# FOTO RESISTENCIAS LDR

Las LDR (Light-Dependent Resistor) son resistencias cuyo valor resistivo (Ω - Ohmios) varía en función de la luz que incide sobre ellas. Los valores típicos son de 1 MΩ en total oscuridad, a 50-100Ω bajo luz brillante.

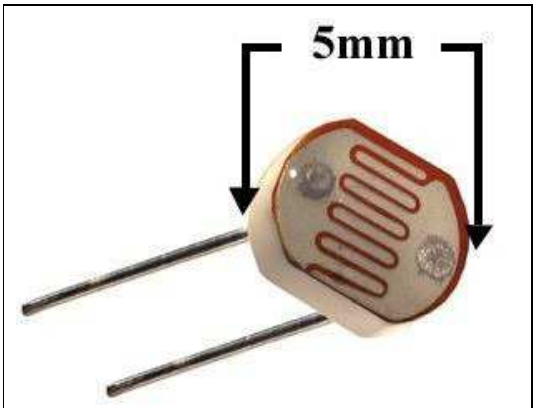
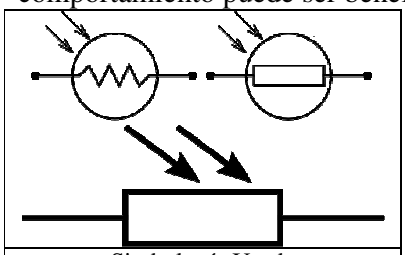


Foto Resistencias Ldr 5mm "Sensor De Luz"

Esto las hace especialmente útiles cuando deseas realizar un proyecto que depende de la luz ambiente. Sin embargo, esta no es la única utilidad que le puedes dar a tus LDR con Arduino. Las LDR no solo se ven afectadas por la luz visible. Su valor resistivo también varía con la luz infrarroja y ultravioleta.

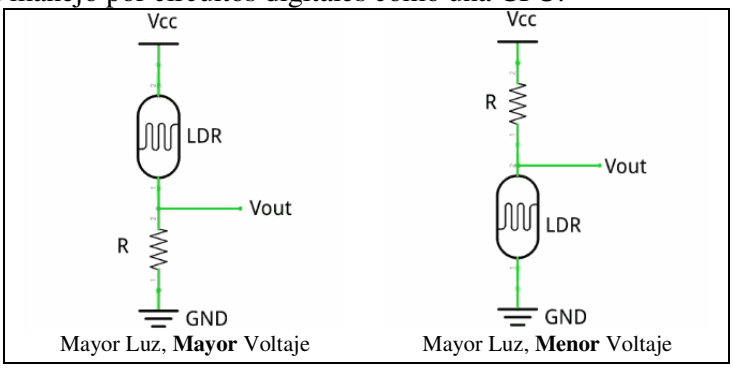
Por otro lado, la variación de la resistencia es relativamente lenta, de 20 a 100 ms en función del modelo. Esta lentitud hace que no sea posible registrar variaciones rápidas, como las producidas en fuentes de luz artificiales alimentadas por corriente alterna. Este comportamiento puede ser beneficioso, ya que dota al sensor de una gran estabilidad.


Simbología Usada
(Cualquiera de las tres será valida)

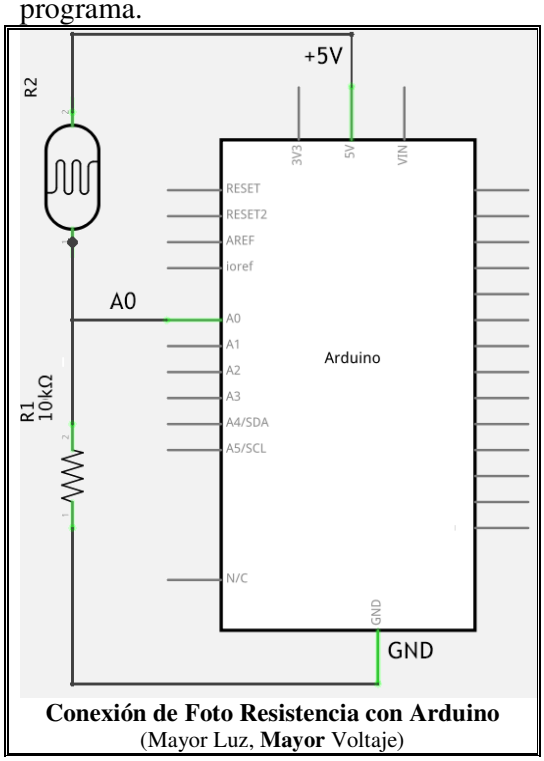
Las LDR están compuestas generalmente por sulfuro de cadmio (CdS). El cadmio que contienen reacciona ante la luz, dejando que sus electrones se muevan libremente, permitiendo el paso de la corriente.

**<u>Conversión Analógico-Digital (ADC):</u>** Es el proceso mediante el cual se convierte una magnitud física como un voltaje, corriente, temperatura, etc. en un número binario (o señal digital) con el propósito de facilitar su manejo por circuitos digitales como una CPU.

Arduino realiza este proceso para cuantificar la cantidad de luz percibida por el LDR y poder procesarla numéricamente con nuestro programa.

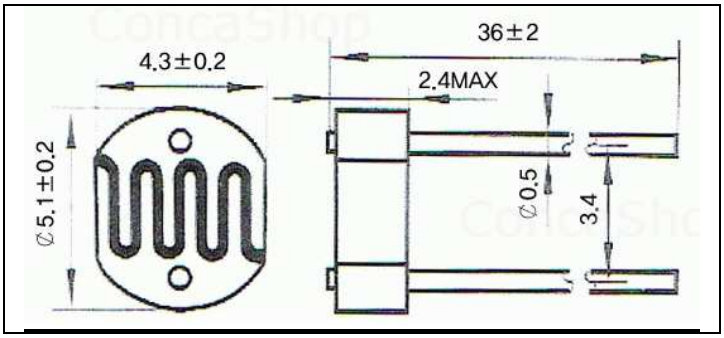


Mayor Luz, **Mayor** Voltaje — Mayor Luz, **Menor** Voltaje



**Conexión de Foto Resistencia con Arduino**
(Mayor Luz, **Mayor** Voltaje)

**<u>A Mayor Luz, Mayor Voltaje:</u>** (Conexión Alta) Si conectas tu LDR con la fuente de voltaje, cuanta más luz incida sobre tu Foto Resistencia, menor será la diferencia de potencial (caída de voltaje) que tendrás entre la fuente y tu Arduino, con lo que placa de Arduino leerá un valor mayor.



**A Mayor Luz, Menor Voltaje:** (Conexión Baja) En este caso Si conectas tu LDR con GND, cuanta más luz incida sobre tu Foto Resistencia, Arduino leerá un valor menor.

**Un LDR es un sensor que resulta adecuado para proporcionar medidas cuantitativas sobre el nivel de luz, tanto en interiores como en exteriores, y reaccionar, por ejemplo, encendiendo una luz, subiendo una persiana, u orientando un robot.**

# FOTORRESISTENCIAS LDR de 5mm CON ARDUINO

En la placa Arduino, tenemos Varios pines analógicos, desde A0 en adelante, la cantidad dependerá del modelo de placa, y su uso común, es la lectura de datos de dispositivos analógicos, como es el caso de una Fotorresistencia LDR.

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.

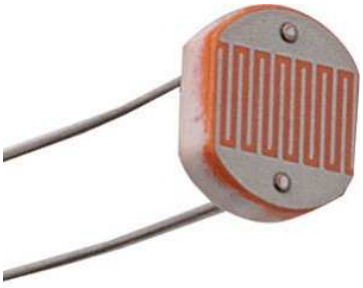
Y como la mejor manera de entender algo es mediante los ejemplos, empezamos con uno en donde podremos ir viendo que valores obtenemos (en un PIN analógico), según vayamos modificando la cantidad de luz que reciba nuestro LDR.

## 1- Reconocimiento de una Foto Resistencia LDR - Lectura de Valores Entregados.

Lecturas de los valores entregados por una Foto Resistencia LDR. Visualización de Resultados en el monitor del Puerto Serie.

