaSpeaker = 500;
  EstadoBeepSpeaker = 0;
  Volumen = 300;
}

void HaceBeep(void){

  if(millis( ) < TiempoInicialSpeaker +
                 TiempoEsperaSpeaker ){

    if(EstadoBeepSpeaker == 1){
      analogWrite(SpeakerPin,Volumen);
    }else{
      analogWrite(SpeakerPin, 0);
    }
  }else{
    EstadoBeepSpeaker = 1 - EstadoBeepSpeaker;
    TiempoInicialSpeaker = millis( );
  }

}

void loop( ){
  HaceBeep( );
  // Acá otras Tareas
  // Acá otras Tareas
}
```

Estos dos programas, hacen parecer que ambas funciones hacen lo mismo, pero no se engañe, existe todo un estudio y librerías con sonidos pre concebidos para las funciones "tone( )" y "noTone( )". Ya lo descubriremos más adelante en próximos ejercicios.

CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 Resistencia de 470Ω, 2 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

-o-

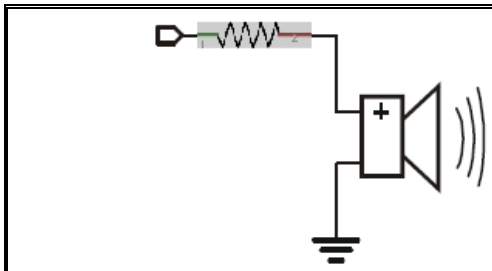


Diagrama de Conexión

Conexión de cables

Abajo, de Izquierda a Derecha: Cable Negro conectar a GND, y cable Azul, conectar al PIN configurado en el programa a través de la constante "SpeakerPin".

8- Timbre. Activar Timbre cuando Botón Pulsador es Accionado.

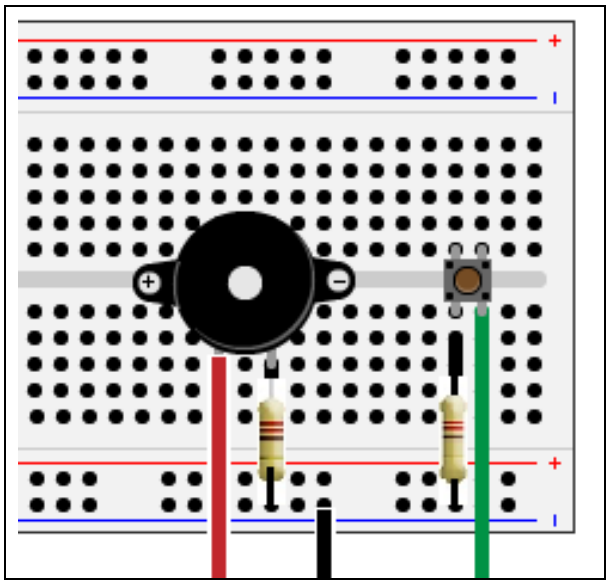
En este ejemplo, Simulamos el Timbre común que tenemos en casa. Dicho en otras palabras, **cuando el botón es presionado, y mientras este presionado**, suena el timbre (entonces en este caso, no hace falta controlar el Rebote).

(Programa "004_Parlante_07_Boton_Acciona_Timbre_01")

