

Acceso aos GPIO con Python

Obxectivos

- Activar os GPIO da Raspberry Pi en Raspbian
- Valorizar o uso dos GPIO da Raspberry Pi en proxectos que integren SW e HW.
- Controlar circuítos electrónicos independentes mediante Python
- Integrar novos dispositivos electrónicos sinxelos
- Revalidar o cinto amarelo do NINJA do terminal

Acceso aos GPIO con Python

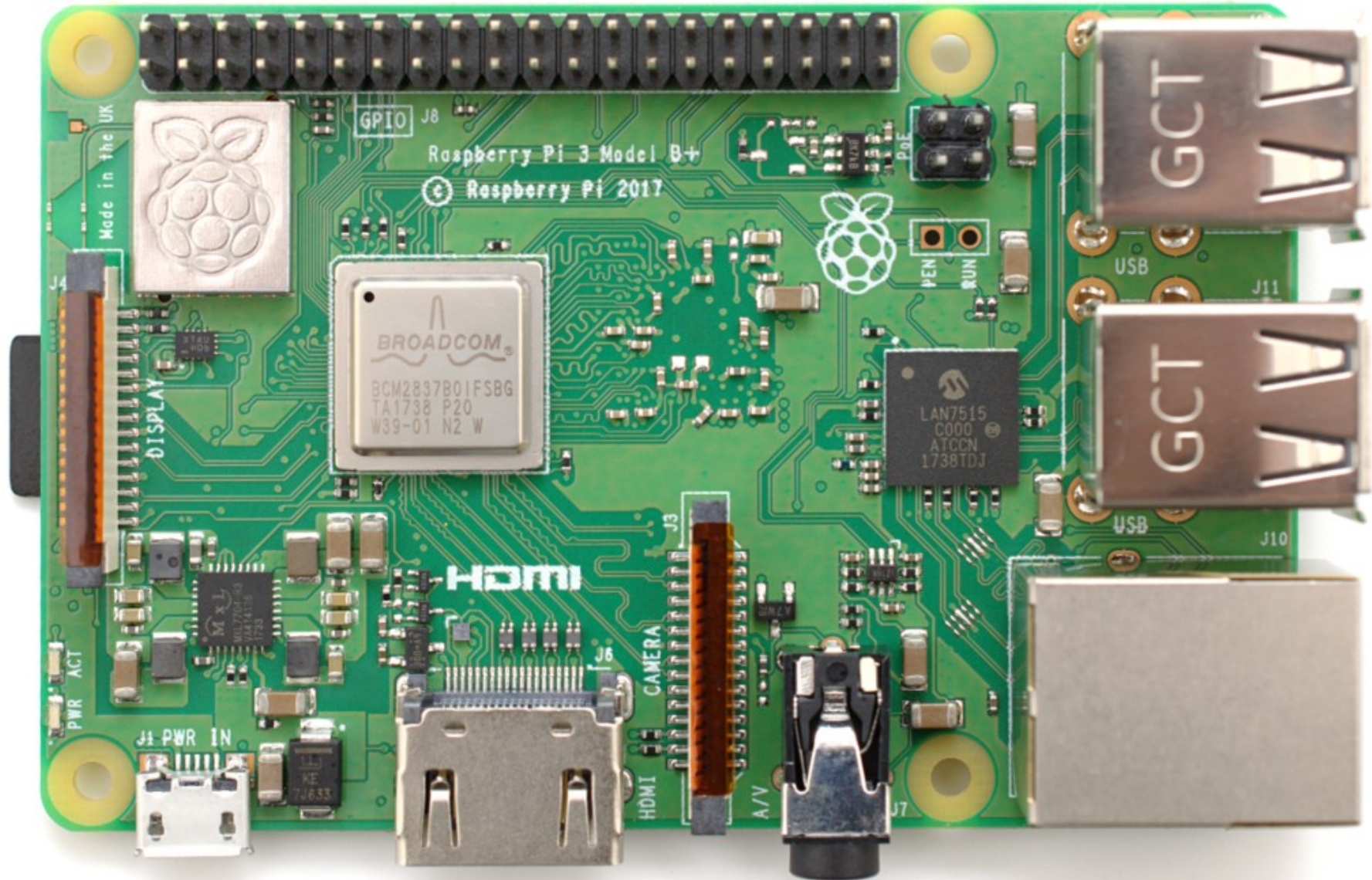
Contidos

- Pinout da Raspberry Pi
- Uso dos GPIO empregando Python en local
- Control dun LED con Python
- Lectura dun pulsador con Python (Pull Up)
- Control dun LED con pulsador + Python (+zumbador)
- ... até o infinito e máis alá! Outros dispositivos.
- Sensor de temperaturas dixital DS18B20