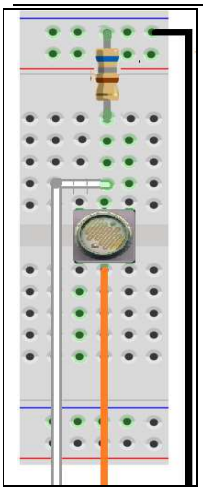
(Programa "006_Foto_Resistencia_Ldr_5_Mm_01")

```
1   #define PIN_LDR A0  // Pin Analógico
2   int ValorSensado;
-
3   void setup( ) {
4     Serial.begin(9600);
5     pinMode( PIN_LDR, INPUT );
6   }
-
7   void loop( ){
8     ValorSensado = analogRead(PIN_LDR);
9     Serial.print("Valor: ");
10    Serial.println(ValorSensado);
11    delay(500);
12  }
```



## CIRCUITO PARA NUESTRO PROYECTO

**Lista de Materiales:** 1 Foto Resistencia LDR – 3 Cables Macho/Macho – 1 resistencia de 10 KΩ 1/4W – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.



**Los cables deberán ser conectados:** Cable Negro a GND, Rojo a 5V, y el cable Blanco al Pin Analógico configurado en el programa.

**Es importante** destacar que la **Fotorresistencia** ha sido conectada, según Diagrama de Conexión: Mayor Luz, Mayor Voltaje, o también llamada Conexión Alta. Para Recordar puede memorizar la secuencia: 5V-LDRPIN-Resistencia_GND.

Tarea Para el alumno: es interesante probar la otra conexión, y comparar resultados, para evitar futuros errores en circuitos más complejos.

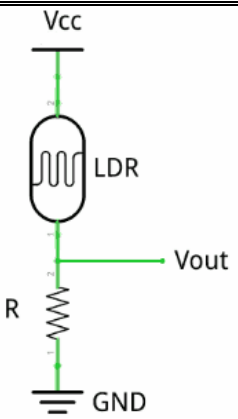


Diagrama de Conexión. Mayor Luz, Mayor Voltaje – Conexión Alta.

## FUNCIONES QUE UTILIZAREMOS EN PROXIMO PROGRAMA.

- max(x, y): Retorna el mayor entre x e y. Dicho de otra forma, retorna el MAYOR de los dos valores pasados como parámetros a la función.
- min(x, y): Retorna el menor entre x e y. Dicho de otra forma, retorna el MENOR de los dos valores pasados como parámetros a la función.

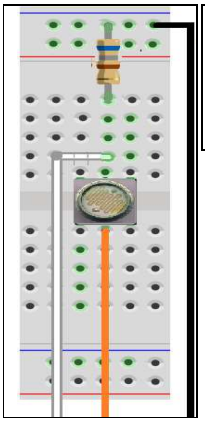
## 2- Busca e Informa la Mayor y Menor intensidad de luz Registrada por la Fotorresistencia LDR..

Buscar y visualizar la mayor y menor luminosidad registrada por la Fotorresistencia LDR a través del monitor del Puerto Serie.

(Programa "006_Foto_Resistencia_Ldr_5_Mm_02_Menor_Y_Mayor_Valor_Registrado")

```
1   #define PIN_LDR A0
-
2   int maximo  = 0,
3       minimo    = 1023,
4       ValorSensado;
-
5   void setup( ) {
6     Serial.begin(9600);
7     pinMode( PIN_LDR, INPUT_PULLUP );
8   }
-
9   void loop( ) {
10    ValorSensado = analogRead(PIN_LDR);
11    maximo = max(maximo,ValorSensado);
12    minimo = min(minimo,ValorSensado);
13    Serial.print(ValorSensado);
13    Serial.print(" , ");
15    Serial.print(maximo);
16    Serial.print(" , ");
17    Serial.println(minimo);
18    delay(500);
19  }
```

# CIRCUITO PARA NUESTRO PROYECTO



**Lista de Materiales:** 1 Foto Resistencia LDR – 3 Cables Macho/Macho – 1 resistencia de 10 KΩ 1/4W – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

**Los cables deberán ser conectados:** Cable Negro a GND, Rojo a 5V, y el cable Blanco al Pin Analógico configurado en el programa.

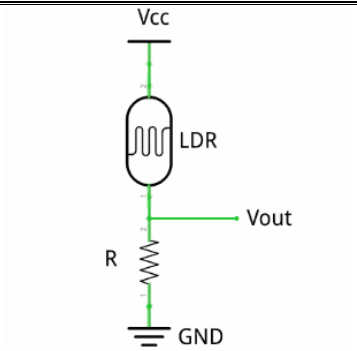


Diagrama de Conexión. Mayor Luz, Mayor Voltaje – Conexión Alta.

**Es importante** notar que la **Fotorresistencia** ha sido conectada, según Diagrama de Conexión. Mayor Luz, Mayor Voltaje – Conexión Alta. Tarea Para el alumno: es interesante probar la otra conexión, y comparar resultados, para evitar futuros errores en circuitos más complejos.

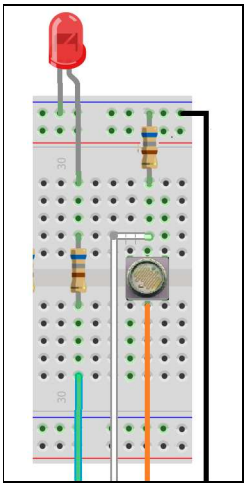
## 3- Prende luz nocturna automáticamente.

Al anochecer, disminuye la luz y este circuito, prendera una luz Nocturna, luego al amanecer, aumenta la luminosidad, y la luz se apagara automáticamente. La intensidad de la luz (luminosidad) se registrará con una Fotorresistencia LDR. Más adelante veremos como prender/apagar iluminación verdadera.

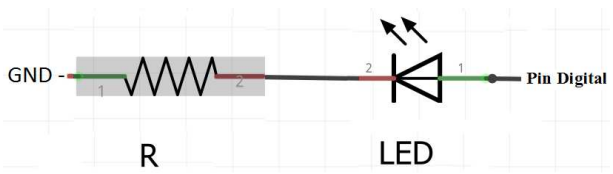
(Programa "006_Foto_Resistencia_Ldr_5_Mm_03_Luz_Nocturna")

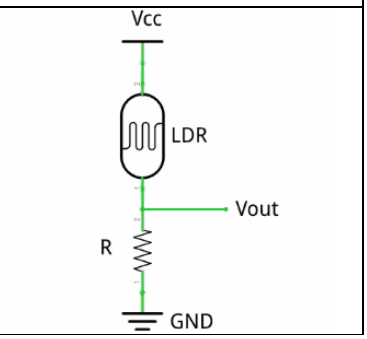
```
1   int PinLuzNocturna = 8;
-
2   // Pin analogico de entrada para el LDR
3   int PinLDR = A0;
4    // Variable donde se almacena el valor del LDR
5   int ValorLDR = 0;
-
6   void setup( ){
7     Serial.begin(9600);
8     pinMode(PinLDR, INPUT);
9     pinMode(PinLuzNocturna, OUTPUT);
10  }
-
11  void ManejaLed(int ValorSensado){
12    if(ValorSensado < 340){ // Si esta anocheciendo
13       digitalWrite(PinLuzNocturna, HIGH);
14       Serial.print("Luz Nocturna Prendida: ");
15    }else{
16       digitalWrite(PinLuzNocturna, LOW);
17       Serial.print("Luz Nocturna Apagada: ");
18    }
19    Serial.println(ValorLDR);
20  }
-
21  void loop( ){
22    ValorLDR= analogRead(PinLDR);
23    ManejaLed(ValorLDR);
-
24    delay(200);
25  }
```

# CIRCUITO PARA NUESTRO PROYECTO



**Lista de Materiales:** 1 Leds (como luz nocturno), 1 Resistencias de 470Ω (Para el Led), 1 Foto Resistencia LDR, 1 resistencia de 10 KΩ 1/4W (Para la Fotorresistencia LDR), – 4 Cables Macho/Macho — 1 Placa Protoboard - Placa Arduino y 1 Cable USB.




**Los cables deberán ser conectados:** Cable Negro a GND, Naranja a 5V, cable Blanco al Pin Analógico configurado en el programa y cable Verde al Pin destinado en el programa para LED (Luz Nocturna).
**Es importante** notar que la **Fotorresistencia** ha sido conectada, según Diagrama de Conexión. Mayor Luz, Mayor Voltaje – Conexión Alta. Tarea Para el alumno: es interesante probar la otra conexión, y comparar resultados, para evitar futuros errores en circuitos más complejos.

