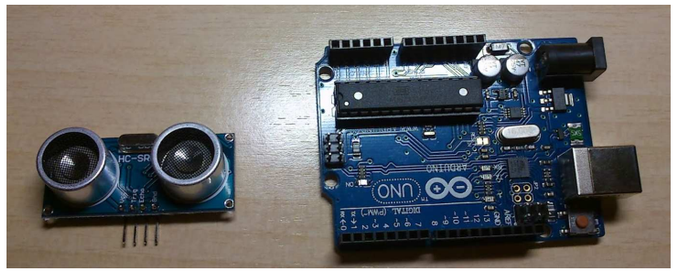
Sensor ultrasonidos HC-SR04 (*Recopilado de Internet - Adaptación Prof: Bolaños DJB. Versión 18-10-19*)

El sensor de ultrasonidos se enmarca dentro de los sensores para medir distancias o superar obstáculos, entre otras posibles funciones.
En este caso vamos a utilizarlo para la medición de distancias. Esto lo consigue enviando un ultrasonido (inaudible para el oído humano por su alta frecuencia) a través de uno de la pareja de cilindros que compone el sensor (un transductor) y espera a que dicho sonido rebote sobre un objeto y vuelva, retorno captado por el otro cilindro.



Este sensor en concreto tiene un rango de distancias sensible entre 3cm y 3m con una precisión de 3mm.

COMO FUNCIONA UN SENSOR ULTRASÓNICODE DISTANCIA

Hemos visto, en los documentales, que los murciélagos son capaces de volar en completa oscuridad y sin embargo, sortear obstáculos o atrapar insectos en vuelo. Sabemos que lo hacen, pero rara vez pensamos como.

Tenemos una vaga idea de que se llama ecolocalización y que más o menos tiene que ver con unos sonidos agudos que emiten y que después recogen con esas enormes orejas que Dios les ha dado, pero rara vez nos planteamos cómo es esto posible.
Delfines y ballenas utilizan un sistema similar para atrapar a sus presas, y hasta hemos visto que, en cualquier película de submarinos, en el momento álgido el capitán ordena emitir un pulso único de sonar para localizar al enemigo.
El concepto básico, es siempre el mismo, sabiendo a qué velocidad viaja el sonido, si emitimos un pulso sónico corto y escuchamos cuanto tiempo tarda en regresar el eco podemos calcular la distancia a la que se encuentra el objeto en el que ha rebotado la señal.

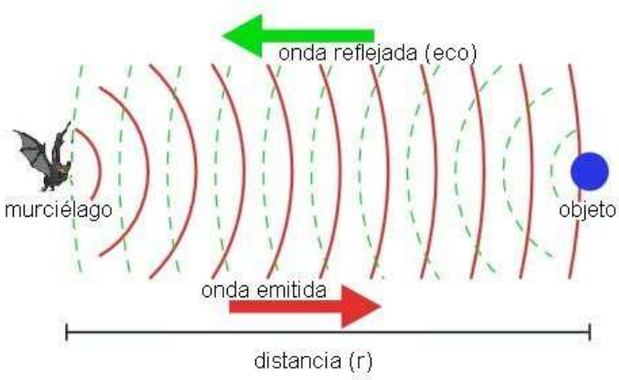
•El radar funciona de modo similar aunque usando ondas de radio frecuencia muy cortas y con una problemática propia descomunal. Un pulso de radiofrecuencia se emite desde la antena y se recoge el eco que vuelve a la velocidad de la luz.

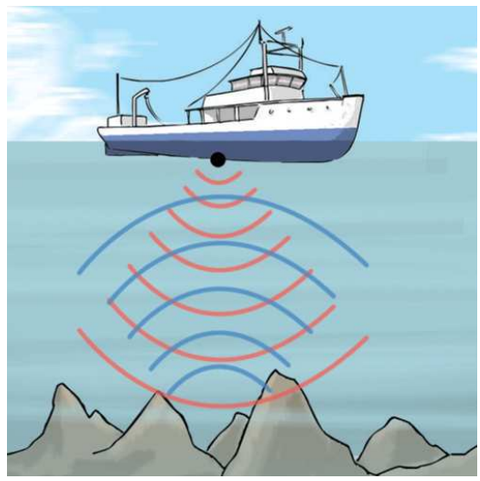
Lo que haremos en esta sesión es utilizar un sensor de distancia sencillo HC-SR04 (y muy parecido a los sensores de aparcamiento de los coches modernos), que nos permite enviar estos pulsos ultrasónicos y escuchar el eco de retorno. Midiendo este tiempo, podemos calcular la distancia hasta el obstáculo.

