09

Mãos na Massa: Criando o script para medir a distância

Criando o script e importando as bibliotecas

Vamos começar criando o script **distancia.py**, dentro da pasta **/home/raspberry/pibot/** e começar a editá-lo.

Dentro do **distancia.py**, vamos começar importando as bibliotecas que usaremos:

```
import RPi.GPIO as GPIO
import time
```

Configurando os GPIOs

Agora, vamos configurar as portas do GPIO, como fizemos com o **controle.py**, setando a pinagem física do GPIO como referência e desabilitando os alertas:

```
import RPi.GPIO as GPIO
import time

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

Mapeando os pinos

Com essas configurações iniciais prontas, podemos iniciar o mapeamento dos pinos físicos. Vamos criar duas variáveis, `ECHO` e `TRIG`, e colocar em cada uma delas o valor de seu pino GPIO:

```
import RPi.GPIO as GPIO
import time

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

ECHO = 16
TRIG = 11
```

Com os pinos configurados, podemos criar uma função de *setup*, que será executada no início do nosso código e configurará os pinos como saída ou entrada. Nesse caso, o nosso pino `ECHO` será a entrada e o pino `TRIG` será a saída:

```
import RPi.GPIO as GPIO
import time

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

ECHO = 16
```

```
TRIG = 11

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

Iniciando a medição

Com todo esse *setup* inicial pronto, vamos começar a criar a função que irá medir a distância calculada pelo sensor. Começamos criando a função `roda_medicao()`, que conterá uma variável global que guardará o valor da distância. Vamos inicializá-la com **0**:

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

Agora vamos criar um loop infinito, que irá realizar a medição de 2 em 2 segundos:

```
def roda_medicao():
    global distancia_cm
    distancia_cm = 0
    while True:
        time.sleep(2)
```

Após o aguardar o tempo, vamos emitir o pulso, colocando o `TRIG`, que é um gatilho para o sensor emita o pulso, em alta. Segundo o site do fabricante, o pulso dura 10 microssegundos, ou seja, devemos aguardar 10 microssegundos e logo depois colocar o `TRIG` em baixa:

```
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 a distância baseando-se no tempo que o `ECHO` está em alta, ou seja, o tempo que o sensor está ***escutando*** o retorno do pulso enviado. Para medir **precisamente** o tempo que o sensor fica em alta, temos que primeiro descobrir o **exato** momento que ele sai do nível `LOW` para o nível `HIGH`. Para isso, faremos um loop `while`, que ficará salvando o tempo enquanto ele estiver em nível baixo, e só sairemos deste loop quando o nível for para `HIGH`, assim o último valor possível será salvo na variável quando ele se alterar, veja:

```
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()
```

Agora que sabemos o **exato momento** em que o pulso emitido foi detectado de volta, só temos que ficar escutando até o fim do pulso. Vamos utilizar uma lógica análoga, onde ficaremos ouvindo enquanto o `ECHO` estiver em `HIGH` e marcar o exato momento da troca do sinal de `HIGH` para `LOW`, que é a indicação que o pulso acabou e devemos medir este tempo.

```
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 os valores do `pulso_inicial` e do `pulso_final` em mãos, conseguimos calcular a sua diferença e obter o tempo que o pulso ficou viajando pelo ar:

```
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
```

Como o pulso de som viaja pelo **ar**, conseguimos calcular a distância percorrida pelo pulso fazendo uma conta matemática, pois o pulso viaja na velocidade do som, que é um valor conhecido de **343 m/s** ou **34300 cm/s**. Mas essa velocidade varia conforme a temperatura ambiente, o ideal é 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

Como o pulso tem que percorrer duas vezes a distância até o objeto (para chegar até o objeto e para voltar até o sensor), precisamos dividir essa distância por dois. Feita esta análise física, vamos calcular a distância multiplicando a velocidade do nosso pulso pelo tempo que ele demorou:

```
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)
```

Como é possível que encontremos um número quebrado, vamos utilizar a função `round()` para arredondá-lo em um número inteiro, com zero casas decimais:

```
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)
```

Agora, para imprimir este valor, vamos utilizar 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:

```
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")
```

Chamando as funções

Não podemos nos esquecer de, ao final do programa, chamar a função de *setup* e de medição:

```
setup_sensor()
roda_medicao()
```

Executando o script

Com tudo pronto, basta executar o comando no terminal:

```
python3 ~/pibot/distancia.py
```

Repare que o terminal irá travar e a cada dois segundos ele irá exibir um novo valor de distância.

Download do distancia.py

Para sua comparação, você pode baixar o arquivo completo aqui (https://s3.amazonaws.com/caelum-online-public/raspberry3/files/cap4/distancia.py) .