CAPÍTULO 6:

SENSOR DE ULTRASONIDO

Sensor ultrasónico

El monitor serie es espacio del IDE Arduino que nos brinda la información del entorno que reciben los sensores conectados al Arduino. En este capítulo vamos a usar por primera vez esta herramienta para ver el funcionamiento de nuestro sensor.

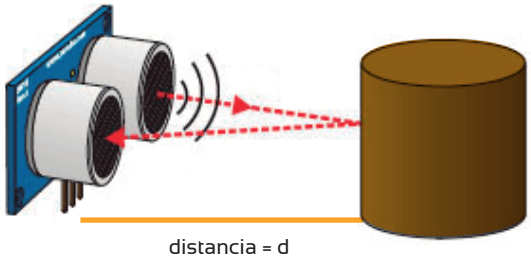
Iremos leyendo la recepción de ondas que recibe el receptor del sensor y ver las variaciones que se generan al desplazar objetos dentro del radio de trabajo del mismo.

Consejo:

El sensor emite y recibe las ondas producidas. El receptor trabaja mejor cuando la onda pega contra caras planas, ya que realizan su rebote es mucho mas parejo que las caras curvas u orgánicas.

Sensor de ultrasonido

Este sensor nos permite medir la distancia a la que se encuentra nuestro robot de un objeto a través de ondas de sonido de alta frecuencia. Estas ondas tienen una velocidad (**v**) estable, por lo que si medimos el tiempo (**t**) que demora en ir y volver hasta un objeto, podemos calcular la distancia (**d**) que lo separa del mismo. Sin embargo, al resultado hay que dividirlo en dos, ya que el tiempo que se tomó fue de ida y vuelta.

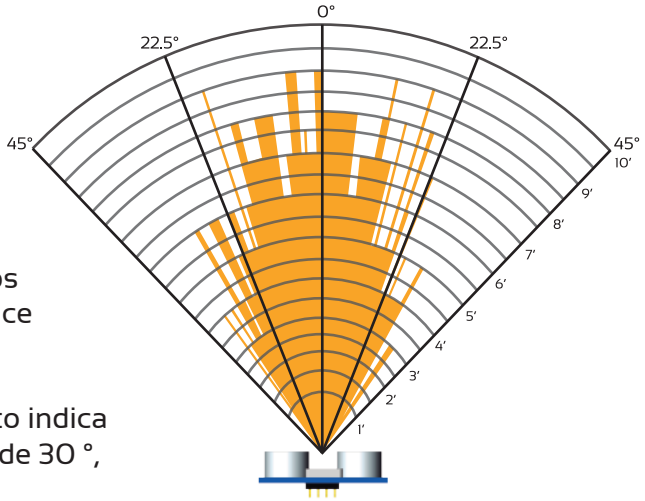


$$v = d / t$$

$$d = v * t$$

$$\mathbf{d = (v * t) / 2}$$

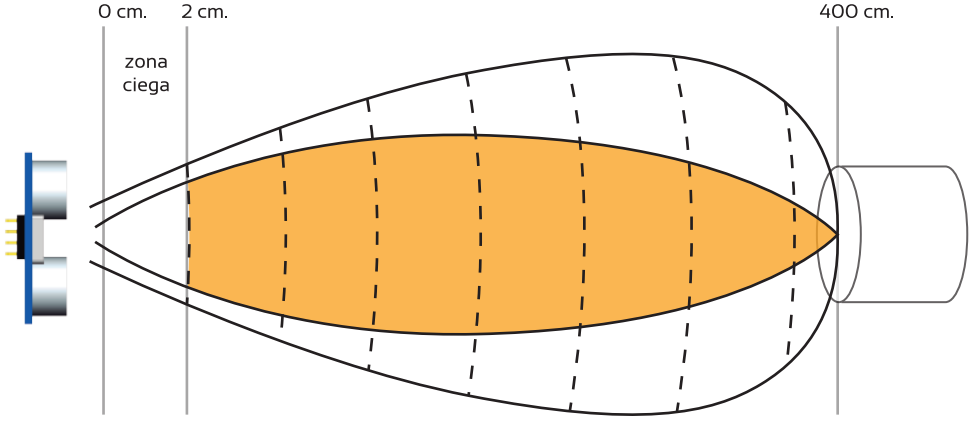
La capacidad de detección varía según la ubicación que tenga el objeto dentro del campo de visión del sensor. Las zonas marcadas en amarillo son las de mayor visibilidad. Esto varía mucho dependiendo del material en que esté hecho el objeto, ya que los que estén construidos con algodón o materiales esponjosos o blandos absorben las ondas y reducen el alcance del sensor.



El ángulo efectivo para este sensor es de 15 °, esto indica que el sensor podrá detectar objetos a un rango de 30 °, 15 ° hacia la derecha e izquierda del ángulo cero.