Acceso aos GPIO con Python





















- Raspberry Pi 3 dispón de 40 portos de E/S de propósito xeral (GPIO, General Purpose I/O).
- Están situados no lado oposto ao porto HDMI
- Existen dúas maneiras de numerar os pins GPIO:
 - BCM, a numeración do fabricante do chipset, e
 - Board, a definida pola fundación Raspberry Pi
- Para idendificalos considéranse dúas columnas de 20 pins

Acceso aos GPIO con Python



Acceso aos GPIO con Python

Raspberry Pi 3 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I ² C)		DC Power 5v	04
05	GPIO03 (SCL1 , I ² C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I ² C ID EEPROM)		(I ² C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

- Na nomenclatura Board
 - columna esquerda: pins impares
 - columna dereita: pins pares

Pinout online:

<https://pinout.xyz/pinout/ground#>

Acceso aos GPIO con Python

- Raspberry non incorpora por defecto a comunicación cos GPIO
- É preciso cargar o driver ou extensión para ter acceso
- Existen varias librerías de Python que nos permiten acceder ao pinout
 - Máis coñecida: RPi.GPIO
 - Máis usable: gpiozero
- Python permítenos acceder os pins de E/S (GPIO) sen limitacións

Acceso aos GPIO con Python

- En Python debemos escoller como accedemos os GPIO:
 - coa nomenclatura Board ou
 - coa nomenclatura BCM
- Imos facer o noso 'Ola mundo' maker encendendo un LED
- Igual que con Scratch, facemos o circuíto na placa de proba, comprobamos e funciona, e
- Elaboramos o script en Python
- Empregaremos nomenclatura BOARD e os pins 11 e 09 (ground)

GPIO en Python - Control dun LED

```
1  #!/usr/bin/python3.5
2
3  import RPi.GPIO as gpio  #Importamos as librerías
4  import time
5
6  gpio.setmode(gpio.BOARD)  #Informamos que usamos nomes BOARD
7
8  led = 11  #Pin 09 para o LED
9  descanso = 2  #Tempo de espera
10
11  gpio.setup(led, gpio.OUT)  #Pin para saída ao LED
12
13
14  while True:
15      gpio.output(led, True)  #Encendemos LED
16      time.sleep(descanso)  #Agardamos 2 seg
17      gpio.output(led, False)  #Apagamos LED
18      time.sleep(descanso)  #Agardamos 2 seg
19
20
21
```

python -m py_compile "led.py" (no directorio: /home/pi/Documents/Python Projects)
A compilación rematou correctamente.

líña: 18 / 21 col: 28 sel: 0 INS TAB modo: LF codificación: UTF-8 tipo de ficheiro: Python ámbito: descoñecido

GPIO en Python - Control dun LED

```
Ola.mundo.py x led.py x
1  #!/usr/bin/python3.5
2
3  import RPi.GPIO as gpio      #Importamos as librerias
4  import time
5
6  gpio.setmode(gpio.BOARD)     #Informamos que usamos nomes BOARD
7
8  led = 11                     #Pin 09 para o LED
9  descanso = 2                 #Tempo de espera
10
11  gpio.setup(led, gpio.OUT)    #Pin para saída ao LED
12
13
14  while True:
15      gpio.output(led, True)   #Encendemos LED
16      time.sleep(descanso)     #Agardamos 2 seg
17      gpio.output(led, False) #Apagamos LED
18      time.sleep(descanso)     #Agardamos 2 seg
19
20
21
```

GPIO en Python - Ler estado pulsador

- Imos agora ler o estado dun pulsador conectado ao pin 10 (BOARD).
- Imos empregar unha resistencia de pull-up para evitar estados de alta impedancia.
- Imprimiremos por pantalla unha mensaxe
- Para que non se nos encha a pantalla de mensaxes, porémoslle unha espera de 0.5 seg

GPIO en Python - Ler estado pulsador

```
Ola.mundo.py x led.py x pulsador.py x
1  #!/usr/bin/python3.5
2
3  import RPi.GPIO as gpio      #Importamos as librerias
4  import time
5
6  gpio.setmode(gpio.BOARD)     #Informamos que usamos nomes BOARD
7
8  pulsador = 10                #Pin 09 para o pulsador
9  estado = gpio.HIGH           #Pull-up
10 descanso = .5                #Tempo de espera
11
12 gpio.setup(pulsador, gpio.IN) #Pin para saida ao pulsador
13
14
15 while True:
16     estado = gpio.input(pulsador)
17     if(estado):
18         print('Boton pulsado')
19         time.sleep(descanso)
20     else:
21         print('Boton NON pulsado')
22         time.sleep(descanso)
23
```

