06

Programando o sensor

Transcrição

Com o sensor devidamente montado no nosso carrinho, já podemos programar o código que irá calcular a distância dos objetos à frente do PiBOT. Assim como fizemos para mover o carrinho, vamos nos conectar ao Raspberry Pi e criaremos um novo arquivo, chamado **distancia.py**.

Assim como antes, vamos importar as bibliotecas, definir como referência a pinagem física do GPIO e remover as mensagens de alerta:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
```

O pinos do sensor **Echo** e **Trigger** são, respectivamente, os pinos 29 e 31 do Raspberry Pi. Então vamos defini-los:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

ECHO = 29
TRIG = 31
```

O `ECHO` recebe dados, logo é uma porta de entrada, já o `TRIG` é uma porta de saída. Vamos definir isso, dentro de uma função chamada `setup_sensor` :

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

ECHO = 29
TRIG = 31

def setup_sensor():
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.setup(TRIG, GPIO.OUT)
```

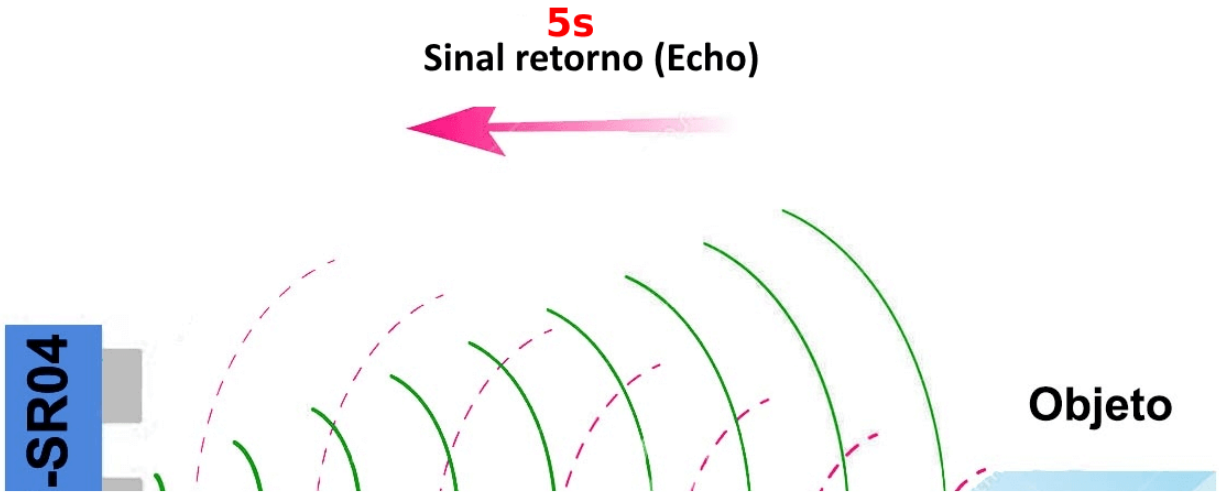
Agora iremos criar uma função chamada `roda_medicao` , que calculará efetivamente a distância dos objetos em relação ao sensor. Começaremos criando uma variável global para guardar o valor da distância e inicializando essa variável com **0**:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

ECHO = 29
TRIG = 31

def setup_sensor():
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.setup(TRIG, GPIO.OUT)

def roda_medicao():
    global distancia_cm
    distancia_cm = 0
```

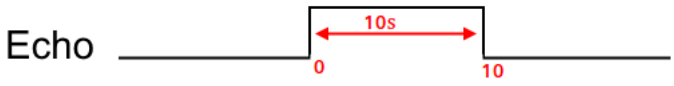
Agora, precisamos criar um laço infinito, e dentro desse laço irmos medindo as distâncias dos objetos à nossa frente. Vamos medir distâncias de 2 em 2 segundos, ou seja, já devemos começar o laço aguardando 2 segundos. Após o aguardo, vamos emitir o pulso, colocando o `TRIG` , que é um gatilho para o sensor emita o pulso, em alta. Segundo o site do fabricante (http://www.micropik.com/PDF/HCSR04.pdf), o pulso dura 10 microssegundos, ou seja, devemos aguardar 10 microssegundos e logo depois colocar o `TRIG` em baixa:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