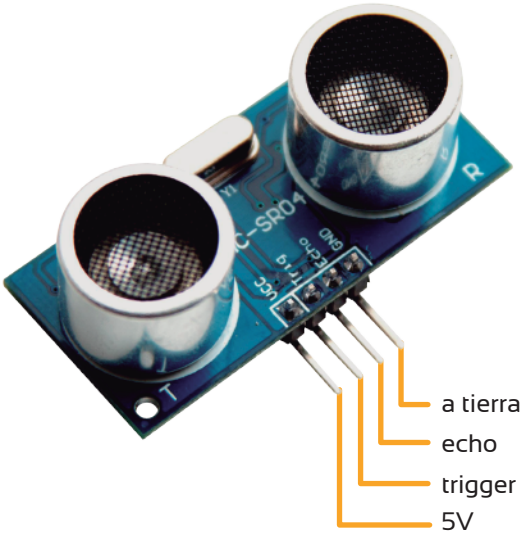
¿Cuáles son las ventajas y desventajas de este tipo de sensores?

Los sensores de ultrasonido pueden detectar objetos con diferentes formas, colores, superficies e, incluso, materiales. Otra gran ventaja de esta clase de sensores es que, al no necesitar el contacto físico con el objeto, ofrecen la posibilidad de detectar objetos frágiles o con pintura fresca. Los materiales que capta pueden ser sólidos, líquidos o polvorientos, sin embargo han de ser deflectores de sonido para que el sensor pueda recibir de regreso las ondas que envía. Trabajan solamente en el aire y tienen una ***zona ciega*** comprendida entre el sensor y el alcance mínimo en el que un objeto puede detectarse de forma fiable; dentro de esta zona, los resultados arrojados por el sensor pueden ser erróneos.



Referencias
2 cm. : alcance mínimo
400 cm.: alcance máximo
0 cm. a 2 cm. : zona ciega
2 cm. a 400 cm.: rango de sensado

Más información sobre su funcionamiento

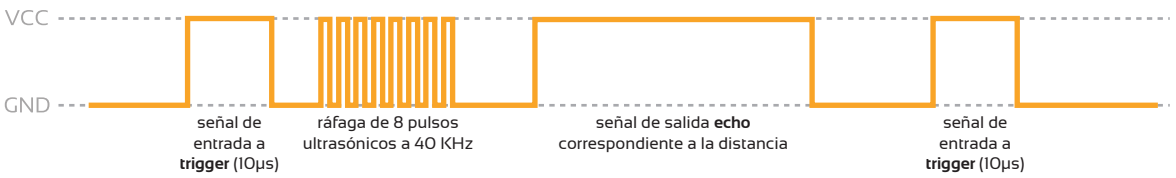


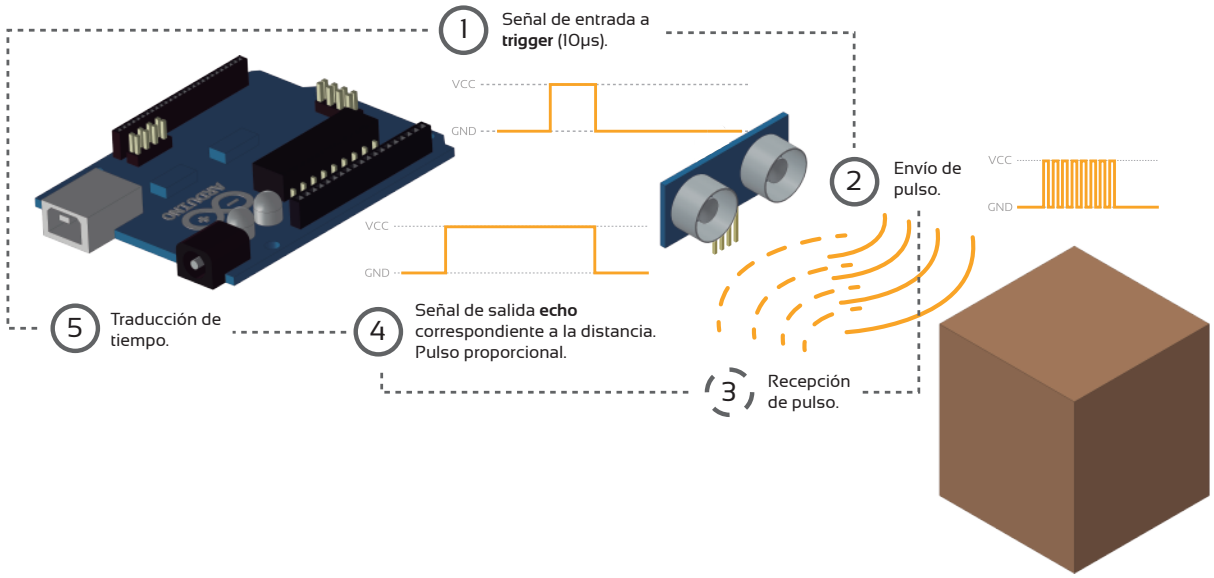
La interacción con arduino se logra mediante dos pines digitales: el pin de trigger (disparador) y echo (eco).