## 4- Prende luz nocturna automáticamente (Permite Regular que tan de noche se prenderá).

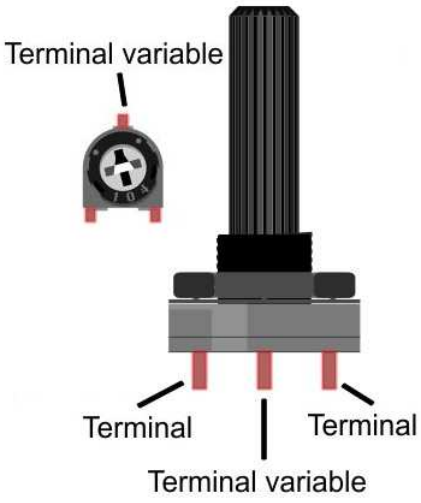
Al anochecer, disminuye la luz y este circuito, prendera una luz Nocturna (Permitiendo regular mínimo de luz natural para prender con un Potenciómetro), luego al amanecer, aumenta la luminosidad, y la luz se apagara automáticamente.
La intensidad de la luz (luminosidad) se registrará con una Fotorresistencia LDR. Más adelante veremos como prender/apagar iluminación verdadera.

(Programa "006_Foto_Resistencia_Ldr_5_Mm_03_Luz_Nocturna")

```
1   int PinLuzNocturna = 8;
-
2   int PinLDR = A0,
3       PinPotenciometro = A1;
-
4   double IntReg;
5   int ValorLDR = 0;
-
6   void setup( ){
7     Serial.begin(9600);
8     pinMode(PinLDR, INPUT); // Fotorresistencia
9     pinMode( PinPotenciometro, INPUT_PULLUP );
10    pinMode(PinLuzNocturna, OUTPUT); // Luz Nocturna
11    IntReg = 340;
12  }
13  double RegulaIntensidad( int PinPotenciometro){
14    double Minimo = analogRead(PinPotenciometro);
15    Serial.print("Intensidad Minima: ");
16    Serial.print(Minimo);
17    Serial.print(" - ");
18    return( Minimo);
19  }
```

## RECORDAR

**Conexión de un Potenciómetro**



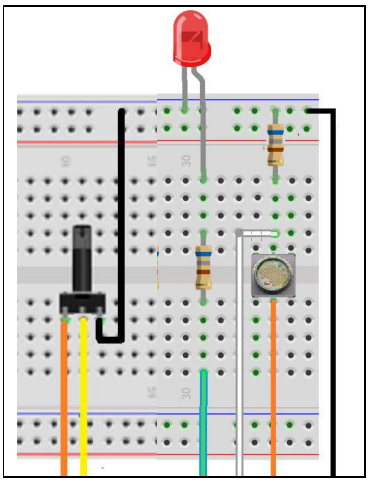

Terminal Variable representa la Entrada en Arduino, generalmente una entrada Analógica. Las Terminales (Izquierda o derecha) corresponden indistintamente a

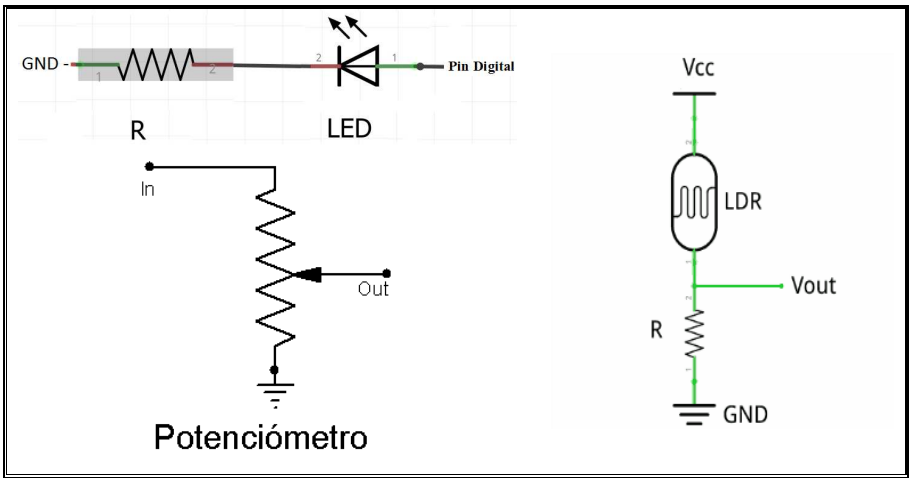
```
20  void ManejaLed(int ValorSensado, double Minimo){
21    if(ValorSensado < Minimo){ // Si esta anocheciendo
22      digitalWrite(PinLuzNocturna, HIGH);
23      Serial.print("Luz Nocturna Prendida: ");
24    }else{
25      digitalWrite(PinLuzNocturna, LOW);
26      Serial.print("Luz Nocturna Apagada: ");
27    }
28    Serial.println(ValorLDR);
29  }
30  void loop( ){
31    ValorLDR= analogRead(PinLDR);
32    IntReg = RegulaIntensidad(PinPotenciometro);
33    ManejaLed(ValorLDR, IntReg);
34    delay(200);
35  }
```

5V y GND.

## CIRCUITO PARA NUESTRO PROYECTO

**Lista de Materiales:** 1 Potenciómetro - 1 Leds (como luz nocturno), 1 Resistencias de 470Ω (Para el Led), 1 Foto Resistencia LDR, 1 resistencia de 10 KΩ 1/4W (Para la Fotorresistencia LDR), – 7 Cables Macho/Macho — 1 Placa Protoboard - Placa Arduino y 1 Cable USB.






**Los cables deberán ser conectados:** Cable **Negro** a GND, Los dos cables **Naranjas,** ambos a 5V, Cable **Amarillo** al pin Analógico destinado para el Potenciómetro, cable **Blanco** al Pin Analógico destinado para la Fotorresistencia en el programa y cable **Verde** al Pin destinado en el programa para LED (Luz Nocturna).

**Es importante** notar que la **Fotorresistencia** ha sido conectada, según Diagrama de Conexión. Mayor Luz, Mayor Voltaje – Conexión Alta. Tarea Para el alumno: es interesante probar la otra conexión, y comparar resultados, para evitar futuros errores en circuitos más complejos.

### 5- Informa la intensidad de luz Registrada por la Fotorresistencia LDR visualmente con 6 LEDs

Informar la luminosidad registrada por la Fotorresistencia LDR a través de 6 leds (Vúmetro Lumínico) y del monitor del Puerto Serie. (Un vúmetro no es más que un instrumento el cual nos indica el nivel de la señal que nos entrega el sensor, y nosotros lo que haremos será representarlo con unos leds para que sea más vistoso).

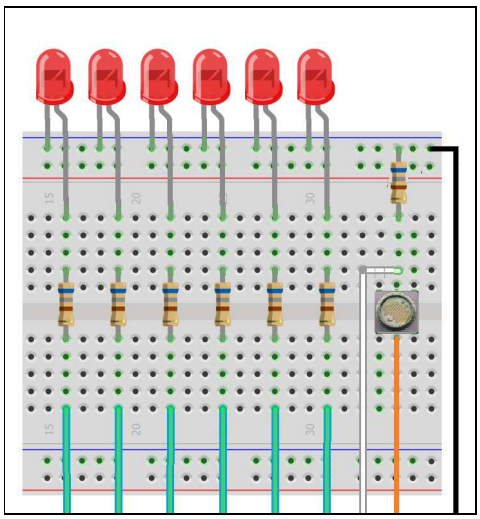
(Programa "006_Foto_Resistencia_Ldr_5_Mm_05_mide_Intensisdad_con_Led")