ECHO = 29
TRIG = 31

def setup_sensor():
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.setup(TRIG, GPIO.OUT)

def roda_medicao():
    global distancia_cm
    distancia_cm = 0
    while True:
        time.sleep(2)
        GPIO.output(TRIG, GPIO.HIGH)
        time.sleep (0.000010)
        GPIO.output(TRIG, GPIO.LOW)
```

O sensor consegue medir quanto tempo os pulsos demoram para bater no objeto e voltar ao sensor. Através desse tempo que conseguiremos calcular a distância até o objeto.



Para o sensor nos informar a quantidade de tempo que o pulso passou viajando, ele emite uma onda quadrada no `ECHO`, de mesmo tempo do tempo de viagem do pulso (ida + volta):



Ou seja, para sabermos o tempo de viagem do pulso , que será usado para calcular a distância até objeto, devemos calcular o tempo em que o `ECHO` fica em `HIGH`. Para fazer isto, vamos ler a entrada e atualizar o valor do pulso inicial até que o sinal se altere de `LOW` para `HIGH`, este será nosso tempo inicial. Vamos utilizar um loop `while` para verificar quando o valor do `ECHO` se altera e salvá-lo na variável `pulso_inicial`:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

ECHO = 29
TRIG = 31

def setup_sensor():
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.setup(TRIG, GPIO.OUT)
```

```
def roda_medicao():
    global distancia_cm
    distancia_cm = 0
    while True:
        time.sleep(2)
        GPIO.output(TRIG, GPIO.HIGH)
        time.sleep (0.000010)
        GPIO.output(TRIG, GPIO.LOW)
        while GPIO.input(ECHO) == 0:
            pulso_inicial = time.time()
```

Vamos utilizar a mesma lógica para calcular o tempo que pulso varia de `HIGH` para `LOW`, que será o tempo final da onda quadrada do `ECHO`:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

ECHO = 29
TRIG = 31

def setup_sensor():
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.setup(TRIG, GPIO.OUT)

def roda_medicao():
    global distancia_cm
    distancia_cm = 0
    while True:
        time.sleep(2)
        GPIO.output(TRIG, GPIO.HIGH)
        time.sleep (0.000010)
        GPIO.output(TRIG, GPIO.LOW)
        while GPIO.input(ECHO) == 0:
            pulso_inicial = time.time()
        while GPIO.input(ECHO) == 1:
            pulso_final = time.time()
```

Com o tempo inicial e o tempo final, basta fazermos a diferença e teremos o tempo total da onda quadrada, que é o tempo que o pulso demorou para ir e voltar até o objeto:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

ECHO = 29
TRIG = 31

def setup_sensor():
    GPIO.setup(ECHO, GPIO.IN)
```

```
    GPIO.setup(TRIG, GPIO.OUT)

def roda_medicao():
    global distancia_cm
    distancia_cm = 0
    while True:
        time.sleep(2)
        GPIO.output(TRIG, GPIO.HIGH)
        time.sleep (0.000010)
        GPIO.output(TRIG, GPIO.LOW)
        while GPIO.input(ECHO) == 0:
            pulso_inicial = time.time()
        while GPIO.input(ECHO) == 1:
            pulso_final = time.time()
        duracao_pulso = pulso_final - pulso_inicial
```

Agora que temos o tempo, podemos calcular a distância. Para isso, precisamos multiplicar o tempo pela velocidade do som, **343 m/s**, ou **34300 cm/s**. Mas como a velocidade do som varia conforme a temperatura ambiente, o ideal é você utilizar o valor da tabela abaixo que mais se aproxima do seu ambiente :