El primero (**trigger)** recibe un pulso de habilitación de parte del microcontrolador, mediante el cual se le indica al módulo que comience a realizar la medición de distancia.

A través de un segundo pin (**echo**), el sensor "muestra" al microcontrolador un pulso cuyo ancho es proporcional al tiempo que tarda el sonido en viajar del transductor al obstáculo y luego de vuelta al módulo.

Los otros dos pines restantes **GND** (ground, a tierra o 0V) y **VCC** (que corresponde al voltaje, en este caso 5V) se encargan de manejar la corriente eléctrica que ingresa al sensor para permitir que **trigger** y **echo** cumplan con su función. Así, cada vez que el sensor reciba la orden de ejecutar un cálculo de distancia precisará energía para hacerlo, lo que significa que un ciclo de consulta a través del sensor de ultrasonido en cuestión de corriente eléctrica se vería de la siguiente manera:





Función pulseIn

Esta función lee un pulso que ingresa por un pin determinado y entrega como resultado el tiempo en microsegundos que duró dicho pulso.

En nuestro caso, el sensor ultrasónico, usando el pin "**echo**", envía al arduino un pulso cuyo ancho es proporcional al tiempo que tarda el sonido en ir desde el sensor al obstáculo y luego de vuelta al sensor (como pueden observar en el gráfico de arriba). Arduino estará controlando el pin que pasamos como parámetro a la función pulseIn y, en el momento que ingrese un nivel alto (5v) comenzará a contar el tiempo. La cuenta se detendrá cuando el voltaje vuelva a ser 0, lo que significa que el pulso ha terminado.

Su sintaxis correcta es: **pulseIn(pin,valor);**

El valor puede ser HIGH o LOW dependiendo del tipo de pulso que se quiera medir. Como resultado, devuelve el tiempo en microsegundos que duró el pulso.

Ejemplo:

```
tiempo = pulseIn(7,HIGH); /*cuando detecte un pulso HIGH en el pin 7 medirá los ms que
                            demora hasta volver a LOW. El resultado será el valor de "tiempo" */
```

Estructuras condicionales

También hay estructuras condicionales con las que podemos trabajar dentro del código para darle instrucciones a nuestro robot y controlar así el flujo de información que recibe.

A continuación veremos que hay dos maneras de declarar una estructura condicional:

La estructura "if"

La primera estructura condicional que veremos es **"if"**, que significa "Si..." en inglés. Dentro de un programa se usa para corroborar si se ha alcanzado una determinada condición. Si ésta se cumple (**true**) el programa ejecutará las instrucciones correspondientes que se encuentran entre llaves. Si la condición no se cumple (**false**) el programa ignorará las instrucciones que le siguen y saltará al siguiente paso.

Su sintaxis correcta es:

```
if (CONDICIÓN) {
        //Instrucciones que quiero ejecutar si la condición es "true"
        }
```

La estructura "if ... else"

La segunda opción es **"if... else"**, que significa "Si... Sino" en inglés.

Esta alternativa de la función if funciona como una bifurcación ya que el robot tendrá la instrucción de que si se cumple la condición haga una cosa y, si no la cumple, haga otra. Esto la diferencia del **if** a secas, ya que en lugar de continuar con el programa en caso de no cumplir la condición (**false**) el **else** le indica entre llaves qué hacer a continuación antes de seguir con el resto del código.

Su sintaxis correcta es:

```
if (CONDICIÓN) {
        //Instrucciones que quiero ejecutar si la condición es "true"
}
else {
//Instrucciones que quiero ejecutar si la condición es "false"
}
```

Otras funciones útiles para tu robot

A continuación encontrarás otras funciones que serán de gran ayuda al momento de darle instrucciones al robot.

constrain (restringir)

Fuerza a un número a permanecer dentro de ciertos límites. Es decir, si el valor es mayor o menor a alguno de los límites especificado por parámetro (a y b), entonces se igualará el valor al límite que haya sobrepasado. Si por el contrario, el número está dentro de los límites, la función devolverá el mismo valor.

Su sintaxis correcta es:

constrain(x, a, b);

Donde sus parámetros son:
x: el número a limitar **a:** el límite inferior **b**: el límite superior

¿Cómo trabaja entonces la función constrain?
Si el valor de **x** se encuentra entre el rango determinado por **a** y **b**, **x** conserva su valor. Sin embargo, si el valor de **x** es menor a **a** (**x < a**) o mayor a **b** (**x > b**), **x** adoptará el valor del límite más cercano; es decir que, si el valor de **x** es menor a **a**, adoptará el valor de **a**, y si su valor es mayor a **b**, pasará a tener el valor de **b**.

Dicho de otro modo:

x > a && x < b x conserva su valor.
x < a x adopta el valor de a, ya que es menor al mínimo especificado.
x > b x adopta el valor de b, ya que es mayor al máximo especificado.