GPIO en Python - Ler estado pulsador

The screenshot shows a remote viewer of a Raspberry Pi desktop. The window title is "pi's X desktop (raspberrypi:1) (1) - Remote Viewer". The desktop environment is Geany, with a terminal window open showing the execution of a Python script named "pulsador.py". The script is designed to read the state of a button connected to a GPIO pin (pin 09) on the Raspberry Pi. It uses the RPi.GPIO library and the time module. The script sets the mode to BOARD, configures pin 09 as an input with a pull-up resistor, and enters a loop that continuously checks the state of the pin. If the pin is HIGH, it prints "Boton pulsado" and sleeps for 0.5 seconds. If the pin is LOW, it prints "Boton NON pulsado" and sleeps for 0.5 seconds.

```
1  #!/usr/bin/python3.5
2
3  import RPi.GPIO as gpio    #Importamos as librerias
4  import time
5
6  gpio.setmode(gpio.BOARD)  #Informamos que usamos nomes BOARD
7
8  pulsador = 10              #Pin 09 para o pulsador
9  estado = gpio.HIGH        #Pull-up
10 descanso = .5              #Tempo de espera
11
12 gpio.setup(pulsador, gpio.IN) #Pin para saída ao pulsador
13
14
15 while True:
16     estado = gpio.input(pulsador)
17     if(estado):
18         print('Boton pulsado')
19         time.sleep(descanso)
20     else:
21         print('Boton NON pulsado')
22         time.sleep(descanso)
23
```

The terminal output shows the command: `python -m py_compile "pulsador.py"` (no directorio: /home/pi/Documents/Python Projects) and the message: "A compilación rematou correctamente."

At the bottom of the window, the status bar shows: "líña: 10 / 23 col: 13 sel: 0 INS TAB modo: LF codificación: UTF-8 tipo de ficheiro: Python ámbito: descoñecido".

GPIO en Python - RPi.GPIO vs gpiozero

- Imos facer un script python que encenda un LED cada vez que detecta unha pulsación, agarde 0.5 seg aceso e valide novamente o estado do pulsador
- Isto é a combinación dos dous scripts anteriores
- A novidade agora é que faremos o script con dúas librerías diferentes:
 - RPi.GPIO, a empregada até agora
 - gpiozero, máis usable (emprega nomenclatura BCM)

GPIO en Python - Ler estado pulsador

```
Ola.mundo.py x led.py x pulsador.py x pulsador.led.py x
1  #!/usr/bin/python3.5
2
3  import RPi.GPIO as gpio      #Importamos as librerias
4  import time
5
6  gpio.setmode(gpio.BOARD)     #Informamos que usamos nomes BOARD
7
8  led = 11                     #Pin 09 para o LED
9  descanso = .5                #Tempo de espera
10 pulsador = 10                #Pin 09 para o pulsador
11 estado = gpio.HIGH           #Pull-up
12
13 gpio.setup(led, gpio.OUT)     #Pin para saida ao LED
14 gpio.setup(pulsador, gpio.IN) #Pin para saida ao pulsador
15
16 while True:
17     estado = gpio.input(pulsador)
18     if(estado):
19         gpio.output(led, False) #Apagamos LED
20         print('Boton NON pulsado')
21         time.sleep(descanso)
22     else:
23         gpio.output(led, True)  #Encendemos LED
24         print('Boton pulsado')
25         time.sleep(descanso)
26
27
```

- Con RPi.GPIO

GPIO en Python - Ler estado pulsador

Ola.mundo.py ✕ led.py ✕ pulsador.py ✕ pulsador.led.py ✕ pulsador.led.gpioze

```
1  #!/usr/bin/python3
2
3  from gpiozero import LED, Button  #Importamos obxectos led e pulsador
4  import time
5
6  led = LED(17)                      #Pin 11 para o LED e gpio17
7  descanso = .5                      #Tempo de espera
8  pulsador = Button(15)              #Pin 10 para o pulsador e gpio15
9                                     #Coidado que e pull-up!!
10
11  while True:
12      if(not pulsador.is_pressed):
13          led.off()                  #Apagamos LED
14          print('Boton NON pulsado')
15          time.sleep(descanso)
16      else:
17          led.on()                   #Encendemos LED
18          print('Boton pulsado')
19          time.sleep(descanso)
20
21
```