```
1   const int PinSpeaker = 10,
2           PinBoton   =  5;
3   const int Frecuencia = 500;
-
4   void setup( ) {
5     Serial.begin(9600);
6     pinMode(PinSpeaker, OUTPUT);
7     pinMode(PinBoton, INPUT_PULLUP);
8   }
-
10  void loop( ) {
11    if (digitalRead( PinBoton ) == LOW){
12      tone(PinSpeaker, Frecuencia);  // Sonido Activado
13      Serial.println("------> ON");
14    }else{
15      noTone(PinSpeaker);  // Sonido Desactivado
16      Serial.println("------> OFF");
17    }
    }
```

La lógica del programa es muy simple. En cuanto detecto que el botón ha sido pulsado, activo el sonido, luego al detectar que el botón ya no es presionado, simplemente se desactiva el sonido.

CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 Resistencia de 470Ω (Speaker), 1 Resistencia de 4K7 Ω (Botón), 3 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

-o-

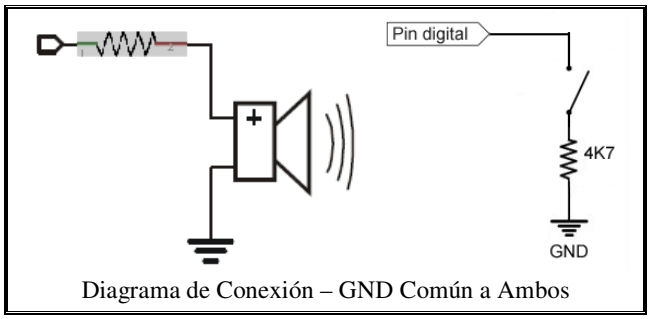


Diagrama de Conexión – GND Común a Ambos

Conexión de Cables: Abajo, de Izquierda a Derecha: Cable Rojo al Pin configurado para el Speaker en el programa, Negro conectar a GND, y cable Verde, conectar al PIN configurado en el programa para el Botón Pulsador. Esta vez, no es necesario cuidar del rebote o velocidad de lectura con el Botón, ya que este ejercicio es muy simple, suena timbre mientras se Active Boton, y deja de sonar cuando ya no se activa.

9- Encender y Apagar un Timbre por medio de 3 botones (en forma indistinta). Con Programación Multitareas. El programa permite prender o Apagar un Timbre por medio de 3 botones (prende desde cualquiera y apaga desde cualquiera de ellos).

(Programa "004_Parlante_08_Boton_Timbre_Prende_Apaga_Con_3_Botones")

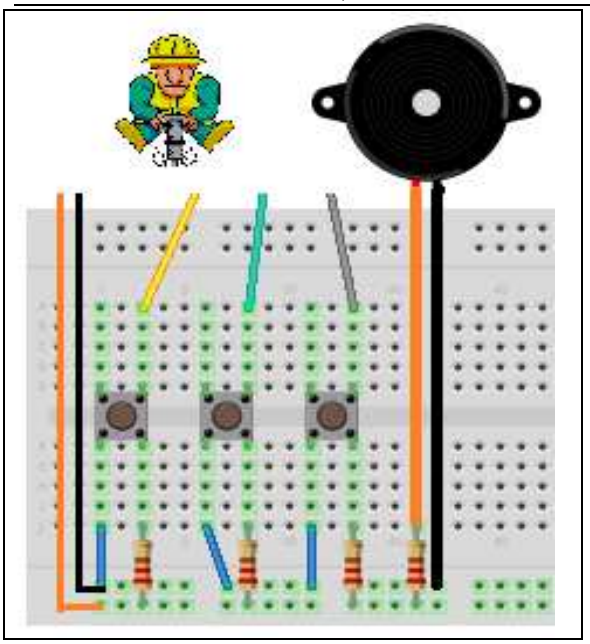
```
1   const int PinBoton_01 = 3,
2             PinBoton_02 = 5,
3             PinBoton_03 = 7,
4             SpeakerPin  = 10;
-
5   int Boton_01_Valor,  // Variable botón 01
6     Boton_02_Valor,  // Variable botón 02
7     Boton_03_Valor,  // Variable botón 03
8     EstadoSpeaker,   // Estado del LED (Prendido o Apagado)
9     Presionado_01,   // Para saber si Este Botón Fue Presionado
10    Presionado_02,   // Para saber si Este Botón Fue Presionado
11    Presionado_03,   // Para saber si Este Botón Fue Presionado
12    Frecuencia;
-
13  void setup( ){
14    Serial.begin (9600);
15    pinMode(PinBoton_01, INPUT_PULLUP);
16    pinMode(PinBoton_02, INPUT_PULLUP);
17    pinMode(PinBoton_03, INPUT_PULLUP);
18    pinMode(SpeakerPin, OUTPUT);
19    Presionado_01 = 0;
20    Presionado_02 = 0;
21    Presionado_03 = 0;
22    EstadoSpeaker  = 0;
23    Frecuencia     = 1500;
```