Ejemplo:
tVal = constrain(tiempoPulso, 1140, 5700);

si **tiempoPulso** es **> 1140 y < 5700**, entonces **tVal = tiempoPulso;**
si **tiempoPulso < 1140** , entonces, **tVal = 1140;**
si **tiempoPulso > 5700**, entonces **tVal = 5700;**

map (mapeo de valores)

Esta función asigna a un número que está dentro del **rango** "**a**" (definido por los parámetros **a1** y **a2**) un valor proporcional o equivalente dentro de un **rango "b"** (definido por **b1** y **b2**).

Su sintaxis es: **map (valor, a1,a2,b1,b2);**

Donde sus parámetros son:
valor: el número a mapear
a1: límite inferior del rango actual del valor **a2**: límite superior del rango actual del valor
b1: límite inferior del rango deseado **b2**: límite superior del rango deseado

Es decir que si el valor inicial coincide con el límite inferior del rango **"a" (a1)** , entonces la función devolverá el mismo límite del rango **"b" (b1)**; si coincide con el límite superior del rango **"a" (a2)**, devolverá también el límite superior del rango **"b" (b2)**; y si el valor es algún número cualquiera entre **a1** y **a2**, devolverá un valor equivalente que esté entre **b1** y **b2**. Esto es lo que se denomina mapear un número.

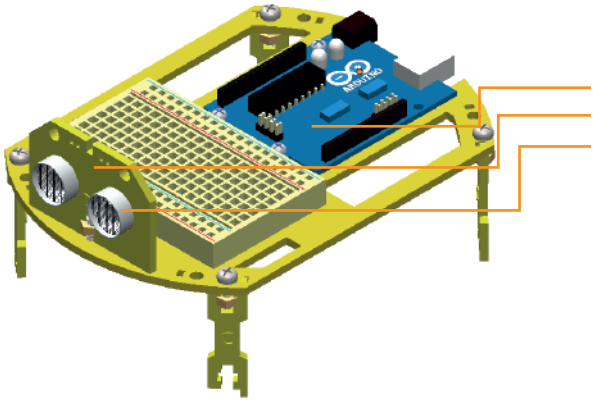
¡IMPORTANTE! Cabe aclarar que la función no restringe los valores si estos no están dentro de los rangos, ya que estos valores a veces pueden ser útiles. La función constrain () se puede usar antes o después de esta función, si se desean limitar los valores para estar dentro de los intervalos.

Ejemplo:
intensidad = map(tVal, 1140, 5700, 0, 255);

si **tval = 1140**, entonces **intensidad = 0** // límites inferiores de ambos rangos
si **tval = 5700**, entonces **intensidad = 255** // límites superiores de ambos rangos
si tval = **3420**, entonces **intensidad = 127** // valores medios de ambos rangos

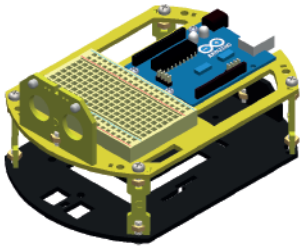
Ensamble

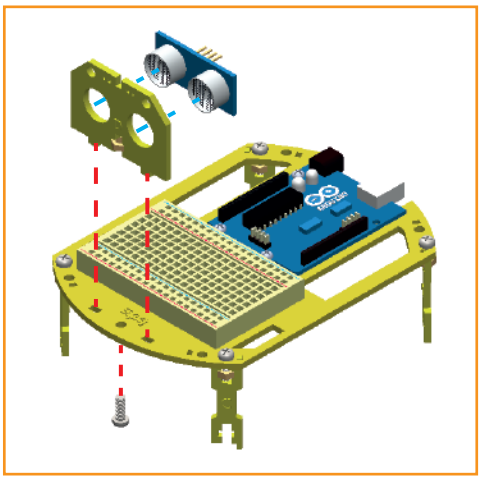
A continuación veremos la estructura y componentes necesarios encender los leds RGB.



Materiales:
-Estructura de Hola Mundo
-Pieza D
-Sensor ultrasónico

NOTA: Podemos usar la estructura de Luces para mi robot, no hay falta que desarmes el robot. Si cambiaremos la electrónica.

