Obligatorio 8: Sensor de distancia

IIE - Facultad de Ingeniería - Universidad de la República

Tallerine Biónico 2025

El objetivo de este obligatorio es comprender el funcionamiento del sensor de distancia. Este sensor permitiría dotar de “ojos” al “bicho”.

1. Sonido

En la figura 1 se muestra el rango de frecuencias del sonido. Observar que existen sonidos no audibles por el ser humano, por debajo de los 20 Hz y por encima de los 20 kHz. Para el taller se va a utilizar un sensor de ultrasonido que utiliza 40 kHz, motivo por el cual no es audible.

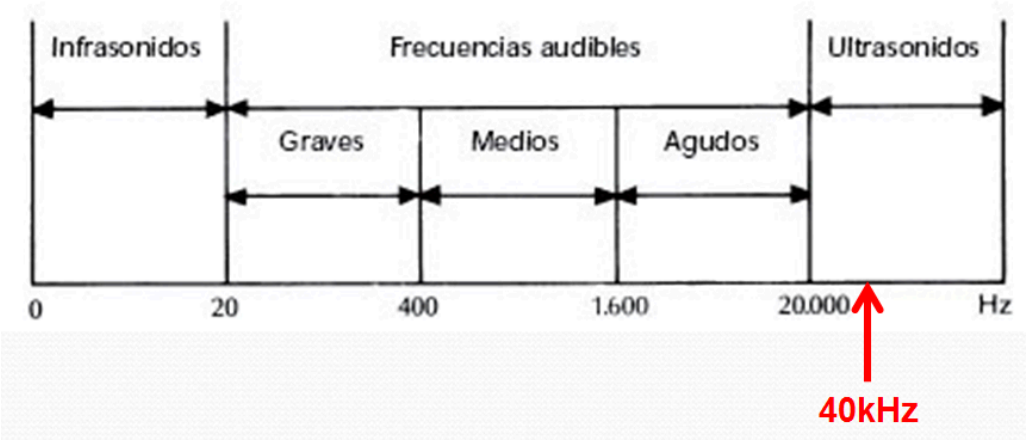

Figura 1: Rango de frecuencias del sonido. 40 kHz es la frecuencia del sensor a utilizar.

2. Sensor de Sonido

Un sensor de ultrasonido es un dispositivo que utiliza el ultrasonido para estimar la distancia a la que se encuentra un obstáculo. El sensor dispone de un transmisor y un receptor. Su principio de funcionamiento es el siguiente:

- El transmisor emite un sonido
- Luego de un tiempo, el receptor recibe el sonido reflejado (eco).
- A partir de dicho tiempo es posible calcular la distancia al obstáculo.

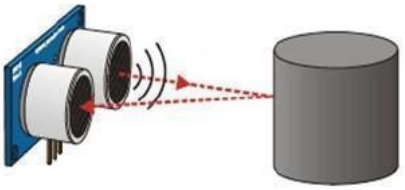

Figura 2: Sensor de ultrasonido. Representación del funcionamiento.

Tiempo = 2.(*Distancia/Velocidad*) → *Distancia* = *Tiempo.Velocidad/*2

Velocidaddelsonido = 1*/*29*cm/µs* → *Distancia* = *Tiempo/*58

En el curso se va a utilizar el sensor HC-SR04 (figura 3). Este sensor tiene las siguientes características:

- Rango: 2 cm a 400 cm
- Resolución: 3mm
- Ángulo de visión: 15°
- Frecuencia: 40 kHz
- Señal de disparo de 10*µs*

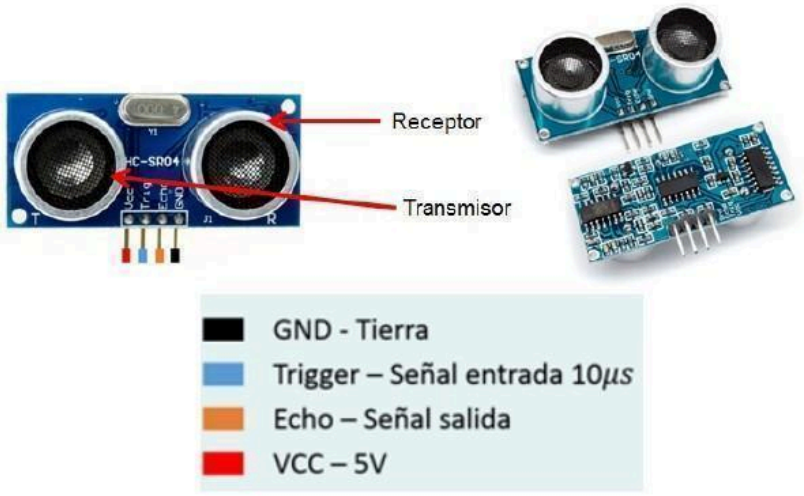

Figura 3: Sensor de ultrasonido HC-SR04.

