Sensor obstáculos IR

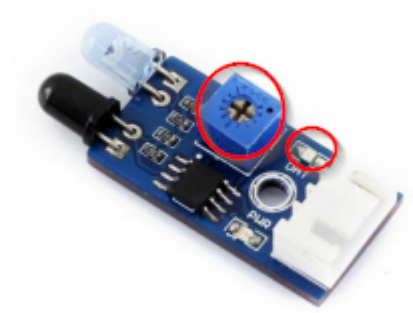
4 Sensor obstáculos IR

4.1 ¿Cómo funciona?

Lo primero que tenemos que hacer es ajustar su sensibilidad con el potenciómetro que tiene de tal manera que detecte de forma correcta un obstáculo:

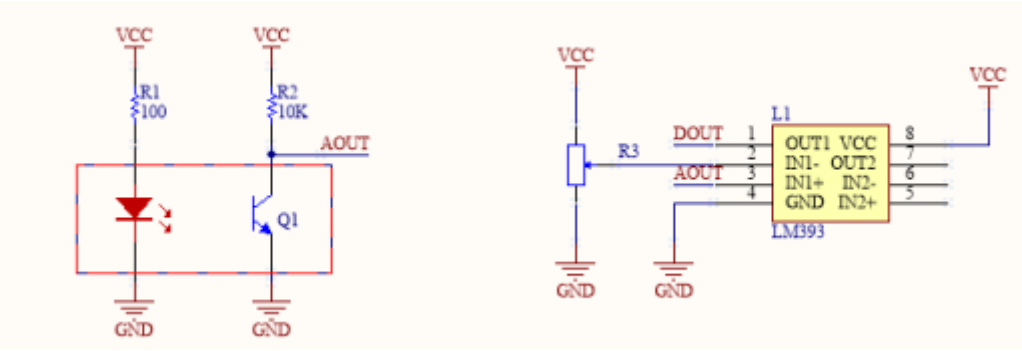
https://www.youtube.com/embed/_89J0WtOqgQ

ES MUY SENSIBLE el punto está justo cuando se apaga:



https://www.youtube.com/embed/aJonvDOttgU

Tiene un led de infrarrojos que cuando hay un obstáculo delante del sensor, refleja esta radiacción y el otro LED (realmente no se podría decir que es LED pues no emite, se tendría que llamar unión PN semiconductor), es un sensor receptor que lo detecta. Un chip comparador LM393 detecta la señal y su sensibilidad se ajusta con el potenciómetro:



La salida DOUT de cada LM393 están conectados a los siguientes puertos GPIO:

- Derecha GPIO 16
- Izquierda GPIO 19

Una vez más vemos que la resistencias del sensor están tipo PULL-UP luego cuando detecta emitirá un 0 lógico y cuando no detecta emitirá un 1 lógico.

4.2 Test

Ejecutamos este pequeño programa:

Fichero 4-2-TestObstaculoIR.py

```
import RPi.GPIO as GPIO
import time

DR = 16
DL = 19

GPIO.setmode(GPIO.BCM)

GPIO.setup(DR,GPIO.IN)
GPIO.setup(DL,GPIO.IN)

for i in range(10000):
    print('\nSensor derecha :',GPIO.input(DR))
    print('\nSensor izquier :',GPIO.input(DL))
```

Y podemos ver en el vídeo que emite un 0 cuando detecta un obstáculo:

https://www.youtube.com/embed/oehMTYNSPHA

4.3 Variables.py

Añadimos más variables a VARIABLES.py

Los marcados en negrita:

---

import RPi.GPIO as GPIO

#####MOTORES

IN1=12 IN2=13 ENA=6 IN3=20 IN4=21 ENB=26

##SENSOR VELOCIDAD MOTORES

DataMotorR = 7 DataMotorL = 8

##########SENSOR OBSTACULOS IR DR = 16 DL = 19

########CONFIGURACION GPIO ENTRADAS SALIDAS

GPIO.setmode(GPIO.BCM) GPIO.setwarnings(False)

####################MOTORES

GPIO.setup(IN1,GPIO.OUT) GPIO.setup(IN2,GPIO.OUT) GPIO.setup(IN3,GPIO.OUT)
GPIO.setup(IN4,GPIO.OUT) GPIO.setup(ENA,GPIO.OUT) GPIO.setup(ENB,GPIO.OUT)

############## VELOCIDAD DE LOS MOTORES

PWMA = GPIO.PWM(ENA,500) PWMB = GPIO.PWM(ENB,500)