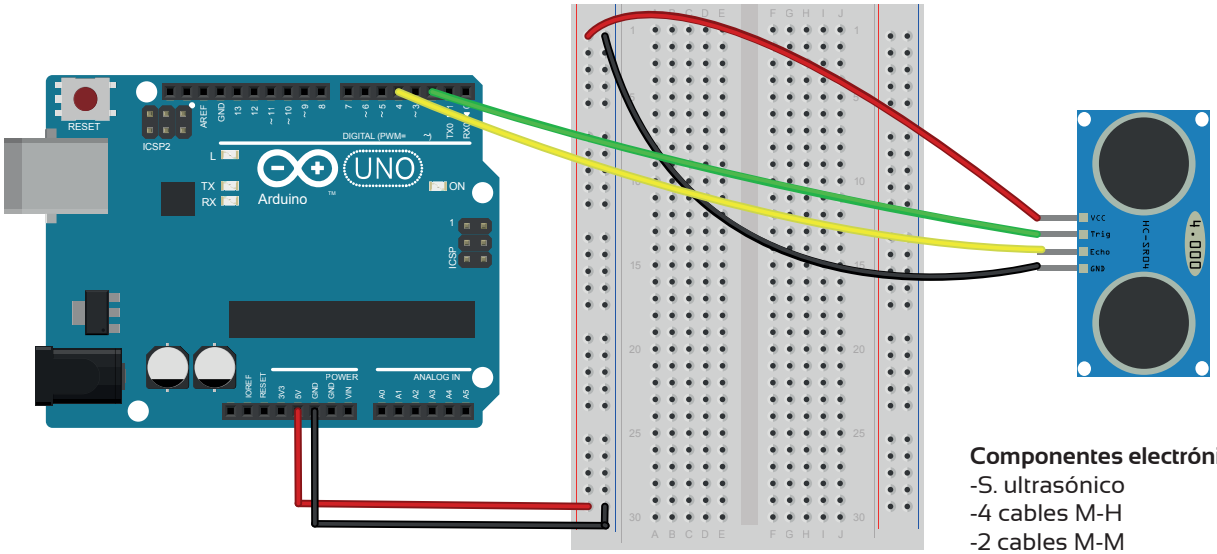




Paso 1:
Colocar el sensor ultrasónico en la pieza D, y ubicar la pieza D en el chasis A.

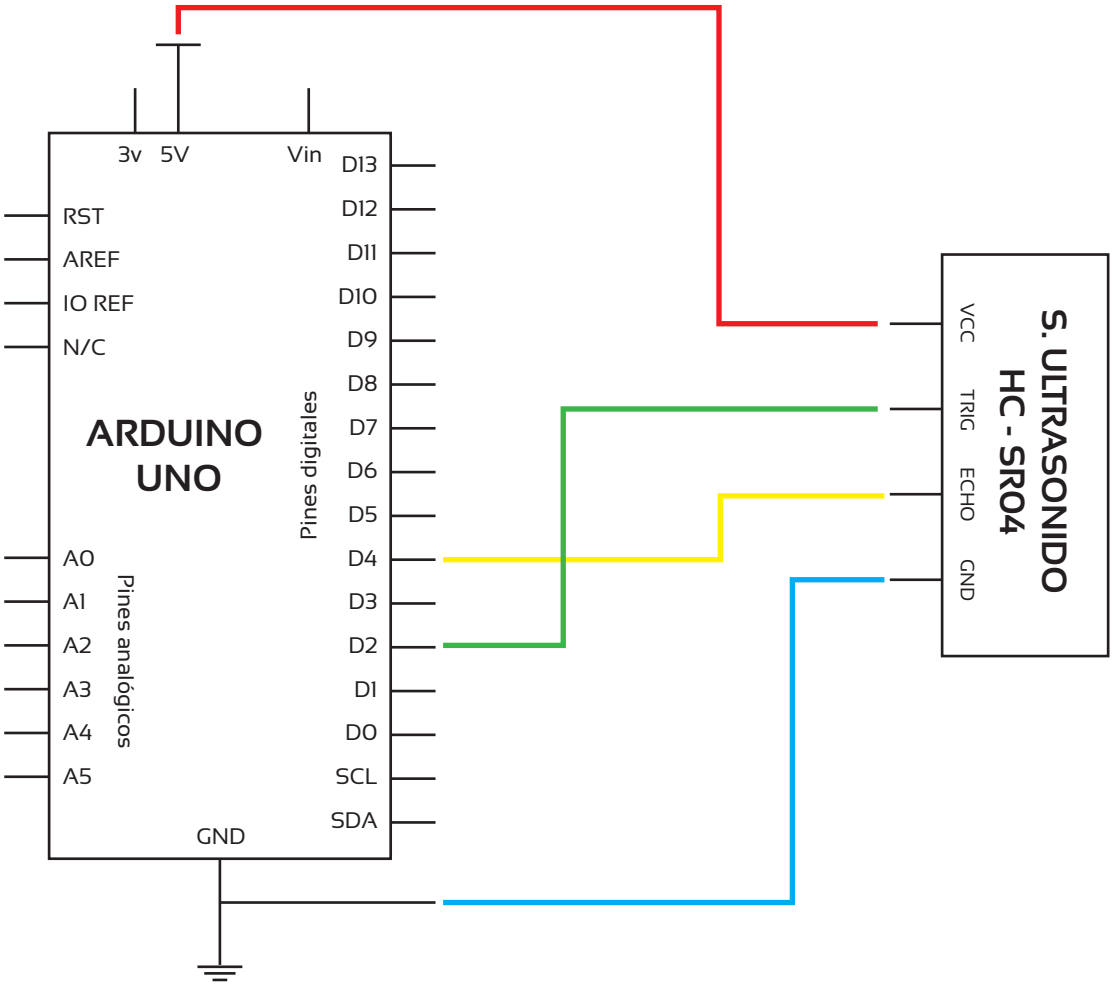
Circuito electrónico

Finalmente observamos las conexiones de nuestro circuito electrónico.