•El oído humano no percibe sonidos por encima de 20kHz. Por eso, a las ondas de mayor frecuencia las llamamos ultrasonidos (mas allá del sonido).

Los sensores de ultrasonidos funcionan sobre los 40 kHz.

•No son perfectos, les influye la temperatura ambiente, la humedad y los materiales en los que reflejan, lo que genera una cierta incertidumbre. Pero a cambio son baratos y efectivos hasta un poco más de 3 metros en condiciones normales si la precisión no es un problema determinante.





¿Qué recibimos en el sensor?

El tiempo que transcurre entre el envío y la recepción del ultrasonido.

¿Cómo vamos a traducir dicho tiempo en distancia?

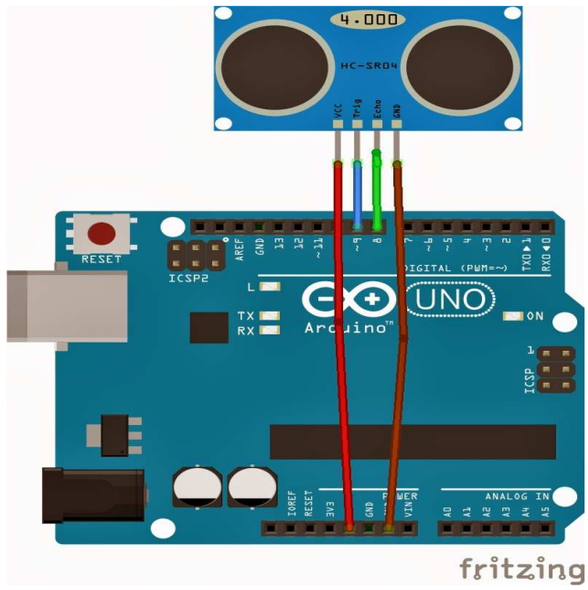
Aprovechando que la velocidad de dicho ultrasonido en el aire es de valor 340 m/s, o 0,034 cm/microseg (ya que trabajaremos con centímetros y microsegundos). Para calcular la distancia, recordaremos que v=d/t (definición de velocidad: distancia recorrida en un determinado tiempo).
De la fórmula anterior despejamos d, obteniendo:

$$d=v\cdot t$$

siendo v la constante anteriormente citada (o sea la velocidad del sonido en el aire) y t 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.

Materiales

Sensor ultrasonidos HC-SR04
Placa Arduino UNO
Cables
Cable USB
Protoboard



El sensor consta de 4 pines:

VCC: conectado a la salida de 5V de la placa.

Trig: conectado al pin digital de la placa encargado de enviar el pulso ultrasónico.

Echo: al pin de entrada digital que recibirá el eco de dicho pulso.

GND: a tierra.

Para el manejo de este sensor podemos usar o no una libreria que hay disponible:

NewPing_v1.5.zip (Para Arduino)

Library-ULTRASONIC.rar (Para simular en PROTEUS -C*arpeta de recursos del Tutor de Arduino*)

Primero analizaremos un caso de programa que no necesita de la librería mencionada, el funcionamiento se basa en la instrucción: **pulseIn (pin , HIGH)**

Es una funcion de Arduino, mide el tiempo 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. O sea el tiempo entre que la señal que le llega a el lo pone en alto hasta que lo pone en bajo. Ese tiempor medido en microsegundos.