```
24  }
 -
25  void loop ( ){
26    Boton_01_Valor = digitalRead(PinBoton_01); // Leemos PinBoton_01.
27    Boton_02_Valor = digitalRead(PinBoton_02); // Leemos PinBoton_02.
28    Boton_03_Valor = digitalRead(PinBoton_03); // Leemos PinBoton_03.
 -
29    if(Boton_01_Valor == LOW){
30      Presionado_01 = 1;     //Activo Botón 01
31    }
 -
32    if(Boton_02_Valor == LOW){
33      Presionado_02 = 1;     //Activo Botón 02
34    }
 -
35    if(Boton_03_Valor == LOW){
36      Presionado_03 = 1;     //Activo Botón 03
37    }
 -
38    Boton_01_Valor = digitalRead(PinBoton_01); // Leemos PinBoton_01.
39    Boton_02_Valor = digitalRead(PinBoton_02); // Leemos PinBoton_02.
40    Boton_03_Valor = digitalRead(PinBoton_03); // Leemos PinBoton_03.
 -
41    if( (Boton_01_Valor == HIGH && Presionado_01 == 1 ||
42                  Boton_02_Valor == HIGH && Presionado_02 == 1 ||
43                        Boton_03_Valor == HIGH && Presionado_03 == 1)){
 -
44      Serial.println("---> Presionó Botón");
45      Presionado_01 = 0;    //La variable vuelve a su valor original
46      Presionado_02 = 0;    //La variable vuelve a su valor original
47      Presionado_03 = 0;    //La variable vuelve a su valor original
 -
48      if(EstadoSpeaker == 0){
49        Serial.println("---------> Activo Sonido");
50        tone(SpeakerPin, Frecuencia);
51      }else{
52        Serial.println("---------> Desactivo Sonido");
53        noTone(SpeakerPin);
54      }
 -
55      EstadoSpeaker = 1 - EstadoSpeaker;
56    }
57    // Acá Realizo Otras Tareas
58    // Acá Realizo Otras Tareas
59  }  // Final de la Funcion loop
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 3 Botones Pulsador de 2 Patas – 1 Speaker - 3 Resistencia de 4K7 Ω (Para el Botón) – 1 Resistencia de 470Ω (Para el Speaker) – 8 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

IMPORTANTE: Notar que los tres botones, tienen una conexión del tipo "**Pull_Up**" (5V-Resistencia-PindeControl-Boton-GND).



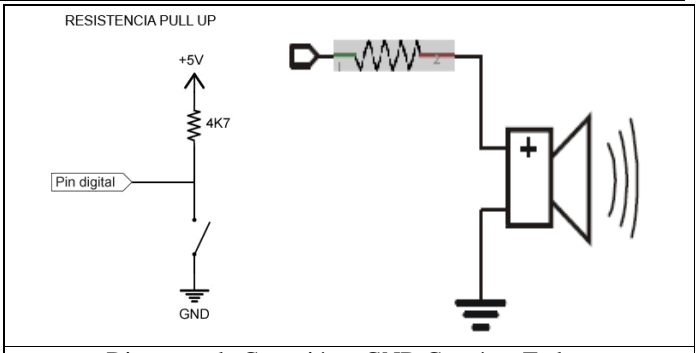


Diagrama de Conexión – GND Común a Todos.

Repetir la Conexión para los tres botones igual.

Conexión de Cables: Arriba, de Izquierda a Derecha: Cable Rojo a 5V. Cable Negro conectar a GND. Cables Amarillo, Verde y Gris, conectar a los PINes respectivos, configurados en el programa para los Botones Pulsador. El estado del Sonido cambia al soltar el botón que se esta presionando.