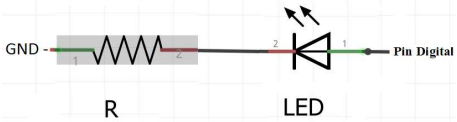
```
1   int PinLed1 = 4,
2       PinLed2 = 5,
3       PinLed3 = 6,
4       PinLed4 = 7,
5       PinLed5 = 8,
6       PinLed6 = 9;
-
7   int PinLDR = A0; // Pin analógico de entrada para el LDR
-
8   int ValorLDR = 0;  // Variable donde se almacena el valor del LDR
-
9    void setup( ){
10    Serial.begin(9600);
11    pinMode(PinLDR, INPUT);
12    pinMode(PinLed1, OUTPUT);
13    pinMode(PinLed2, OUTPUT);
14    pinMode(PinLed3, OUTPUT);
15    pinMode(PinLed4, OUTPUT);
16    pinMode(PinLed5, OUTPUT);
17    pinMode(PinLed6, OUTPUT);
18   }
-
19   void ManejaLed(int ValorSensado){
20    if(ValorSensado > 170){
21      digitalWrite(PinLed1, HIGH);
22    }else{
23      digitalWrite(PinLed1, LOW);
24    }
25    if(ValorSensado > 340){
26      digitalWrite(PinLed2, HIGH);
27    }else{
28      digitalWrite(PinLed2, LOW);
29    }
30    if(ValorSensado > 510){
31      digitalWrite(PinLed3, HIGH);
32    }else{
33      digitalWrite(PinLed3, LOW);
34    }
35    if(ValorSensado > 680){
35      digitalWrite(PinLed4, HIGH);
37    }else{
38      digitalWrite(PinLed4, LOW);
39    }
40    if(ValorSensado > 850){
41      digitalWrite(PinLed5, HIGH);
42    }else{
43      digitalWrite(PinLed5, LOW);
44    }
45    if(ValorSensado > 950){
46      digitalWrite(PinLed6, HIGH);
47    }else{
48      digitalWrite(PinLed6, LOW);
49    }
50   }
```

```
51  void loop( ){
52    // Guardamos el valor leído del ADC en una variable
53    // El valor leído por el ADC (voltaje) aumenta de manera directamente proporcional
54    // con respecto a la luz percibida por el LDR
55    ValorLDR= analogRead(PinLDR);
56     Serial.println(ValorLDR);
57    ManejaLed(ValorLDR);
58    delay(200);
59  }
```

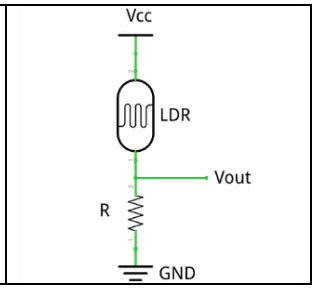
## CIRCUITO PARA NUESTRO PROYECTO

**Lista de Materiales:** 6 Leds, 6 Resistencias de 470Ω (Para el Led), 1 Foto Resistencia LDR, 1 resistencia de 10 KΩ 1/4W (Para la Fotorresistencia LDR), – 9 Cables Macho/Macho — 1 Placa Protoboard - Placa Arduino y 1 Cable USB.






**Repetir esta conexión para todos los LED y un GND común para todos**




**Los cables deberán ser conectados:** Cable Negro a GND, Naranja a 5V, cable Blanco al Pin Analógico configurado en el programa y los Verdes a cada uno de los pines destinados en el programa para LED.
**Es importante** notar que la **Fotorresistencia** ha sido conectada, según Diagrama de Conexión. Mayor Luz, Mayor Voltaje – Conexión Alta. Tarea Para el alumno: es interesante probar la otra conexión, y comparar resultados, para evitar futuros errores en circuitos más complejos.

## 6- Escuchemos cuanta Luz Detecta una Fotorresistencia LDR.

Generar sonido proporcionalmente a la luz registrada por el sensor. Paralelamente, accionar un LED, para que se prenda acompañando el sonido, siempre acorde a la cantidad de luz registrada.

(Programa "006_Foto_Resistencia_Ldr_5_Mm_06_Escuchemos_Cuanta_Luz")

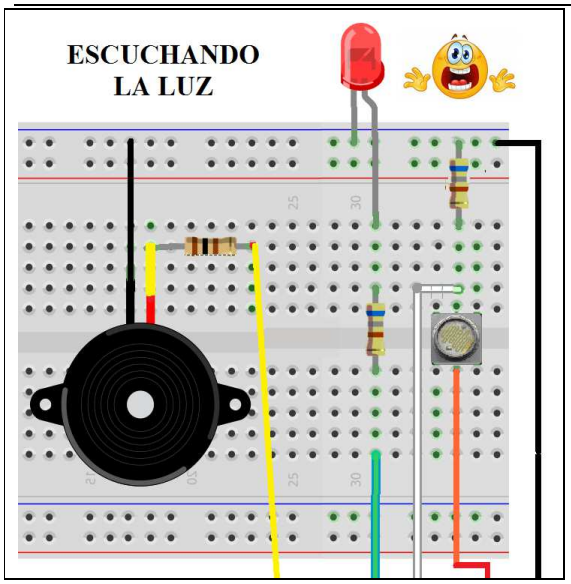
```
1   #define LdrPin    A0
2   #define SpeakerPin 4
3   #define LedPin     6
-
4   long CantidadLuz,
5        FrecuenciaSpeaker;
6   int  IntensidadLed;
7   long SpeakerFrecuenciaMaxima = 2500;
8   long LdrFrecuenciaMaxima = 1023;
-
9   void setup( ) {
10     Serial.begin(9600);
11     pinMode(LdrPin, INPUT);
```

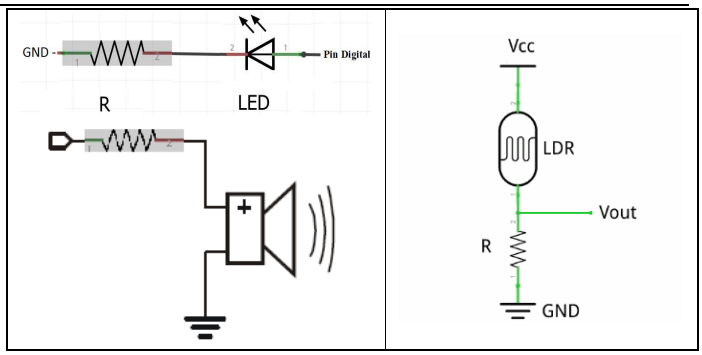
```
12      pinMode(SpeakerPin, OUTPUT);
13      pinMode(LedPin, OUTPUT);
14  }
-
15  void loop( ) {
16      CantidadLuz = analogRead(LdrPin); // Valores 0-1023
17  //    Serial.print("Luz Sensada: ");
18  //    Serial.print(CantidadLuz);
19  //    Serial.print(" - ");
-
20      FrecuenciaSpeaker = (CantidadLuz * SpeakerFrecuenciaMaxima) / LdrFrecuenciaMaxima;
21      IntensidadLed     = (int) ((CantidadLuz * 255) / LdrFrecuenciaMaxima);
-
22      Speaker(SpeakerPin, FrecuenciaSpeaker, 10 );
23      Led(LedPin, IntensidadLed);
24  }
-
25  void Led(int PinLed, int IntLed) {
26      analogWrite(PinLed, IntLed);
27  //    Serial.print(" - Led: ");
28  //    Serial.println(IntLed);
29  }
-
30  void Speaker(int targetPin, long FrecSpr, long length) {
31      long delayValue = 1000000/FrecSpr/2;
32      long numCycles = FrecSpr * length/ 1000;
-
33      for (long i=0; i < numCycles; i++){
34          digitalWrite(targetPin,HIGH);
35          delayMicroseconds(delayValue);
35          digitalWrite(targetPin,LOW);
37          delayMicroseconds(delayValue);
38      }
-
39  //    Serial.println(" - Speaker: ");
40  //    Serial.print(FrecSpr);
41  }
```

El texto informativo que se envía por el monitor se encuentra comentariado. Solo hay que activarlas y podrá leer los valores censados y calculados por Arduino. Notara que todo funciona un poco más lento, esto se debe al tiempo que se pierde enviado los datos.

## CIRCUITO PARA NUESTRO PROYECTO

**Lista de Materiales:** 1 Led, 1 Resistencias de 470Ω (Para el Led), 1 Foto Resistencia LDR, 1 resistencia de 10 KΩ 1/4W (Para la Fotorresistencia LDR), – 1 Speaker (o parlante reciclado), 1 resistencia de 470Ω (Para speaker), 5 Cables Macho/Macho – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.