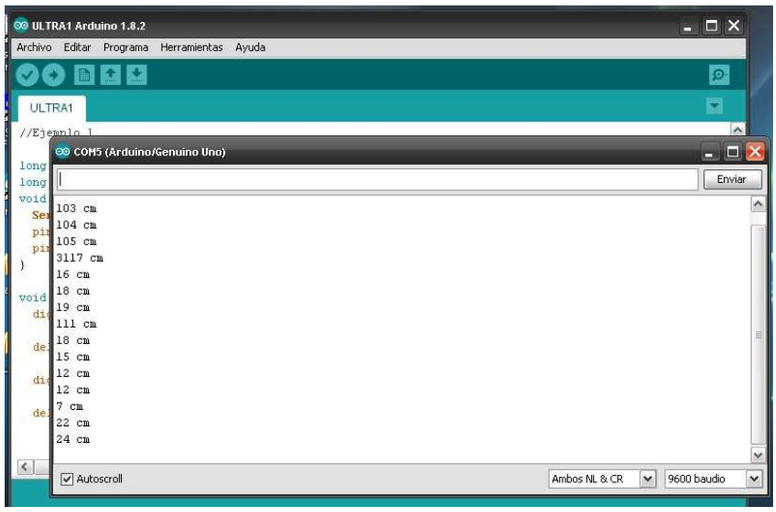
Programa

```
//Programa ULTRA1

long distancia;
long tiempo;
void setup(){
  Serial.begin(9600);
  pinMode(9, OUTPUT); /*activación del pin 9 como salida: para el pulso ultrasónico*/
  pinMode(8, INPUT); /*activación del pin 8 como entrada: tiempo del rebote del ultrasonido*/
}

void loop(){
  digitalWrite(9,LOW); /* Por cuestión de estabilización del sensor*/
  delayMicroseconds(5);
  digitalWrite(9, HIGH); /* envío del pulso ultrasónico*/
  delayMicroseconds(10);
  tiempo=pulseIn(8, HIGH); /* pulseIn es Función de Arduino, y es para medir la longitud del pulso entrante. Mide el tiempo que transcurrido entre el envío
  del pulso ultrasónico y cuando el sensor recibe el rebote, es decir: des de que el pin 8 empieza a recibir el rebote, HIGH, hasta que
  deja de hacerlo, LOW, la longitud del pulso entrante*/
  distancia= int(0.017*tiempo); /*fórmula para calcular la distancia obteniendo un valor entero*/
  /*Monitorización en centímetros por el monitor serial*/
  Serial.println("Distancia ");
  Serial.println(distancia);
  Serial.println(" cm");
  delay(1000);
}
```

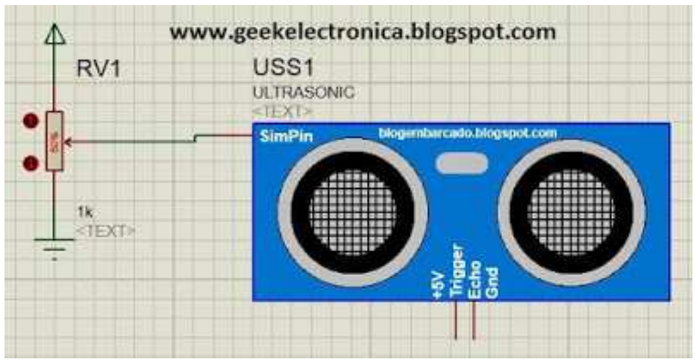
RESULTADO:



Simulación en Proteus del Sensor de ultrasonidos HC-SR04

Si instalamos la biblioteca Simulino para simular Arduino, entonces también tendremos el sensor de sonido mencionado para simular.

Luego de seleccionar e importar el sensor al área de trabajo, también es necesario seleccionar un potenciómetro que se debe configurar como divisor de voltaje y con el cual se va a simular la variación de distancia, el potenciómetro debe ser conectado al terminal del sensor llamado SimPin tal como se muestra a continuación.

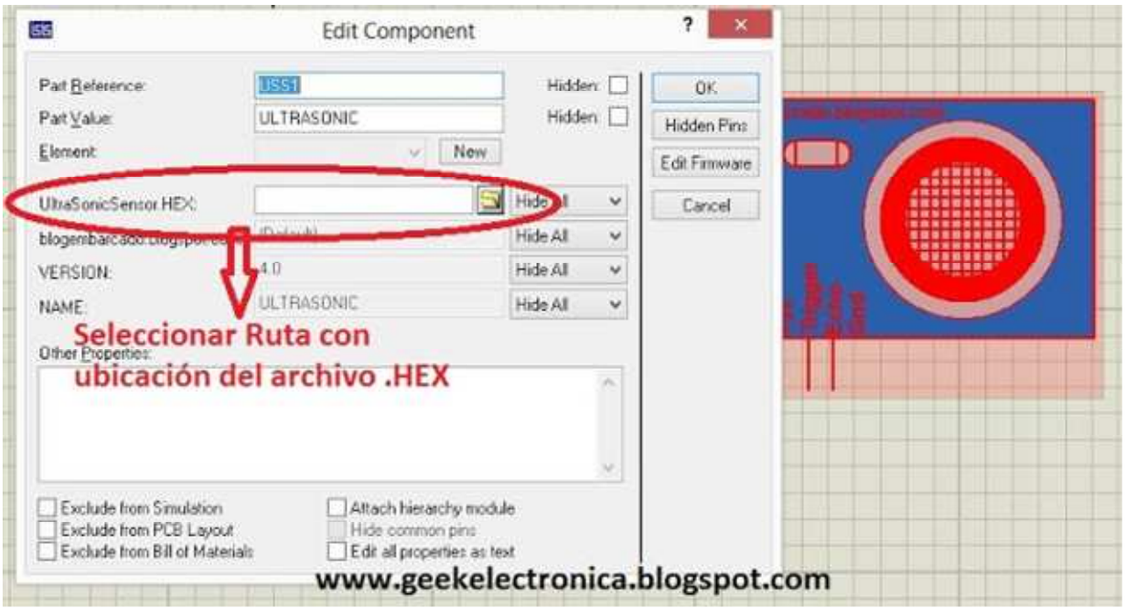


Este modelo de sensor está construido a partir de un microcontrolador que contiene internamente y es el que recibe la variación de voltaje del potenciómetro con lo que se indica una variación de distancia, los otros dos terminales que tiene este sensor deben ser conectados a la tarjeta arduino, microcontrolador o cualquier dispositivo con el que se vaya a completar el proyecto.