10- Generando sonido con un potenciómetro.

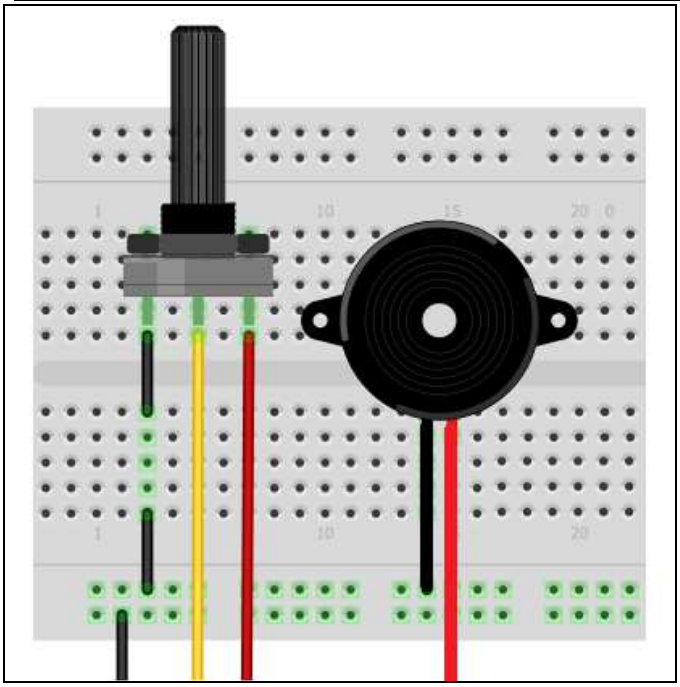
La idea de este ejercicio es generar diversos tonos en el Parlantre/Buzzer a partir del estado analógico del potenciómetro. Esto nos ayuda a comprender un poco mas el funcionamiento.