Componentes electrónicos:
-S. ultrasónico
-4 cables M-H
-2 cables M-M

Circuito electrónico en símbolos



Código de programación

Realizaremos un experimento puerto serie, el primero de ellos, donde usaremos el sensor ultrasónico para disparar y recibir onda y ver el resultado en el monitor serie.

```
1  int disparo = 2;                          //INICIO VARIABLES: Vamos a asignar a la variable "disparo" el pin 2  que se comportara como salida. El diparo enviara la señal para que se genere un ultrasonido. Para recibir la información definimos la variable "echo" en el pin 4, que se comportará como entrada. Dependiendo de la distancia el valor en alto de esta entrada va a variar. Para ello vamos a definir dos variables mas del tipo de dato long
2  const int echo = 4;
3  int ledTest = 13;
4
5
6
7
8
9
10
11 long tiempoPulso;                          //Inicio variables temporales: Definimos las variables que van a guardar los datos del sensor y procesarlos.
12 float dstMedida;
13
14
15 void setup() {
16   Serial.begin(9600);                     //Con Serial.begin(9600)  habilitamos la entrada y salida de datos por el puerto serial para poder comunicarnos con otros dispositovs, o bien verlos en el monitor serie de la PC (que es lo que vamos a hacer en este caso).
17
18
19
20
21
22   pinMode(ledTest, OUTPUT);               //INICIO SETUP ENTRADAS / SALIDAS: Configuramos los pines que necesitamos como salidas y entradas utilizando las variables declaradas anteriormente
23   pinMode(disparo, OUTPUT);
24   pinMode(echo, INPUT);
25
26   digitalWrite(disparo, LOW);             Una vez realizado esto, nos aseguramos que están apagadas las salidas. Para encender y apagar las salidas utilizaremos las constantes HIGH y LOW respectivamente.
27
28
29
30
31   digitalWrite(ledTest, HIGH);            //Saludo inicial
32   delay(500);
33   digitalWrite(ledTest, LOW);
34   delay(500);
35   digitalWrite(ledTest, HIGH);
36   delay(500);
37   digitalWrite(ledTest, LOW);
38   delay(500);
39   digitalWrite(ledTest, HIGH);
40   delay(500);
41   digitalWrite(ledTest, LOW);
42   delay(500);
43 }                                          //Cierre de setup de línea 15
44
45 void loop() {
46
47   digitalWrite(disparo, HIGH);            //Disparo de la señal: envío de pulso ultrasonico.
48   delayMicroseconds(10);                  //Hace una pausa en el programa por la cantidad de tiempo (en microsegundos) especificado como parámetro hay un millón de microsegundos en un segundo.
49
50
51
52 digitalWrite(disparo, LOW);               //Disparo de la señal: apaga el pulso ultrasonico.
```

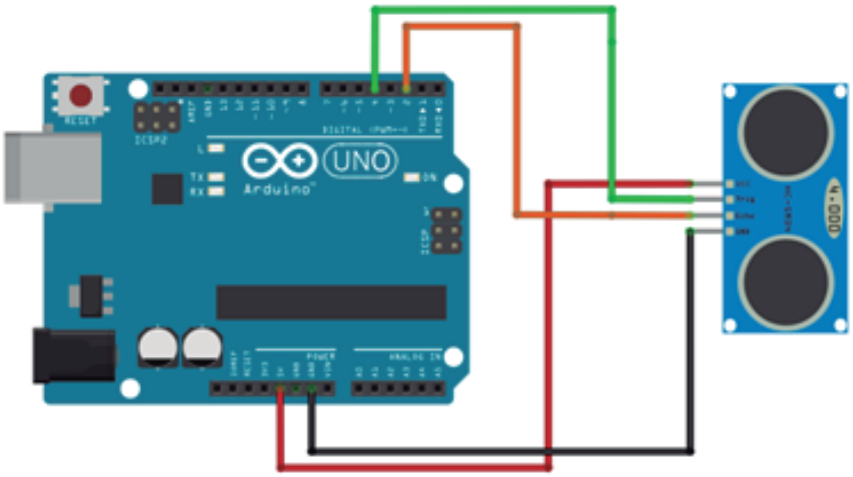
```
53    tiempoPulso = pulseIn(echo, HIGH);          //Medimos la longitud del puso entrante la utiliza-
54                                                mos para medir el tiempo que transcurrido entre el
55                                                envío  del pulso ultrasónico y cuando el sensor
56                                                recibe el rebote, es decir: desde que el pin "echo"
57                                                empieza a recibir el rebote, HIGH, hasta que deja de
58                                                hacerlo,  LOW, la longitud del pulso entrante. El
59                                                resultado lo da en microsegundos.
60
61   dstMedida = 0.0340 / 2 * tiempoPulso;        //Fórmula para calcular la distancia obteniendo un
62                                                valor flotante 340 m/s = 0.034 cm/us;  340 m/s es
63                                                la velocidad del sonido, este parámetro lo podemos
64                                                ajustar ya que depende de la densidad del aire.
65
66    Serial.println("Tiempo / Distancia ");      //Serial.println se usa para mostrar la información
67                                                pasada por parametro y continuaren la linea de
68                                                abajo, como si apretarámos "Enter".
69
70    Serial.print(tiempoPulso);                  //Serial.print se usa para mostrar en el monitor
71                                                serial, sin bajar de linea, es decir que la informacion
72                                                que enviemos en la proxima instruccion serial.print
73                                                se escribira a continuaciónde la anterior, en la
74                                                misma linea.
75   Serial.print(" / ");
76    Serial.println(dstMedida);
77
78
79    delay(500);
80   }                                            //Cierre de loop de línea 45
```