**Los cables deberán ser conectados:** Amarillo, el Pin destinado para Speaker, cable Verde al PIN destinado para el LED, cable Blanco al PIN Analógico configurado en el programa para la Fotorresistencia LDR, cable Rojo a 5V y finalmente cable Negro a GND.
**Es importante** notar que la **Fotorresistencia** ha sido conectada, según Diagrama de Conexión. Mayor Luz, Mayor Voltaje – Conexión Alta.

# Sensor de Distancia Ultrasónico HC-SR04

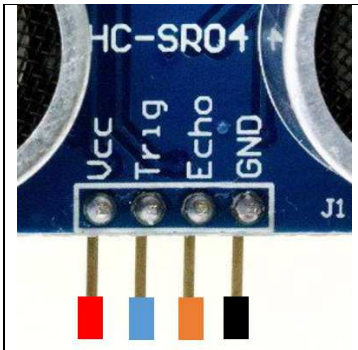
El HC-SR04 es un sensor mide el tiempo desde que se envía hasta que retorna al sensor el pulso ultrasónico.

Este sensor es capaz de detectar objetos y calcular el tiempo de retorno de la señal de distancia que se encuentra en un rango de 2 a 450 cm.
Su uso es tan sencillo como enviar el pulso y medir el tiempo que tarda en regresar. De muy pequeño tamaño, el HC-SR04 se destaca por su bajo consumo, gran precisión y bajo precio.


**Sensor de Distancia de Ultrasonido HC-SR04**

**CARACTERÍSTICAS TECNICAS**
Dimensiones del circuito: 43 x 20 x 17 mm
Tensión de alimentación: 5 Vcc
Frecuencia de trabajo: 40 KHz
Rango máximo: 4.5 m
Rango mínimo: 1.7 cm
Duración mínima del pulso de disparo (nivel TTL): 10 µS.
Duración del pulso eco de salida (nivel TTL): 100-25000 µS.
Tiempo mínimo de espera entre una medida y el inicio de otra 20 mS.

Pines de conexión

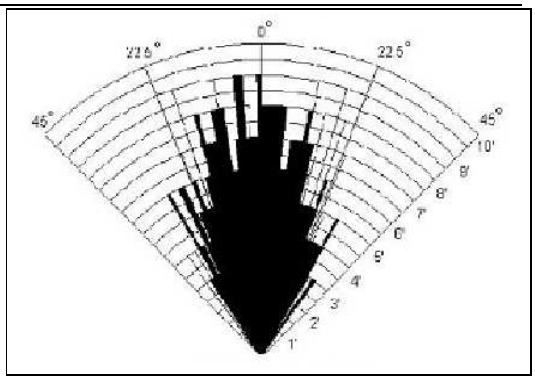



| | |
|---|---|
| VCC | 5V |
| Trig | Pin de Salida |
| Echo | Pin de Entrada |
| GND | GND |
| | |

Este sensor funciona exactamente igual que un radar, de hecho es un pequeño radar. Emite un pulso de sonido a una frecuencia tan alta que es imperceptible para el oído humano y cronometra el tiempo que el sonido tarda en llegar a un obstáculo, rebotar y volver al sensor.

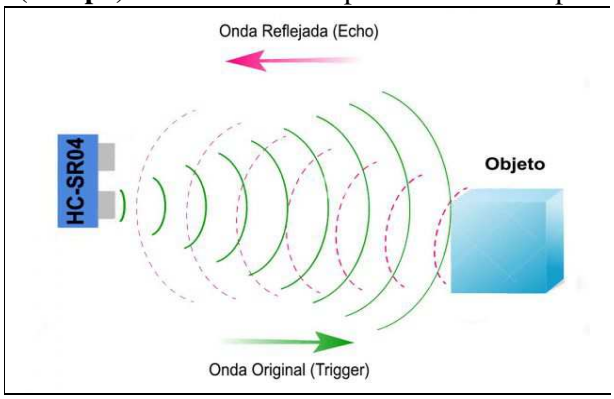
**¿Qué recibimos exactamente en el sensor?**
El tiempo que transcurre entre el envío y la recepción del ultrasonido.

**¿Cómo vamos a traducir dicho tiempo en distancia?**
Aprovechando que la velocidad de dicho ultrasonido en el aire es de valor 340 m/s, o 0,034 cm/microseg (ya que trabajaremos con centímetros y microsegundos).




Para calcular la distancia, recordaremos que **v=d/t** (definición de velocidad: distancia recorrida en un determinado tiempo).

De la fórmula anterior despejamos d, obteniendo **d=v•t**, siendo **v** (**velocidad**) la constante anteriormente citada y **t (tiempo)** el valor devuelto por el sensor a la placa Arduino.



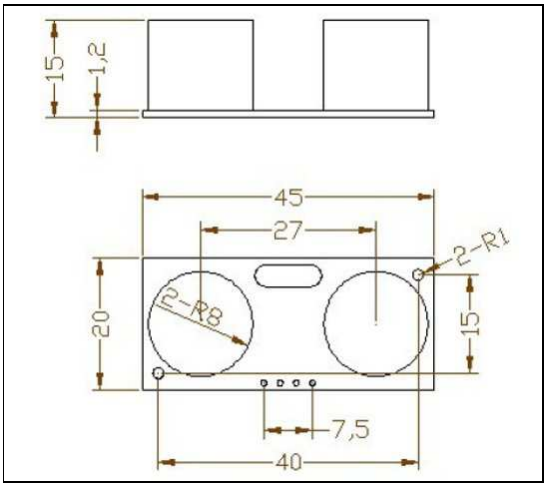

También habrá que dividir el resultado entre 2 dado que el tiempo recibido es el tiempo de **ida y vuelta**.

Arduino es capaz de manejar este sensor sin usar Librerías, sin embrago hay Librerías muy difundidas que ayudan a manejar al "Sensor de Distancia Ultrasónico HC-SR04", tal es el caso de la librería llamada "NewPing" la que podrá encontrar en el paquete de librerías adicionales, junto con la presente guía. Por si resulta de interés, he incluido un Archivo llamado "**Descripción_NewPing.doc**" Que explica las funciones y algunos ejemplos de como usar esta librería. La forma de uso e instalación de esta librería, es la explicada en guías anteriores. **Personalmente soy partidario de utilizar librerías sólo cuando son realmente necesarias.**

## Medidas del Sensor HC-SR04

Acá se muestran las medidas principales del Sensor de Distancia de Ultrasonido HC-SR04. Con ellas podrá diseñar y construir los brackets para sus proyectos o comprarlos en casas especializadas, son muy baratos y prácticos.

**Distintos modelos de Brackets**




**Puede Fabricarlos usted mismo o comprarlos**

## FUNCIONES QUE UTILIZAREMOS EN NUESTRO PROGRAMA.

En este ejercicio usaremos una función distinta a las anteriores, sirve para detectar y leer un pulso o señal entrante por uno de los puertos digitales de Arduino. La Función es "**pulseIn(** )", la cual tiene dos formatos distintos.

- pulseIn (Pin, Valor)
- pulseIn (Pin, Valot, TimeOut)

**Ejemplo " pulseIn (Pin, Valor)"**

```
int Pin = 7;
unsigned long duracion;

void setup( ){
  pinMode(Pin, INPUT);
}
void loop( ){
  duracion = pulseIn(Pin, HIGH);
}
```

## Parámetros:

**Pin:** el número de pin en el que desea leer el pulso. (int)
Valor: tipo de pulso a leer: HIGH o LOW. (int)
**TimeOut:** Este es un parámetro opcional, y es el número de microsegundos que espera a que el pulso se complete, la función devuelve 0 si el pulso completo no se recibe dentro del tiempo de espera. Por defecto es de un segundo, y el dato o variable que lo contenga debe ser del tipo "long" sin signo.
**Nota:** si se usa TimeOut, es recomendable que sea, al menos, 1,3 veces superior a la duración del pulso a medir. Por ejemplo, si se mide un pulso de duración 0,01 segundo, TimeOut deber ser al menos 0,013.

## Retorno

La longitud del pulso (en microsegundos) o 0 si el pulso no se completa dentro del tiempo de espera. El tipo de dato del resultado será "long" sin signo.