(Programa "004_Parlante_09_Potenciometro_Genera_Sonido")

```
1   const int PinSpeaker = 5,
2             PinPotenciometro = A0;
-
3   int ValorLeido,
4       Frecuencia,
5       Duracion;
-
6   void setup( ){
7     Serial.begin (9600);
8     pinMode(PinSpeaker, OUTPUT);
9     pinMode(PinPotenciometro,  INPUT);
10    Duracion = 250;
11  }
-
12  void loop( ){
13    ValorLeido = analogRead(PinPotenciometro);
14    Serial.println(ValorLeido);
15    Frecuencia = map(ValorLeido, 0, 1023, 0, 5000);
16    tone(PinSpeaker, Frecuencia, Duracion);
17    delay(100);
18  }
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Potenciómetro – 1 Speaker – 1 Resistencia de 470Ω (Para el Speaker) – 6 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.



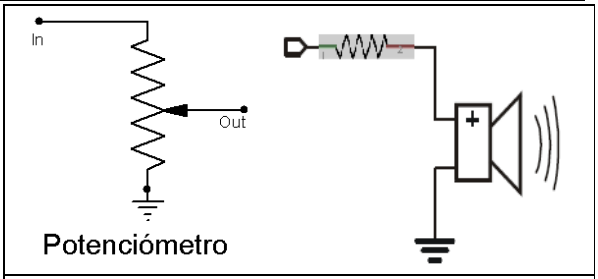


Diagrama de Conexión – GND Común a Todos.

En este Proyecto he omitido la resistencia del Parlante, porque he utilizado un parlante de radio reciclado (Son más resistentes). Para el caso de que uses un Buzzer, puedes poner una resistencia igual que proyectos anteriores.

Conexión de Cables: Abajo, de Izquierda a Derecha: Cable Negro conectar a GND. Cable Marrón Conectar a 5V. Cable Amarillo Al pin Configurado para el Potenciómetro, y el Rojo, al pin configurado para el Parlante.

11- Mini Piano Digital. Con Programación Multitareas.

El programa permite componer melodías con ocho notas: ‘c’, ‘d’, ‘e’, ‘f’, ‘g’, ‘a’, ‘b’ y ‘C’. El presente es un muy simple ejemplo creación de música. Luego les dejare algunos proyectos con melodías concretas, que servirán seguramente de inspiración.

(Programa “004_Parlante_10_Piano_01”)

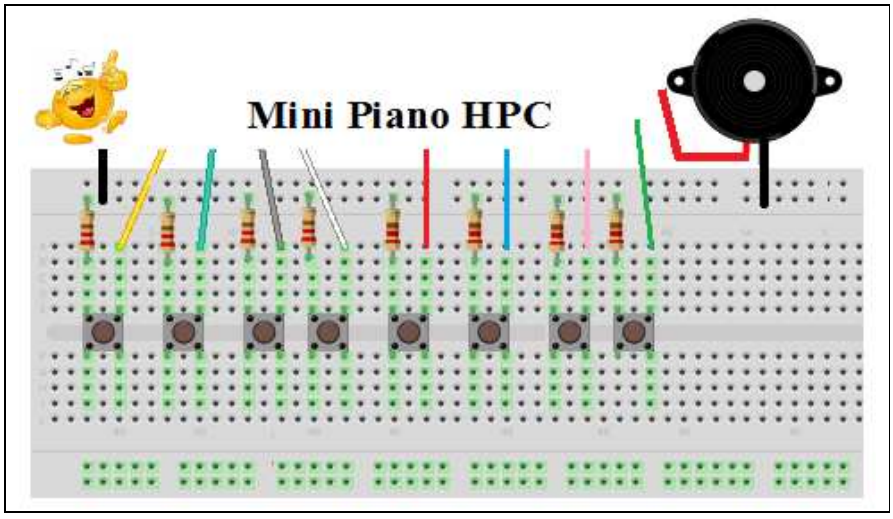
```
1   const int Boton_C = 2,
2             Boton_D = 3,
3             Boton_E = 4,
4             Boton_F = 5,
5             Boton_G = 6,
6             Boton_A = 7,
7             Boton_B = 8,
8             Boton_Cup = 9;
9   const int PinSpeaker = 10;
10  int EstadoBoton_C = 0,
11      EstadoBoton_D = 0,
12      EstadoBoton_E = 0,
13      EstadoBoton_F = 0,
14      EstadoBoton_G = 0,
15      EstadoBoton_A = 0,
16      EstadoBoton_B = 0,
17      EstadoBoton_Cup = 0;
-
18  //NOTAS          'c' ,   'd' ,   'e' ,   'f' ,   'g' ,   'a' ,   'b' ,   'C'
19  int Tono[ ] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 }; //frecuencias
20  // Posiciones      0       1       2       3       4       5       6       7
-
21  int TonoSeleccionado = 0;
-
22  void setup( ){
23    Serial.begin(9600);
24    pinMode(Boton_C, INPUT_PULLUP );
25    pinMode(Boton_D, INPUT_PULLUP );
```