MATERIAL COMPLEMENTARIO

Testing de sensor

Sensor ultrasónico

Para probar este sensor debes armar el circuito como muestra la siguiente imagen, conectando el arduino a la PC.



Una vez que tengas listo el ensamble, abre el IDE y pega el siguiente código. Ejecuta el programa de prueba y observa el monitor serie. Si colocamos la mano frente al sensor y la acercamos o alejamos, la distancia mostrada en el monitor serie debe variar, esto es lo que indicará que el sensor ultrasónico funciona correctamente.

Código para probar el sensor ultrasónico (copiar y pegas en IDE Arduino)

```
int disparo =4;
const int echo = 2;
int ledTest = 13;
long tiempoPulso;
float dstMedida;

void setup() {
  Serial.begin(9600);
  pinMode(ledTest, OUTPUT);
  pinMode(disparo, OUTPUT);
  pinMode(echo, INPUT);
  digitalWrite(disparo, LOW);
}

void loop() {
  digitalWrite(disparo, HIGH);
  delayMicroseconds(10);
  digitalWrite(disparo, LOW);
  tiempoPulso = pulseIn(echo, HIGH);
  dstMedida = 0.0340 / 2 * tiempoPulso;
  Serial.println("Tiempo / Distancia");
  Serial.print(tiempoPulso);
  Serial.print(" / ");
  Serial.println(dstMedida);
  delay(500);
}
```

ACTIVIDADES Y EJERCITACIÓN

Cap. 6: Actividades complementarias

En base a los contenidos vistos en el **"Capítulo 6: Sensor ultrasónico"** realiza las siguientes actividades complementarias.

Actividad 1

Modifica el código utilizado en este capítulo de manera tal que muestre una línea de caracteres "*" proporcional a la distancia en que coloques la mano del sensor.

```
int disparo = 2;
const int echo = 4;

long tiempoPulso;
float dstMedida;

void setup() {
  Serial.begin(9600);
  pinMode(disparo, OUTPUT);
  pinMode(echo, INPUT);

  digitalWrite(disparo, LOW);
}

void loop() {
  digitalWrite(disparo, HIGH);
  delayMicroseconds(10);
  digitalWrite(disparo, LOW);

  tiempoPulso = pulseIn(echo, HIGH);

  dstMedida = 0.0340 / 2 * tiempoPulso;

  Serial.println("Tiempo / Distancia");
  Serial.print(tiempoPulso);
  Serial.print(" / ");
  Serial.println(dstMedida);

  delay(500);
}
```

Actividad 2

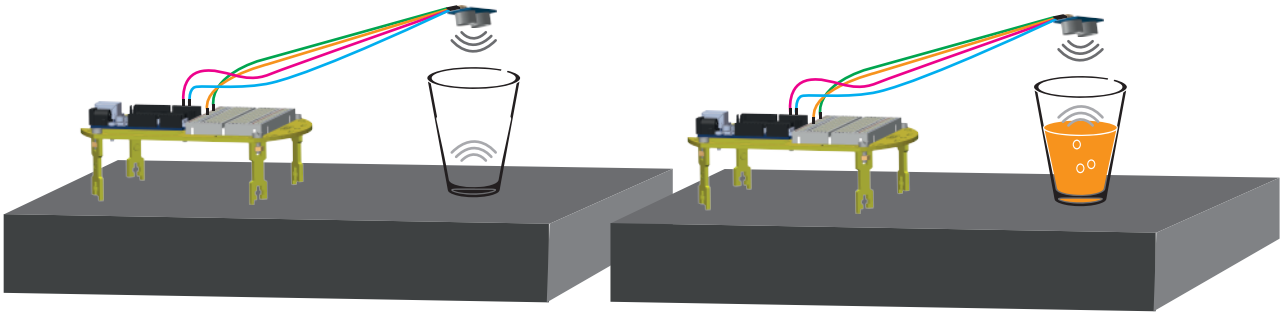
Desafío práctico: crear un medidor de nivel.