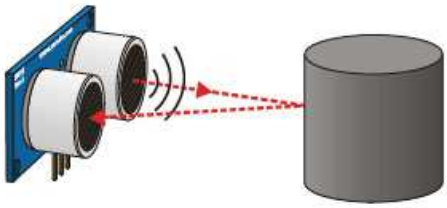
## 7- Reconocimiento de un Sensor de Distancia Ultrasónico HC-SR04. Cálculo de distancia con Valores Entregados.

Lecturas de los valores entregados y cálculo de la distancia. Visualización de Resultados en el monitor del Puerto Serie.

(Programa "007_Distancia_HC_SR04_01")

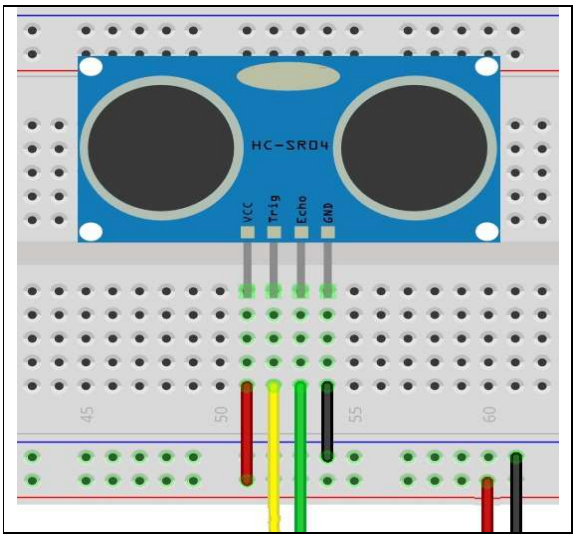
```
1  // Programa que usa librerías propias de Arduino
2  const int PinPulsoSaliente = 9,
3            PinPulsoEntrante = 8;
4  long      TiempoTotal,
5            DistanciaCentimetros;
-
6  void setup( ){
7    Serial.begin(9600);
8    pinMode(PinPulsoSaliente, OUTPUT);
9    pinMode(PinPulsoEntrante, INPUT);
10 }
11 void loop( ){
12   digitalWrite(PinPulsoSaliente,LOW);   //Estabilización del Sensor
13   delayMicroseconds(3);                 // Debo esperar entre 2 y 5 Milisegundos
14   digitalWrite(PinPulsoSaliente, HIGH); //Envío pulso ultrasónico
15   delayMicroseconds(10);                // Espero regreso
16   TiempoTotal = pulseIn(PinPulsoEntrante, HIGH);
17   DistanciaCentimetros= int(TiempoTotal*0.017); //Calcula Distancia en Centímetros
18   Serial.print("Distancia: ");
19   Serial.print(DistanciaCentimetros);
20   Serial.println(" cm");
21   delay(100);
22 }
```

| EXPLICACION LINEAS IMPORTANTES | |
|---|---|
| 8 | Salida del pulso ultrasónico |
| 9 | Entrada del Pulso de Retorno |
| | |
| 12 y 13 | Se apaga el Pulso por un intervalo de 2 a 5 Milisegundo – de esta forma se da por terminado el |

| | |
|---|---|
| | pulso anterior, quedando listo para recomenzar |
| 14 y 15 | Se envía un nuevo pulso por 15 milisegundos, para censar el entorno |
| 16 | Se toma del sensor el tiempo total que tardo el pulso desde que salio hasta que regreso al sensor.<br>Mide el Tiempo Total que transcurrido entre el envío del pulso ultrasónico y cuando el sensor recibe el rebote, es decir: desde que el pin empieza a recibir el rebote, HIGH, hasta que deja de hacerlo, LOW, la longitud del pulso entrante. |
| 17 | Sabiendo que el Pulso recorre 0,034 centímetros por milisegundo, y que el tiempo total, es el que tardo en llegar al objeto y regresar, simplemente se divide por 2 (la mitad de recorrido - Solamente en llegar al objeto), entonces:<br>Distancia = (TiempoTotal*0,034)/2.<br><br>**O lo que es igual:**<br>Distancia = TiempoTotal*0,017.<br><br>  <br>Tiempo = 2 * (Distancia / Velocidad)<br>Distancia = Tiempo · Velocidad / 2<br><br>En el programa, para simplificar los caculos, en la distancia, solo se muestra la parte entera de la división, ignorando los decimales, que representan Milímetros. |
| 18 a 20 | Se muestra por el monitor del puerto serie la distancia en centímetros al objeto que tiene delante |

## CIRCUITO PARA NUESTRO PROYECTO

**Lista de Materiales:** 1 Sensor de Distancia de Ultrasonido HC-SR04, 6 Cables Macho/Macho – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.




**<u>Los cables deberán ser conectados:</u>** De izquierda a derecha, cable <u>Amarillo</u>, el Pin desde el que se enviará el Pulso, cable <u>Verde</u> al PIN desde el que se leerá el pulso de rebote o entrante, Cable Rojo a 5V y cable negro a GND.


## 8- Alarma de Proximidad Versión 01.

Prender un Led rojo como Alarma cuando un objeto se acerque más de una distancia de seguridad. Mostrar por el puerto serie la distancia calculada.

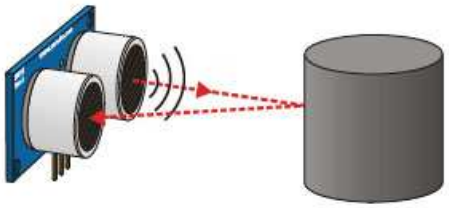
(Programa "007_Distancia_HC_SR04_02_Con_Alarma_de_Led")

```
1  // Programa que usa librerías propias de Arduino
2  const int PinPulsoSaliente = 9,
3            PinPulsoEntrante = 8;
```

```
4   const int PinLedVerde = 5,
5             PinLedRojo  = 7;
6   int     DistanciaSeguraCentimetros = 20,
7           EstadoAlarma;
-
8   long TiempoTotal,
9        DistanciaCentimetros,
10       TiempoAlarmaBeep,
11       TiempoIniAlarma;
-
12  void setup( ){
13    Serial.begin(9600);
14    pinMode(PinPulsoSaliente, OUTPUT);
15    pinMode(PinPulsoEntrante, INPUT);
16    pinMode(PinLedVerde, OUTPUT);
17    pinMode(PinLedRojo,  OUTPUT);
18    EstadoAlarma = 0;
19    TiempoAlarmaBeep = 250;
20    TiempoIniAlarma = millis( );
21  }
22  void loop( ){
23    digitalWrite(PinPulsoSaliente,LOW);
24    delayMicroseconds(3); // entre 2 y 5 Milisegundos
25    digitalWrite(PinPulsoSaliente, HIGH); /*envío pulso ultrasónico*/
26    delayMicroseconds(10);
-
27    TiempoTotal = pulseIn(PinPulsoEntrante, HIGH);
28    DistanciaCentimetros = int(0.017*TiempoTotal);
-
29    if(DistanciaCentimetros <= DistanciaSeguraCentimetros){
30      digitalWrite(PinLedVerde, LOW);
31      if(millis( ) > TiempoIniAlarma + TiempoAlarmaBeep ){
32        EstadoAlarma = 1 - EstadoAlarma;
33        TiempoIniAlarma = millis( );
34      }
35      digitalWrite(PinLedRojo, EstadoAlarma);
36    }else{
37      digitalWrite(PinLedVerde, HIGH);
38      digitalWrite(PinLedRojo, LOW);
39      TiempoIniAlarma = millis( );
40      EstadoAlarma = 0;
41    }
42    Serial.print("Distancia en Centímetros: ");
43    Serial.print(DistanciaCentimetros);
44    Serial.println(" cm");
45   // delay(100);
46  }
```