```
26    pinMode(Boton_E, INPUT_PULLUP );
27    pinMode(Boton_F, INPUT_PULLUP );
28    pinMode(Boton_G, INPUT_PULLUP );
29    pinMode(Boton_A, INPUT_PULLUP );
30    pinMode(Boton_B, INPUT_PULLUP );
31    pinMode(Boton_Cup, INPUT_PULLUP );
32    pinMode(PinSpeaker, OUTPUT);
33   }
34   void loop( ){
35    EstadoBoton_C = digitalRead(Boton_C);
36    EstadoBoton_D = digitalRead(Boton_D);
37    EstadoBoton_E = digitalRead(Boton_E);
38    EstadoBoton_F = digitalRead(Boton_F);
39    EstadoBoton_G = digitalRead(Boton_G);
40    EstadoBoton_A = digitalRead(Boton_A);
41    EstadoBoton_B = digitalRead(Boton_B);
42    EstadoBoton_Cup = digitalRead(Boton_Cup);
 -
43    if((EstadoBoton_C == LOW) || (EstadoBoton_E == LOW) ||
44          (EstadoBoton_G == LOW) || (EstadoBoton_D == LOW) ||
45             (EstadoBoton_F == LOW) || (EstadoBoton_A == LOW) ||
46                (EstadoBoton_B == LOW) || (EstadoBoton_Cup == LOW) ){
 -
47      if (EstadoBoton_C == LOW) {
48       TonoSeleccionado = Tono[0];
49      }
50      if (EstadoBoton_E == LOW){
51       TonoSeleccionado = Tono[1];
52      }
53      if (EstadoBoton_G == LOW){
54       TonoSeleccionado = Tono[2];
55      }
56      if (EstadoBoton_D == LOW){
57       TonoSeleccionado = Tono[3];
58      }
59      if (EstadoBoton_F == LOW){
60       TonoSeleccionado = Tono[4];
61      }
62      if (EstadoBoton_A == LOW){
63       TonoSeleccionado = Tono[5];
64      }
65      if (EstadoBoton_B == LOW){
66       TonoSeleccionado = Tono[6];
67      }
68      if (EstadoBoton_Cup == LOW){
69       TonoSeleccionado = Tono[7];
70      }
71      tone(PinSpeaker,TonoSeleccionado);
72      Serial.println("Si");
73     }else{ // Cuando el Botn No sea Precionado, Apago Speaker
74      noTone(PinSpeaker);
75      Serial.println("No Boton");
76     }
77   }
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Speaker – 8 Resistencia de 4K7 Ω (Para el Botón) – 1 Resistencia de 470Ω (Para el Speaker)– 10 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

IMPORTANTE: Notar que todos botones, tienen una conexión del tipo "**Pull_Up**" (5V-Resistencia-PindeControl-Boton-GND).



Conexión de Cables: Arriba, de Izquierda a Derecha: Cable Negro conectar a GND. Cables Amarillo, Verde Claro, gris, Blanco, Rojo, Celeste, Rosa, Verde, se deben conectar a los pines configurados en el programa, destinados a cada Botón. Finalmente el rojo del Parlante, al pin designado para ello en el programa, y el negro del parlante, al GND común del circuito.
En este Proyecto se ha omitido la resistencia del Parlante, porque he utilizado uno de radio reciclado (Son más resistentes). Para el caso de que uses un Buzzer, puedes poner una resistencia igual que proyectos anteriores.
Para este caso, Todos los botones Pulsadores, pueden ser conectados como se muestra, ignorando el efecto rebote, ya que el tono debe sonar mientras este presionado, sin importar las veces que sea leído.

Repetir esta conexión para los 8 botones, y recordar conectar todos a un GND Común.

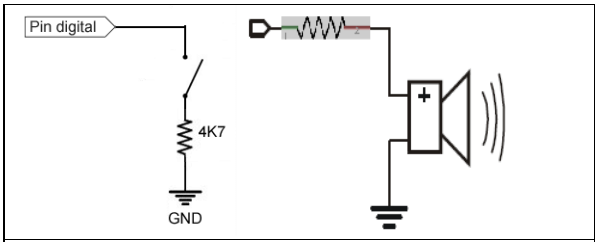


Diagrama de Conexión – GND Común a Todos.