Para este experimento vamos a necesitar un vaso o un contenedor cualquiera. Primero vamos a medir la distancia que devuelve el sensor cuando éste está vacío. Luego llenamos el contenedor hasta el nivel que consideremos máximo (no necesariamente el tope) y medimos nuevamente la distancia con ayuda del sensor. A continuación hacemos la resta entre ambos valores obtenidos, el resultado es nuestra referencia de capacidad, el 100%.

capacidad = distanciaVacío - distanciaLleno

Ahora llenamos el contenedor hasta un determinado nivel, medimos la distancia, y nuevamente restamos este valor de la distancia obtenida cuando estaba vacío. Con este dato, hacemos una regla de tres para conocer el nivel al que se llenó nuestro contenedor.

nivel = distanciaVacío - distanciaActual
Porcentaje lleno = nivel * 100% capacidad



* Las ilustraciones son esquemáticas.

Consideraciones:

· **Cuando el contenido sea sólido**, por lo general se formará una montaña en el lugar donde estemos vertiendo el material. Para obtener el nivel real, sacudir un poquito el recipiente para **nivelar** el contenido.

· **Si usamos agua** no tendremos el problema anterior, pero vamos a tener que esperar un poco hasta que se estabilice la superficie. En este caso, **podemos aumentar el delay para que tome muestras más espaciadas**.

· Con este medidor de nivel podemos realizar otros experimentos similares:
Dificultad fácil: medir el porcentaje de llenado del recipiente.
Dificultad media: si conocemos la cantidad exacta del líquido o material de relleno en alguna unidad de medida específica (por ejemplo, sabemos que el nivel que representa el 100% del contenedor lleno son 100 gramos de harina) entonces podemos usar una regla de tres para calcular los gramos que contiene en base al porcentaje de llenado sensado.

Actividad 3

En la unidad 6 vimos que existen diferentes **tipos de datos** por lo que un mismo valor puede ser expresado de diferentes maneras según el tipo de dato que le asignemos.

Por ejemplo: int b= 79;

Serial.print (b, DEC); // imprime la cadena "79"
Serial.print (b, BIN); // imprime la cadena "1001111" que equivale a 79 en binario.

a. ¿Cómo se expresa el número 103 en binario (BIN)?

b. ¿A qué caracter del código ASCII corresponde el valor 54? ¿Y el 76?

Cap. 6: Respuestas

Actividad 1

```
int disparo = 2;
const int echo = 4;

long tiempoPulso;
float dstMedida;

void setup() {
  Serial.begin(9600);
  pinMode(disparo, OUTPUT);
  pinMode(echo, INPUT);

  digitalWrite(disparo, LOW);
}

void loop() {
  digitalWrite(disparo, LOW);
  delayMicroseconds(5);
  digitalWrite(disparo, HIGH);
  delayMicroseconds(10);
  digitalWrite(disparo, LOW);

  tiempoPulso = pulseIn(echo, HIGH);

  dstMedida = 0.0340 / 2 * tiempoPulso;

  for(int i=0; i<=dstMedida; i++){
    Serial.print("*");
    }
  Serial.println("*");
  delay(50);
}
```

Actividad 2

```
int disparo = 2;
const int echo = 4;
long tiempoPulso;
float dstMedida;

const int valVacio = 12;  // medición con el recipiente vacío.
const int capacidad = 7;  // medición del recipiente vacío, menos medición con el recipiente al 100%

float nivel = 0;
float porcentaje = 0;

void setup() {
  Serial.begin(9600);
```

```
  pinMode(disparo, OUTPUT);
  pinMode(echo, INPUT);
  digitalWrite(disparo, LOW);
}

void loop() {
  digitalWrite(disparo, LOW);
  delayMicroseconds(5); //se pone en bajo y 5 ms para lograr un disparo mas limpio después.
  digitalWrite(disparo, HIGH);
  delayMicroseconds(10);
  digitalWrite(disparo, LOW);

  tiempoPulso = pulseIn(echo, HIGH);
  dstMedida = 0.0340 / 2 * tiempoPulso;

  nivel = valVacio - dstMedida; //el valor de nivel no está en ninguna unidad. Solo es referencial.
  porcentaje = nivel * 100 / capacidad;
  Serial.print("El recipiente está ");
  Serial.print(porcentaje);
  Serial.println("% lleno");

  delay(500);
}
```

Este experimento es personal por lo que los valores sensados variarán en cada caso.

Actividad 3

a. ¿Cómo se expresa el número 103 en binario (BIN)?

103 = 1100111 en binario

b. ¿A qué caracter del código ASCII corresponde el valor 54? ¿Y el 76?

54 = 6
76 = L