Influência da temperatura do ar na velocidade do som				
ϑ em °C (K)	c em m/s	C em km/h	ρ em kg/m³	Z em N·s/m³
-30 °C (243,15 K)	312,7	1.171,4	1,438	453,4
-25 °C (248,15 K)	315,9	1.171,4	1,413	449,1
-20 °C (253,15 K)	319,1	1.171,4	1,388	444,8
-15 °C (258,15 K)	322,2	1.171,4	1,363	440,6
-10 °C (263,15 K)	325,3	1.171,4	1,339	436,5
-5 °C (268,15 K)	328,4	1.182,6	1,316	432,4
0 °C (273,15 K)	331,5	1.193,4	1,293	428,3
5 °C (278,15 K)	334,5	1,204,2	1,269	424,5
10 °C (283,15 K)	337,5	1.215,0	1,247	420,7
15 °C (288,15 K)	340,5	1.226,0	1,225	417,0
20 °C (293,15 K)	343,4	1.237,0	1,204	413,5
25 °C (298,15 K)	346,3	1.246,7	1,184	410,0
30 °C (303,15 K)	349,2	1.257,12	1,164	406,6

Sabendo a velocidade e a duração que o pulso demorou, podemos facilmente calcular a distância percorrida com simples fórmula:

```
distancia(m) = velocidade(m/s) x tempo(s)
```

Só que vale considerar que o pulso percorre **duas vezes** a distância até o objeto, pois o mesmo tem que ir até o objeto e em seguida voltar. Se queremos apenas a distância até objeto, devemos dividir o tempo pela metade, ficando:

```
distancia(m) = velocidade(m/s) x (tempo(s) / 2)
```

Como queremos a distância em *cm*, vamos utilizar a velocidade em **cm/s** em vez de **m/s** .

Implementando este mero cálculo em Python, temos:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

ECHO = 29
TRIG = 31

def setup_sensor():
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.setup(TRIG, GPIO.OUT)

def roda_medicao():
    global distancia_cm
    distancia_cm = 0
    while True:
        time.sleep(2)
        GPIO.output(TRIG, GPIO.HIGH)
        time.sleep (0.000010)
        GPIO.output(TRIG, GPIO.LOW)
        while GPIO.input(ECHO) == 0:
            pulso_inicial = time.time()
        while GPIO.input(ECHO) == 1:
            pulso_final = time.time()
        duracao_pulso = pulso_final - pulso_inicial
        distancia_cm =  34300 * (duracao_pulso/2)
```

Agora podemos remover as casas decimais da distância e imprimi-la:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

ECHO = 29
TRIG = 31

def setup_sensor():
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.setup(TRIG, GPIO.OUT)

def roda_medicao():
    global distancia_cm
    distancia_cm = 0
    while True:
        time.sleep(2)
        GPIO.output(TRIG, GPIO.HIGH)
        time.sleep (0.000010)
        GPIO.output(TRIG, GPIO.LOW)
        while GPIO.input(ECHO) == 0:
            pulso_inicial = time.time()
        while GPIO.input(ECHO) == 1:
            pulso_final = time.time()
```

```
        duracao_pulso = pulso_final - pulso_inicial
        distancia_cm =  34300 * (duracao_pulso/2)
        distancia_cm = round(distancia_cm, 0)
        print(distancia_cm, 'cm   ',end="\r")
```

Na hora da impressão, utilizamos o `end="\r"` para que a nova impressão (uma nova distância é impressa a cada 2 segundos) não fique abaixo da antiga, e sim em cima, no mesmo local, assim não precisamos ficar fazendo *scroll* na tela.

Agora só não podemos nos esquecer de chamar as duas funções ao final do programa:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

ECHO = 29
TRIG = 31

def setup_sensor():
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.setup(TRIG, GPIO.OUT)

def roda_medicao():
    global distancia_cm
    distancia_cm = 0
    while True:
        time.sleep(2)
        GPIO.output(TRIG, GPIO.HIGH)
        time.sleep (0.000010)
        GPIO.output(TRIG, GPIO.LOW)
        while GPIO.input(ECHO) == 0:
            pulso_inicial = time.time()
        while GPIO.input(ECHO) == 1:
            pulso_final = time.time()
        duracao_pulso = pulso_final - pulso_inicial
        distancia_cm =  34300 * (duracao_pulso/2)
        distancia_cm = round(distancia_cm, 0)
        print(distancia_cm, 'cm   ',end="\r")

setup_sensor()
roda_medicao()
```

Com o programa pronto, no próximo vídeo iremos testar o funcionamento do sensor no nosso carrinho!