3. Conexionado

En la conexión se debe tener en cuenta que el sensor HC-SR04 necesita tener una alimentación de 5V, por lo tanto se alimentará mediante los pines SVi.
La conexión a la placa micro:bit a través de la placa Kitronik se realiza de la siguiente forma:

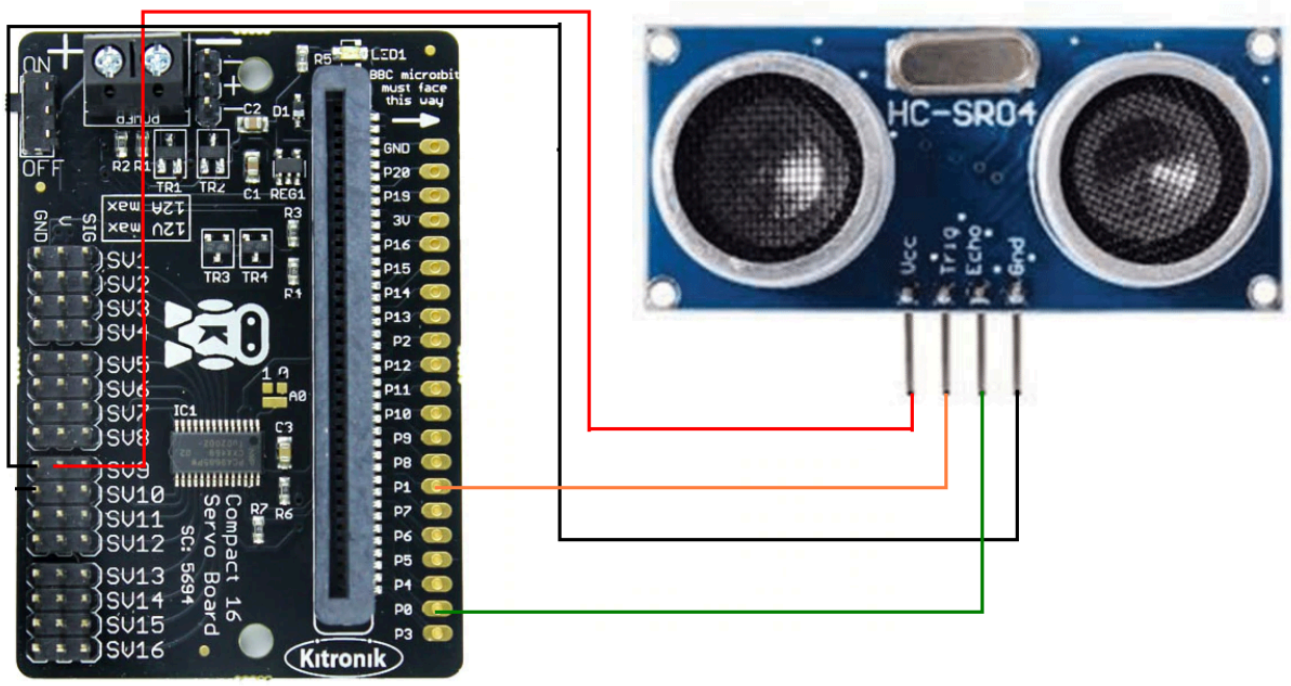

Figura 4: Conexionado entre el sensor de ultrasonido HC-SR04 y la placa Kitronik.

Placa Kitronik	Sensor HC-SR04
V	VCC
P0	ECHO
P1	TRIG
GND	GND

Aclaración: Para la conexión a través de la placa **Kitronik** *(servo driver)* recordar que los pines 19 y 20 de la placa Kitronik son para la comunicación I2C, por esta razón no se podrán utilizar para la conexión del sensor. Para que los pines de la placa Kitronik y la microbit coincidan se debe insertar la microbit con la orientación correcta.