############SENSOR VELOCIDAD MOTORES

GPIO.setup(DataMotorR,GPIO.IN) GPIO.setup(DataMotorL,GPIO.IN)

####################SENSORES OBSTACULOS IR GPIO.setup(DR,GPIO.IN) GPIO.setup(DL,GPIO.IN)

4.4 Roomba

Bueno ahora hay que hacer el típico programa que evite los obstáculos. Ya sabes detectar los obstáculos, ya sabes controlar el movimiento ¿a qué esperas?

https://giphy.com/embed/kviKlWqjoBdCw

via GIPHY

https://www.youtube.com/embed/5LvBqHv1wM4

Solución

La solución no es única, una propuesta es hacerlo con las librerías que hemos aprendido: *

Ponemos las librerías fichero MOVIMIENTOS.py y MOVIMIENTOSPASO.py en la misma carpeta que vamos a crear este programa y las incorporamos en el programa con **import**. * También incorporamos las variables definidas en **VARIABLES.py** * Si no detecta nada, que sigua hacia delante. * Si detecta algo, según los dos o uno, que de unos pasos atrás y que gire.

%accordion%Solución%accordion%

Fichero Roomba.py

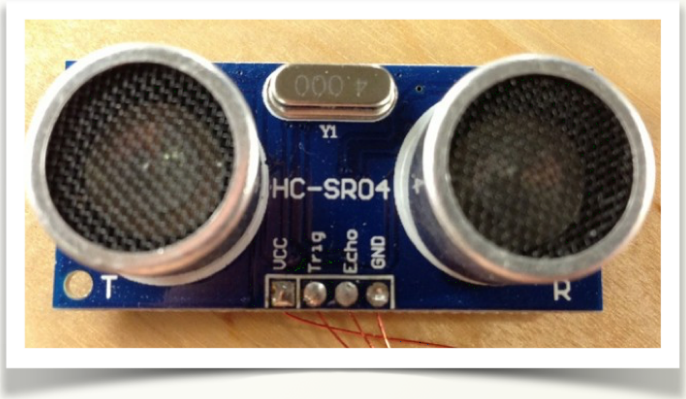
```
import RPi.GPIO as GPIO
import time
from VARIABLES import *
import MOVIMIENTOS
import MOVIMIENTOSPASO

while True:
    sensorR= not (GPIO.input(DR))
    sensorL= not (GPIO.input(DR))
    if not(sensorR and sensorL):
        MOVIMIENTOS.FORDWARD(50)
    if (sensorR and sensorL):
```

```
        MOVIMIENTOSPASO.BOTH(50,-10,50,-10)
    if (sensorR and not(sensorL)):
        MOVIMIENTOSPASO.BOTH(50,-5,50,-5)
        MOVIMIENTOSPASO.BOTH(40,-5,40,5)
    if (not(sensorR) and (sensorL)):
        MOVIMIENTOSPASO.BOTH(50,-5,50,-5)
        MOVIMIENTOSPASO.BOTH(40,5,40,-5)
```