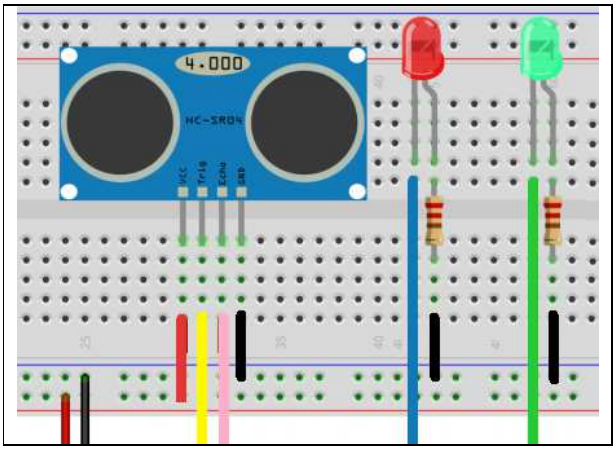
## EXPLICACION LINEAS IMPORTANTES

| | |
|---|---|
| 1 | Explicación de Librerías. |
| | |
| 8 | Salida del pulso ultrasónico |
| 9 | Entrada del Pulso de Retorno |
| | |
| 23 y 24 | Se apaga el Pulso por un intervalo de 2 a 5 Milisegundo – de esta forma se da por terminado el |

| | |
|---|---|
| | pulso anterior, quedando listo para recomenzar. |
| 25 y 26 | Se envía un nuevo pulso por 15 milisegundos, para censar el entorno |
| 27 | Se toma del sensor el tiempo total que tardo el pulso desde que salio hasta que regreso al sensor.<br>Mide el Tiempo Total que transcurrido entre el envío del pulso ultrasónico y cuando el sensor recibe el rebote, es decir: desde que el pin empieza a recibir el rebote, HIGH, hasta que deja de hacerlo, LOW, la longitud del pulso entrante. |
| 28 | Sabiendo que el Pulso recorre 0,034 centímetros por milisegundo, y que el tiempo total, es el que tardo en llegar al objeto y regresar, simplemente se divide por 2 (la mitad de recorrido - Solamente en llegar al objeto), entonces:<br>Distancia = (TiempoTotal*0,034)/2.<br><br>**O lo que es igual:**<br>Distancia = TiempoTotal*0,017.<br><br>  <br>Tiempo = 2 * (Distancia / Velocidad)<br>Distancia = Tiempo · Velocidad / 2<br><br>En el programa, para simplificar los caculos, en la distancia, solo se muestra la parte entera de la división, ignorando los decimales, que representan Milímetros. |
| 29 a 36 | Si el objeto esta dentro de la zona de seguridad, apago LED Verde, y ALARMA (hago parpadear el rojo) |
| 36 a 41 | Si no hay elemento dentro de la zona de seguridad, apago Alarma y Prendo LED Verde |
| 18 a 20 | Se muestra por el monitor del puerto serie la distancia en centímetros al objeto que tiene delante. |

## CIRCUITO PARA NUESTRO PROYECTO

**Lista de Materiales:** 1 Sensor de Distancia de Ultrasonido HC-SR04, 1 Led Vere, 1 Led Rojo, 2 Resistencias de 470Ω (Para el Led), 10 Cables Macho/Macho – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

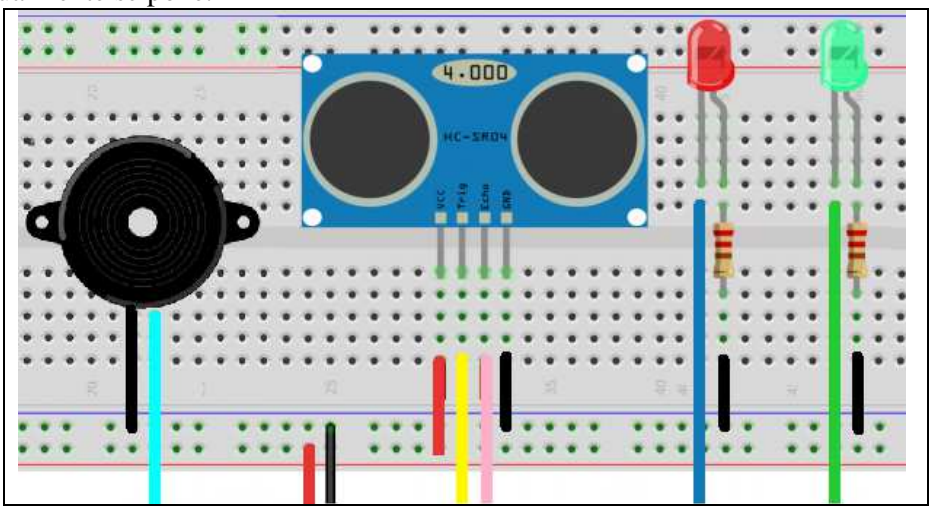



**Los cables deberán ser conectados:** De izquierda a derecha, cable Rojo a 5V y cable negro a GND, cable Amarillo al Pin desde el que se enviará el Pulso, cable Rosa al PIN desde el que se leerá el pulso de rebote o entrante, Cable Azul al PIN destinado al LED rojo de alarma, y cable Verde al PIN destinado al LED Verde.


**9- Alarma de Proximidad Versión 02. Modificar Ejercicio Anterior para que haga un sonido de Alarma, simultáneamente al parpadeo del LED rojo.** Cuando un objeto se encuentre dentro de la distancia segura. Tarea para el alumno. Les dejo la imagen del circuito.

**Ejercicios Propuestos:**

**a-** Agregar un sonido de alarma, que suene simultáneamente mientras parpadea el LED rojo. En este ejercicio he utilizado un parlante (reciclado) de 1,5W y 8Ω por lo tanto no Hace falta la resistencia que usualmente se pone.




**b-** Modificar la versión A del programa, para que la distancia segura se divida en dos partes, entonces si el objeto que ingresa al área segura se encuentra en la mitad exterior, parpadee el LED Rojo, y haga un sonido indicando la alerta, pero cuando el objeto ingrese al área segura más pequeña, cambie el sonido a uno más chillón.

**c-** Agregar un Potenciómetro que permita configurar el valor de la Distancia segura.

(Programa "007_Distancia_HC_SR04_04_Alarma_de_Led_Pot_y_Soni")

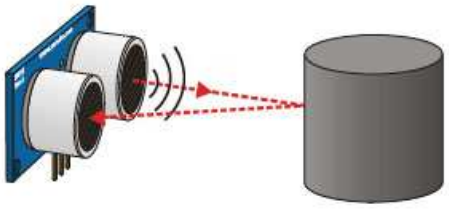
```
1   // Programa que usa librerías propias de Arduino
2   const int PinLedVerde = 4,
3             PinLedRojo  = 5;
-
4   const int PinPulsoSaliente = 7,
5             PinPulsoEntrante = 8;
-
6   const int SpeakerPin  = 9,
7             PinPotencio = A0;
-
8   int     DistanciaSeguraCentimetros,
9           EstadoAlarma,
10          ValorFrecuencia = 1500,
11          Frecuencia = 0;
-
12  long    TiempoTotal,
13          DistanciaCentimetros,
14          TiempoAlarmaBeep,
15          TiempoIniAlarma;
-
16  void setup( ){
17    Serial.begin(9600);
18    pinMode(PinLedVerde, OUTPUT);
19    pinMode(PinLedRojo,  OUTPUT);
20    pinMode(PinPulsoSaliente, OUTPUT);
21    pinMode(PinPulsoEntrante, INPUT);
22    pinMode(SpeakerPin,  OUTPUT);
23    pinMode(PinPotencio, INPUT_PULLUP );
24    EstadoAlarma = 0;
25    TiempoAlarmaBeep = 150;
```

Todas las constantes del mismo tipo, podrían haberse definido juntos, pero para aportar mayor claridad en el código decidí agruparlas:
**Primero:** los pines correspondientes a los LEDS
**Segundo:** Pulsos entrante y Saliendo del Sensor.
**Tercero:** Pines usados por el Speaker y Potenciómetro.
**Cuarto:** Variables de proceso.