4. Librerías y funciones

Para utilizar el sensor en un programa se dispone de la librería **NewPing**. Al igual que con los servos, se debe definir una variable objeto de la siguiente forma:

```
NewPing misonar(TRIGGER PIN, ECHO PIN, [MAX DISTANCE]);
```

misonar: es la variable objeto (nombre a elección).
TRIGGER PIN: Pin donde se conecta la salida
ECHO PIN: Pin donde se conecta la entrada.
MAX DISTANCE: Máxima distancia en cm (opcional). Sirve para minimizar el tiempo de timeout en caso de que no se detecten objetos.

Algunas funciones asociadas

```
misonar.ping() // Obtiene una medida en s
misonar.ping_median(n) // Obtiene el promedio de n medidas en s
// por defecto n = 5.
misonar.ping_cm() // Obtiene una medida en cm
misonar.convert_cm(Time) // Convierte un valor Time de s a cm
```

Para más detalles: https://bitbucket.org/teckel12/arduino-new-ping/wiki/Home.

Se presenta un ejemplo sencillo **sonar_microbit.ino** (disponible en EVA) que mide una distancia a un objeto y la imprime en el monitor serie en cm.

```
#include <NewPing.h>

/*Aqui se definen costantes a utilizar */
#define TRIGGER_PIN  1
#define ECHO_PIN     0
#define MAX_DISTANCE 200

/*Crear el objeto de la clase NewPing*/
NewPing mi_sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup() {
  // esto se ejecuta una vez cuando se enciende o resetea la Microbit
  Serial.begin(9600); // define velocidad de comunicaci n serial

  Serial.println("microbit pronta para comenzar!"); // mensaje de inicio

}

void loop() {
  // Esperar 1 segundo entre mediciones
  delay(1000);
  // Obtener medicion de tiempo de viaje del sonido y guardar en variable uS
  int tiempo = mi_sonar.ping_median();
  // Imprimir la distancia medida a la consola serial
  Serial.print("Distancia: ");
  // Calcular la distancia con base en una constante
  Serial.print(tiempo / 58);
  Serial.println("cm");
}
```

Sketch 1: Ejemplo Sonar

Nota: El texto a continuación de "//" son comentarios para aclarar lo que realiza el código. Este texto no es considerado como parte del programa que luego va a ser ejecutado por la microbit. Es una muy buena práctica (y en el curso es obligación) utilizar estos comentarios para ir aclarando lo que se va haciendo.

5. Función millis()

La función `millis()` devuelve el número de milisegundos que han pasado desde que se está ejecutando el programa actual. No requiere parámetros.
El valor que devuelve es de tipo `unsigned long` (entero entre 0 y 4294967295).
Si se ejecuta la función `millis()`, guardando su valor, y luego de un tiempo se vuelve a ejecutar `millis()` nuevamente, la diferencia entre los 2 valores devueltos indica el tiempo transcurrido en milisegundos entre una ejecución y otra.

6. Ejercicios

Ejercicio 1

1. En el siguiente ejercicio se deben realizar los siguientes pasos:

    a. Cargar en la placa el código **`sonar_microbit.ino`**.
    b. Realizar el conexionado del sensor de ultrasonido.
    c. Probar el código en la placa.
    d. Utilizando la matriz de LEDs de la placa, modificar el código de forma que:
        - Si la distancia al objeto es menor a 20 cm encender un LED de la matriz indicando detección de obstáculo.
        - Si la distancia al objeto es mayor o igual a 20 cm el LED debe ser apagado.
        - Si la distancia al objeto es 0 encender de manera intermitente el LED indicando fuera de rango.

Ejercicio 2

A. Estudiar la función `millis()`.
B. Estudiar la función `retardo(int t)` (disponible en EVA) y verificar que se comporta del mismo modo que la función `delay(t)` ya conocida.
C. Implementar una prueba de la función creada utilizando el ejercicio de los tres leds que encienden en forma secuencial del Obligatorio 1 sustituyendo `delay(t)` por la función `retardo(t)`.

Ejercicio 3

Disponer de la función `retardo(t)` va a permitir realizar otras tareas mientras se espera que finalice el tiempo de espera.

- Sustituir en el código del "bicho" la función `delay(t)` por `retardo(t)`.
- Integrar en `retardo(t)` el código del Ejercicio 1, para que mientras realice un movimiento pueda ir detectando si hay objetos a su alrededor.