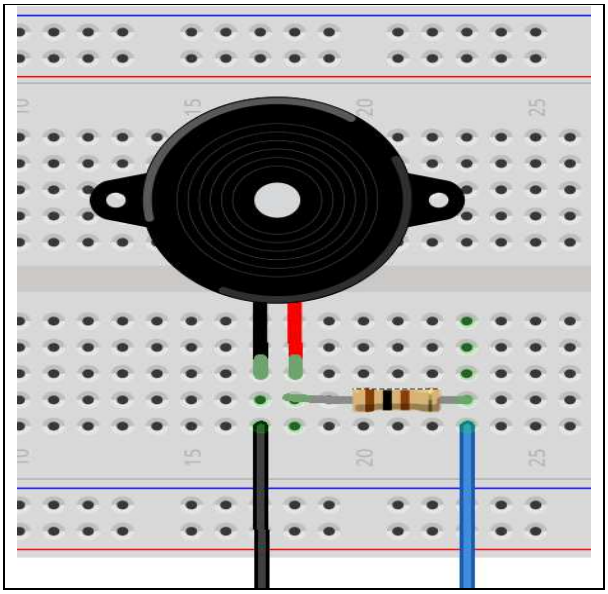
12- Recorrido de octava. – Idea para los que entienden de Música.

Almacenaremos sólo el valor de la frecuencia inicial, y las sucesivas notas tendrán la frecuencia de la anterior multiplicada por 1,059. Usamos Programación Multitarea.
(Programa "004_Parlante_11__Recorrido_de_Octava_01")

```
1  /*****************************************/
2  /*              Recorrido de Octava              */
3  /*****************************************/
4  #define SpeakerPin 12
5  #define TiempoPrendido  900
6  #define TiempoApagado   300
7  #define Repeticiones     12
8  long TiempoInicial,
9       TiempoEspera,
10      Contador,              // variable para el contador
11      Frecuencia = 220;   // frecuencia correspondiente a la nota La
12 int  EstadoSpeaker;
13 float m = 1.059;        // constante para multiplicar frecuencias
```

```
14  void setup( ) {
15    pinMode(SpeakerPin, OUTPUT);
16    TiempoInicial = millis( );
17    TiempoEspera = TiempoPrendido;
18    EstadoSpeaker = HIGH;
19    Contador = 0;
20    Frecuencia = 220;
21  }
22  void loop( ) {
23    if( millis( ) > TiempoInicial + TiempoEspera ){
24      if(EstadoSpeaker == HIGH){
25        TiempoEspera = TiempoApagado;
26      }else{
27        TiempoEspera = TiempoPrendido;
28        Frecuencia = Frecuencia * m;     // actualiza la frecuencia
29        Contador = Contador + 1;
30      }
31      EstadoSpeaker = 1 - EstadoSpeaker;
32      TiempoInicial = millis( );
33    }
 -
34    if(EstadoSpeaker == HIGH){
35      tone(SpeakerPin,Frecuencia); // Emite el tono
36    }else{
37      noTone(SpeakerPin);          // Detiene el tono
38    }
 -
39    if(Contador >= Repeticiones){
40      Contador = 0;
41      Frecuencia = 220;
42-   }
43  } // Final Función loop
```

CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 Resistencia de 470Ω, 2 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

-o-

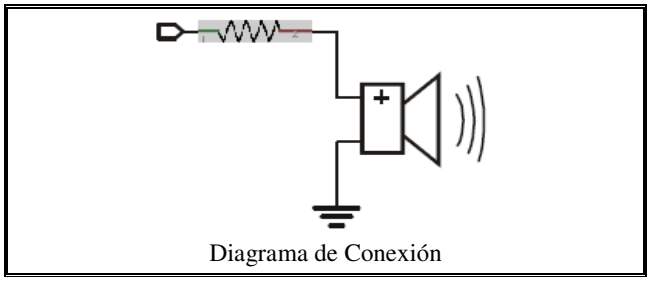


Diagrama de Conexión

Conexión de cables

Abajo, de Izquierda a Derecha: Cable Negro conectar a GND, y cable Azul, conectar al PIN configurado en el programa a través de la constante “SpeakerPin”.

01010010 01101111 01100010 11000011 10110011 01110100 01101001 01100011 01100001 00101100
00100000 01110101 01101110 01100001 00100000 01110110 01101001 01100100 01100001 00100000
01100001 01110000 01100001 01100011 01101001 01101111 01100001 01101110 01110100 01100101
00101110 00100001

Si tienes algunas Correcciones y/o Sugerencias, por favor contáctame.