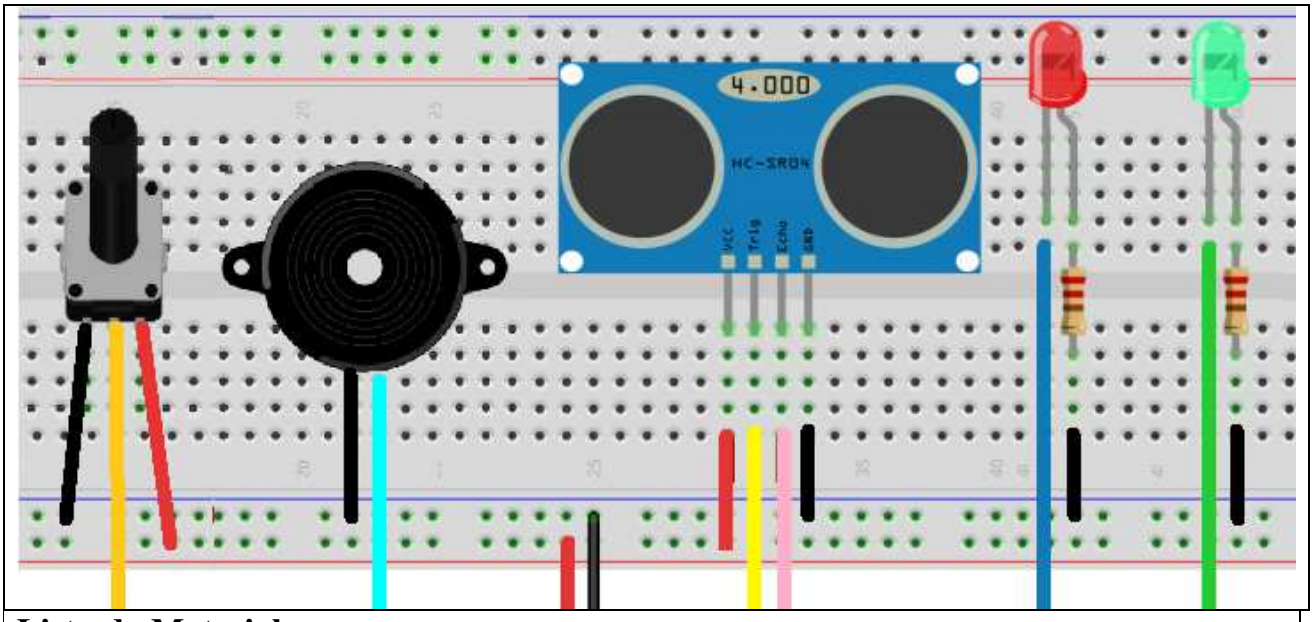
```
26    TiempoIniAlarma = millis( );
27   }
28   void loop( ){
29    digitalWrite(PinPulsoSaliente,LOW);   /* Apago sensor para Iniciar proceso*/
30    delayMicroseconds(3);                 // Espero entre 2 y 5
31    digitalWrite(PinPulsoSaliente, HIGH); /* envío del pulso ultrasónico*/
32    delayMicroseconds(10);
33    TiempoTotal=pulseIn(PinPulsoEntrante, HIGH);
-
34    DistanciaCentimetros = int(0.017*TiempoTotal);
-
35    if(DistanciaCentimetros <= DistanciaSeguraCentimetros){
36      digitalWrite(PinLedVerde, LOW);
37      if(millis( ) > TiempoIniAlarma + TiempoAlarmaBeep ){
38        EstadoAlarma = 1 - EstadoAlarma;
39        Frecuencia   = ValorFrecuencia - Frecuencia; // Cambia Frecuencia del Sonido
40        TiempoIniAlarma = millis( );
41      }
42      if(DistanciaCentimetros <= DistanciaSeguraCentimetros/2){
43        tone(SpeakerPin, Frecuencia);
44      }else{
45        tone(SpeakerPin, 0);
46      }
47      digitalWrite(PinLedRojo, EstadoAlarma);
48    }else{
49      digitalWrite(PinLedVerde, HIGH);
50      digitalWrite(PinLedRojo, LOW);
51      Frecuencia = 0;
52      EstadoAlarma = 0;
53      noTone(SpeakerPin);
54    }
-
55    DistanciaSeguraCentimetros = analogRead(PinPotencio)/3;
-
56    Serial.print("D. Segura: ");
57    Serial.print(DistanciaSeguraCentimetros);
58    Serial.print(" – D. Objeto: ");
59    Serial.print(DistanciaCentimetros);
60    Serial.println(" cm");
61    //delay(50); // Solo para ver mejor Mensajes en Monitor Puerto Serie
62   }
```

## EXPLICACION LINEAS IMPORTANTES

| | |
|---|---|
| 1 | Explicación de Librerías. |
| 2 y 3 | Declararon Constantes con los números de Pines usados por LED verde y Rojo |
| 4 y 5 | Declararon Constantes con los números de Pines usados para los Pulsos Entrantes y Salientes del Sensor de Distancia de Ultrasonido HC-SR04 |
| 6 y 7 | Declararon Constantes con los números de Pines usados para el Speaker y Potenciómetro. |
| 8 a 11 | Declararon Variables de Proceso del tipo int. |
| 12 a 15 | Declararon Variables de Proceso del tipo long |
| 16 a 27 | Sección de configuración y procesos únicos en el programa |
| | |
| 29 y 30 | Se apaga el Pulso por un intervalo de 2 a 5 Milisegundo – de esta forma se da por terminado el pulso anterior, quedando listo para recomenzar. |

| | |
|---|---|
| 31 y 32 | Se envía un nuevo pulso por 15 milisegundos, para censar el entorno |
| 33 | Se toma del sensor el tiempo total que tardo el pulso desde que salio hasta que regreso al sensor.<br>Mide el Tiempo Total que transcurrido entre el envío del pulso ultrasónico y cuando el sensor recibe el rebote, es decir: desde que el pin empieza a recibir el rebote, HIGH, hasta que deja de hacerlo, LOW, la longitud del pulso entrante. |
| 34 | Sabiendo que el Pulso recorre 0,034 centímetros por milisegundo, y que el tiempo total, es el que tardo en llegar al objeto y regresar, simplemente se divide por 2 (la mitad de recorrido - Solamente en llegar al objeto), entonces:<br>Distancia = (TiempoTotal*0,034)/2.<br><br>**O lo que es igual:**<br>Distancia = TiempoTotal*0,017.<br><br>  <br>Tiempo = 2 * (Distancia / Velocidad)<br>Distancia = Tiempo · Velocidad / 2<br><br>En el programa, para simplificar los caculos, en la distancia, solo se muestra la parte entera de la división, ignorando los decimales, que representan Milímetros. |
| 35 a 48 | Si el objeto esta dentro de la zona de seguridad, apago LED Verde, y ALARMA |
| 48 a 54 | Si no hay elemento dentro de la zona de seguridad, apago Alarma y Prendo LED Verde |
| 55 | Leo el valor de configuración del potenciómetro, por si se ha cambiado o reconfigurado el valor en centímetros de la distancia segura. |
| 56 a 61 | Se muestra por el monitor del puerto serie la distancia en centímetros al objeto que tiene delante. |
| 62 | Fin del programa. |

## CIRCUITO PARA NUESTRO PROYECTO




**Lista de Materiales:** 1 Sensor de Distancia de Ultrasonido HC-SR04, 1 Led Verde, 1 Led Rojo, 2 Resistencias de 470Ω (Para el Led), 1 Potenciómetro, un speaker (En este ejercicio he utilizado un parlante (reciclado) de 1,5W y 8Ω por lo tanto no Hace falta la resistencia que usualmente ponemos), 15 Cables Macho/Macho – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

**Los cables deberán ser conectados:** De izquierda a derecha, cable Mostaza al Pin Analógico destinado al Potenciómetro, cable Celeste, al PIN destinado al Speaker, Rojo a 5V y cable Negro a GND, cable Amarillo, al Pin desde el que se enviará el Pulso, cable Rosa al PIN desde el que se leerá el pulso de

rebote o entrante, Cable Azul al PIN destinado al LED rojo de alarma, y cable Verde al PIN destinado al LED Verde.

**IDEAS:** Podríamos haber usado un Segundo Potenciómetro para regular el tiempo entre Beep y Beep de la alarma, y una Fotorresistencia LDR para hacer que esta alarma solo se active al anochecer. Etc. Otra versión de la alarma podría ser prenderla mediante un botón (Prendido/Apagado) y luego si esta prendida, automáticamente se active al anochecer. Más adelante retomaremos estas ideas cuando usemos un **Display** para visualizar directamente en la Placa Arduino mensajes de estado.


01001101 01101001 01110011 00100000 01010000 01100101 01101110 01110011 01100001
01101101 01101001 01100101 01101110 01110100 01101111 01110011 00100000 01010000
01110010 01100101 01101110 01100100 01100101 01101110 00100000 01001100 01110101
01100011 01100101 01110011 00100000 01100101 01101110 00100000 01110101 01101110
00100000 01000011 01101001 01110010 01100011 01110101 01101001 01110100 01101111

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