%/accordion%

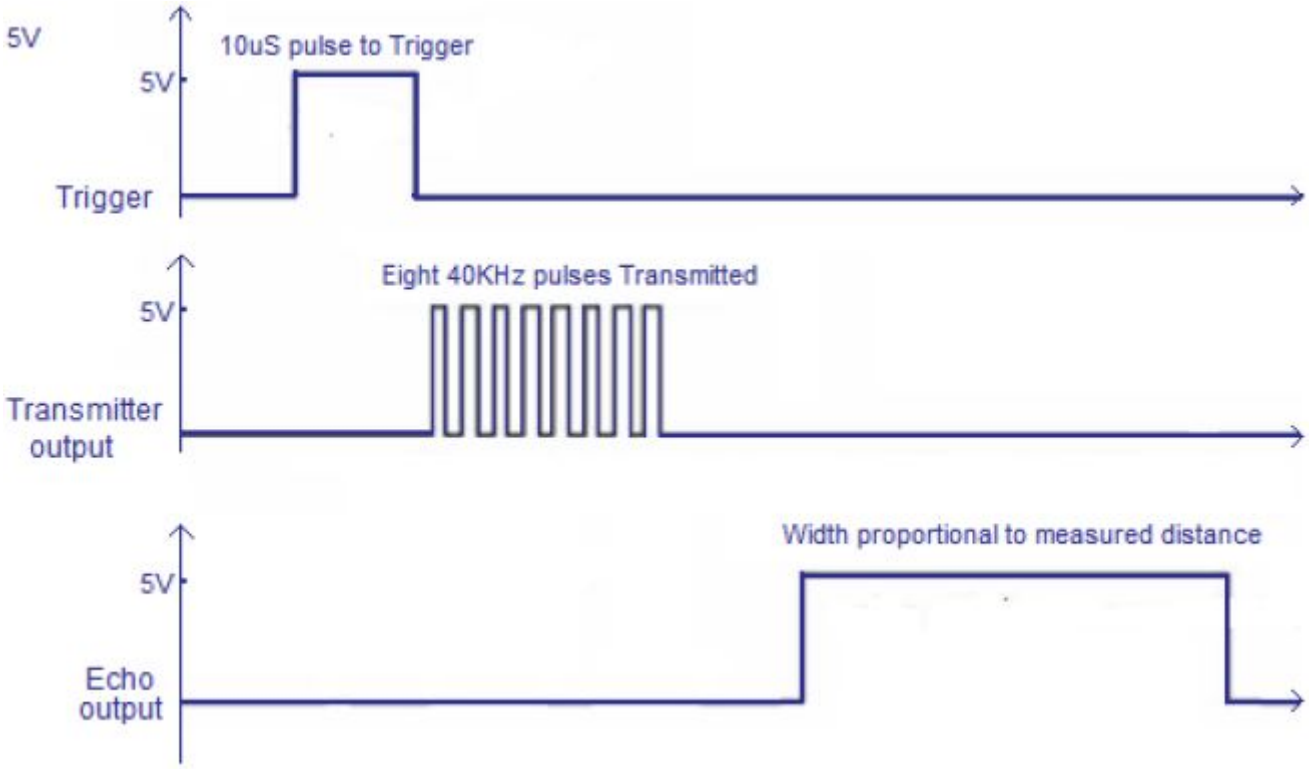
4.5 Posibilidad ultrasonidos

Se puede conseguir más precisión añadiendo un tercer sensor y mucho más preciso: El **sensor de Ultrasonidos.**



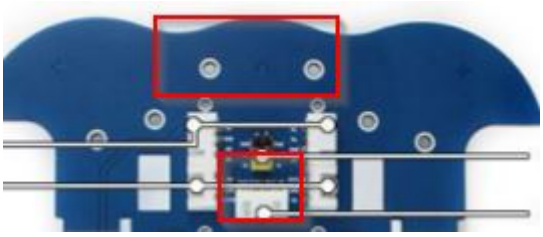
Mira en esta página para saber cómo se utiliza con el Arduino.

Básicamente se emite un pulso por el pin **Trigger**, él emite una señal de 40kHz y según el eco recibido saca por **Output** un pulso cuyo ancho es proporcional a la distancia:

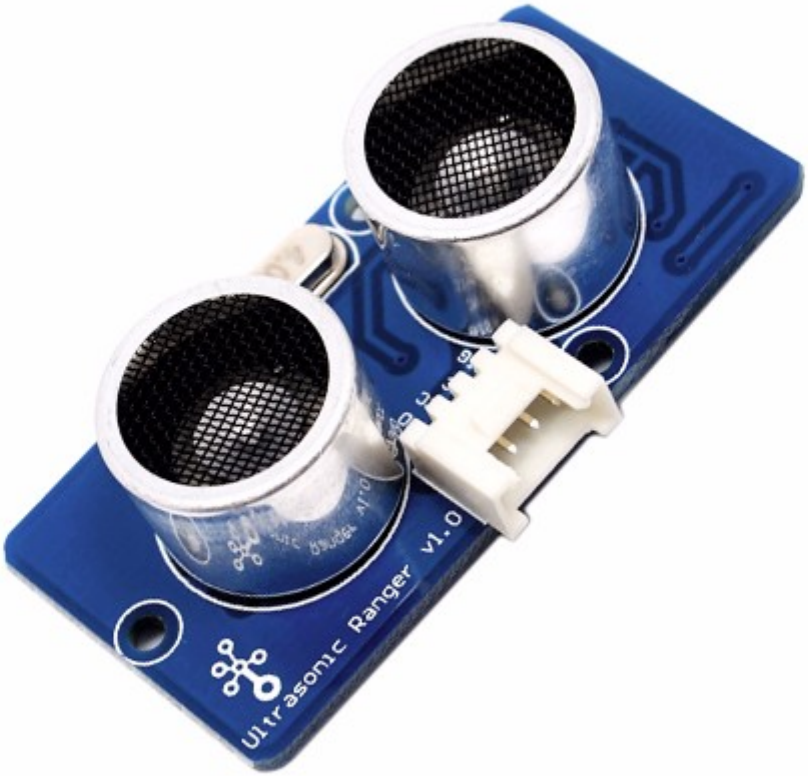


Conexión en alphabot

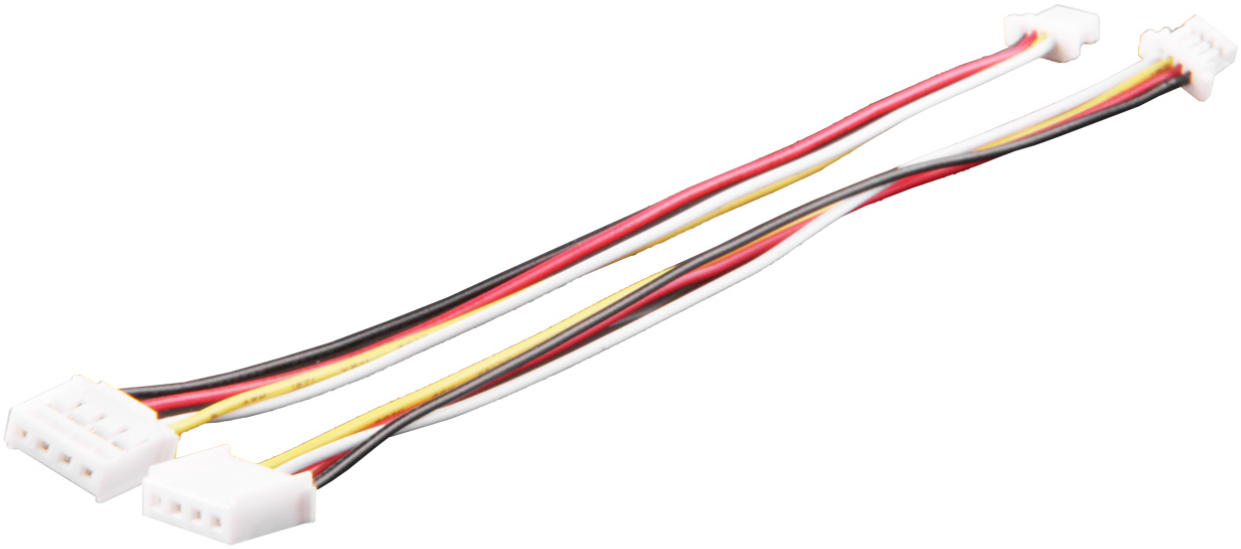
En Alphabot se conectaría los cables en el conector blanco de abajo y el ultrasonidos con unos tornillos en los dos agujeros de la parte delantera:



El sensor de ultrasonidos tiene que estar adaptado a los cables compatibles con la Shield Grove de Arduino. Por ejemplo este:



CHAPUZA : No tiene su orden standard GND-Vcc-DATA1-DATA2 sino es GND-DATA1-DATA2-VCC o sea GND-Trg-Echo-5V luego habría que hacer alguna chapucilla de intercambiar cables, habría que elegir unos cables largos, cortarlos e intercambiarlos:



Para sujetar el sensor ultrasonidos al chasis habría que comprar un soporte:

Otra opción es quitar la cámara y poner el sensor de ultrasonidos:

EL KIT DE CATEDU NO PROPORCIONA EL SENSOR ULTRASONIDOS

Bueno, si aún así me decido ponerlo ¿cómo se programa?

Muy fácil, el conector blanco de abajo está conectado con los siguientes GPIO:

- Echo en el GPIO 5
- Trigger en el GPIO 17

Por lo tanto, viendo la teoría, una posible función en código Python para utilizarlo sería:

- Emitir un pulso alto por TRIG durante 15 microsegundos.
- Esperar el pulso alto de ECHO
- Cronometrar el pulso alto de ECHO
- La distancia será velocidad por tiempo o sea: la diferencia el tiempo del pulso ECHO multiplicado por la velocidad del sonido y dividido por 2 pues es el recorrido del sonido ida y vuelta.

```
TRIG = 17
ECHO = 5

GPIO.setup(TRIG,GPIO.OUT,initial=GPIO.LOW)
GPIO.setup(ECHO,GPIO.IN)

def Distance():
    GPIO.output(TRIG,GPIO.HIGH)
    time.sleep(0.000015)
    GPIO.output(TRIG,GPIO.LOW)
    while not GPIO.input(ECHO):
        pass
```

```
    t1 = time.time()
    while GPIO.input(ECHO):
        pass
    t2 = time.time()
    return (t2-t1)*34000/2
```