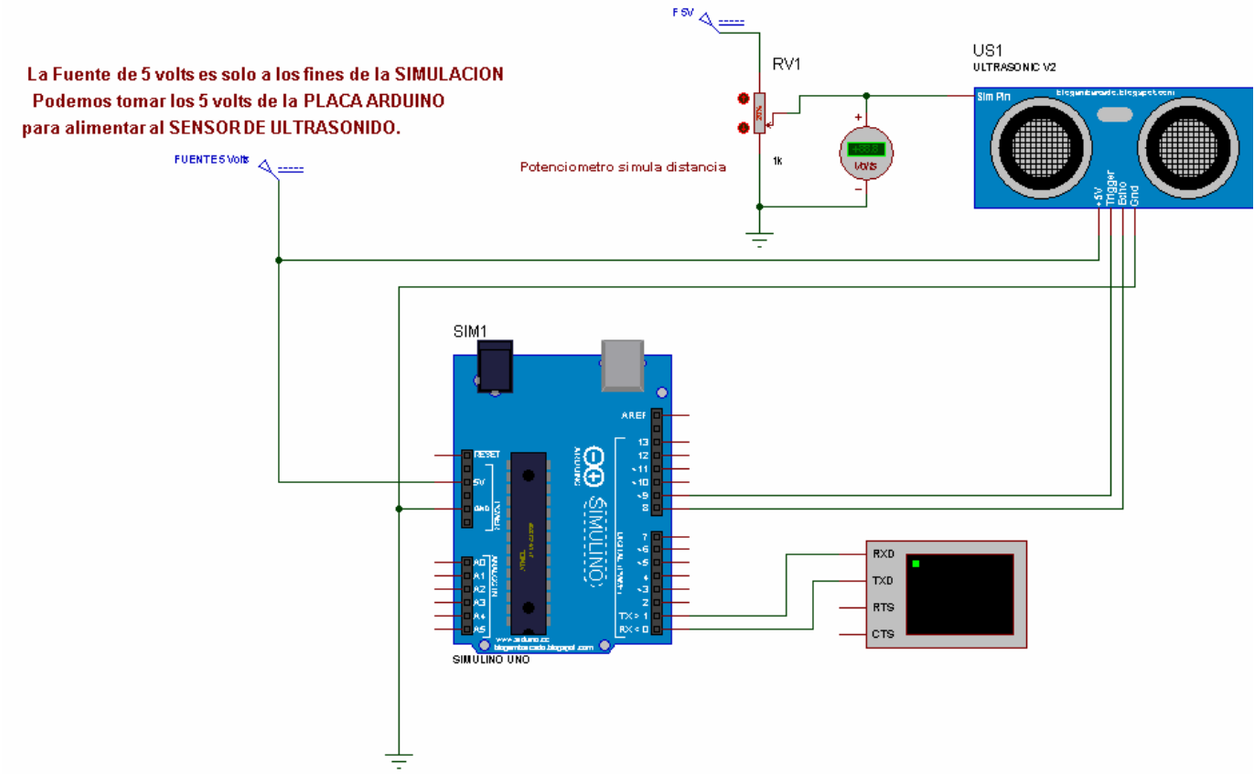
El terminal **Trigger** recibe un pulso que le indica al sensor el momento en el que se va a realizar una medición.

El terminal **Echo** entrega un pulso que indica la distancia que se está midiendo, esta distancia se calcula a partir del tiempo que dura el pulso y de la velocidad del sonido.

Por último como ya se menciono, este modelo está construido a partir de un microcontrolador, por lo tanto antes de simular es necesario cargar el archivo .HEX que contiene el programa de ese microcontrolador que se encuentra al interior del modelo, para esto se siguen los pasos acostumbrados para cargar un programa en un microcontrolador en proteus, se hace un doble click sobre el dispositivo y en la ventaja emergente se selecciona la opción UltraSonicSensor.HEX y se selecciona la ruta en donde se encuentre alojado el archivo .HEX que viene incluido en la carpeta que se descargo previamente.



A continuación vemos el esquemático para Proteus:



Se observa que al mover el potenciómetro simula que mide diferentes distancias.
NO OLVIDAR que se debe cargar nuestro programa Arduino en la placa Arduino y el programa **UltraSonicSensor.HEX** en el sensor de ultrasonido.

APENDICE: Explicación de donde sale la formula

```
distancia= int(0.017*tiempo);
```

```
/*fórmula para calcular la distancia obteniendo un valor entero*/
```

Para escuchar el pulso vamos a usar otra función, pulseIn() .

Básicamente lo que hace es escuchar el pin que le pasamos, buscando una señal que pase de LOW a HIGH (si le pasamos HIGH como parámetro) y cuenta el tiempo que tarda en volver a bajar desde que sube.
Ahora ya sabemos el tiempo que tarda en volver el eco en µs. Como la velocidad del sonido es de 343 metros / segundo, Necesitamos 1/343 = 0,00291 segundos para recorrer un metro.
Para usar una medida más cómoda podemos pasar esto a microsegundos por centímetro:

$$0{,}00291 * \frac{1.000.000\ \mu s}{segundo} * \frac{1\ metro}{100\ cm} = 29{,}1\ \mu s / cm$$

Como nuestro eco mide el tiempo que tarda el pulso en ir y venir la distancia recorrida será la mitad:

$$distancia = \frac{duracion}{29{,}1} * \frac{1}{2}$$

Si operamos sobre los números: **(1/ 2* 29,1) = 0.017**

Ejemplo usando la librería NewPing

*(Puede encontrar una copia en carpeta BIBLIOTECAS del Tutor de Arduino con el nombre **NewPing_v1.5.zip**)*

```
#include <NewPing.h>

#define TRIGGER_PIN 9 // Arduino pin trigger
#define ECHO_PIN 8    // Arduino pin echo
#define MAX_DISTANCE 200

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // Configura pines

void setup()
        {
           Serial.begin(9600); // Open serial monitor a 115200 baud         }

void loop()
       {
           delay(50);
           unsigned int uS = sonar.ping(); //Enviar ping, obtener tiempo de ping
                                           // en microseconds (uS)
           Serial.print("Ping: ");
           Serial.print(uS / US_ROUNDTRIP_CM);
           Serial.println("cm");
       }
```

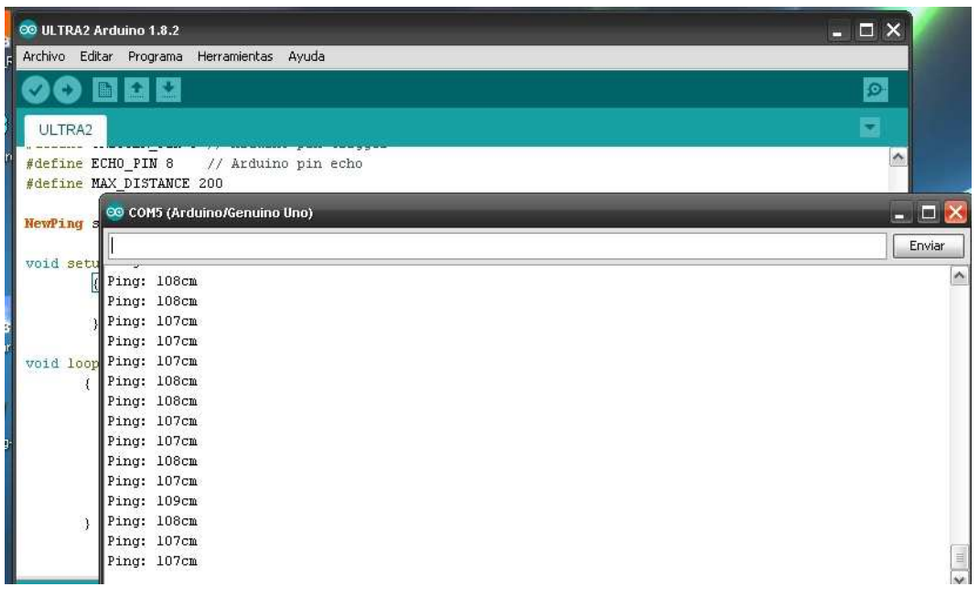
Como vemos la librería se encarga de inicializar los pines necesarios, enviar los pulsos, escuchar el eco de retorno y de hacer los cálculos.

Las instrucciones claves son primero inicializar la librería con:

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE) ;

Y después medir la distancia: unsigned int uS = sonar.ping() ;

El resultado sería:



(Continuara)