- Con gpiozero

GPIO en Python

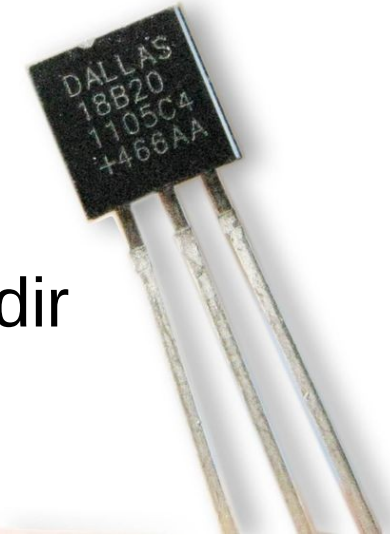
- Co que levamos visto podemos realizar o mesmo que xa vimos con Scratch
- Podemos empregar zumbadores, LDRs, motores, etc
- Sen embargo RPi.GPIO permítenos empregar ademais algúns pins como PWM (pulse-width modulation), para usar p.ex. con servos.

GPIO en Python - Sensor DS18B20

- O sensor DS18B20 é un sensor dixital de temperatura que non precisa calibración.
- Emprega o bus 1-wire, de forma que podemos poñer varios en paralelo para medir diferentes temperaturas cun único pin.
- Para usalo dende Python necesitaremos:
 - activar 1-wire con raspi-config
 - cargar as librerías para python
 - desenvolver o script Python

GPIO en Python - Sensor DS18B20

- O sensor DS18B20 é un sensor dixital de temperatura que non precisa calibración.
- Emprega o bus 1-wire, de forma que podemos poñer varios en paralelo para medir diferentes temperaturas cun único pin.
- Para usalo dende Python necesitaremos:
 - activar 1-wire con raspi-config
 - cargar as librerías para python
 - desenvolver o script Python

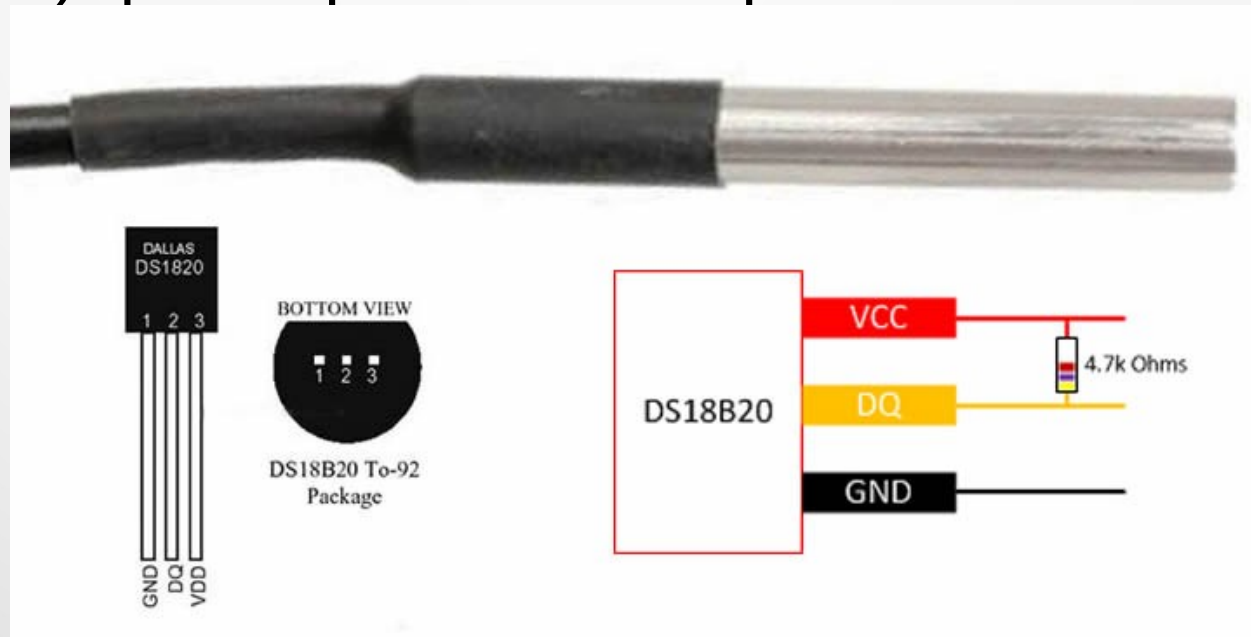


GPIO en Python - Sensor DS18B20

- Sobre o sensor:

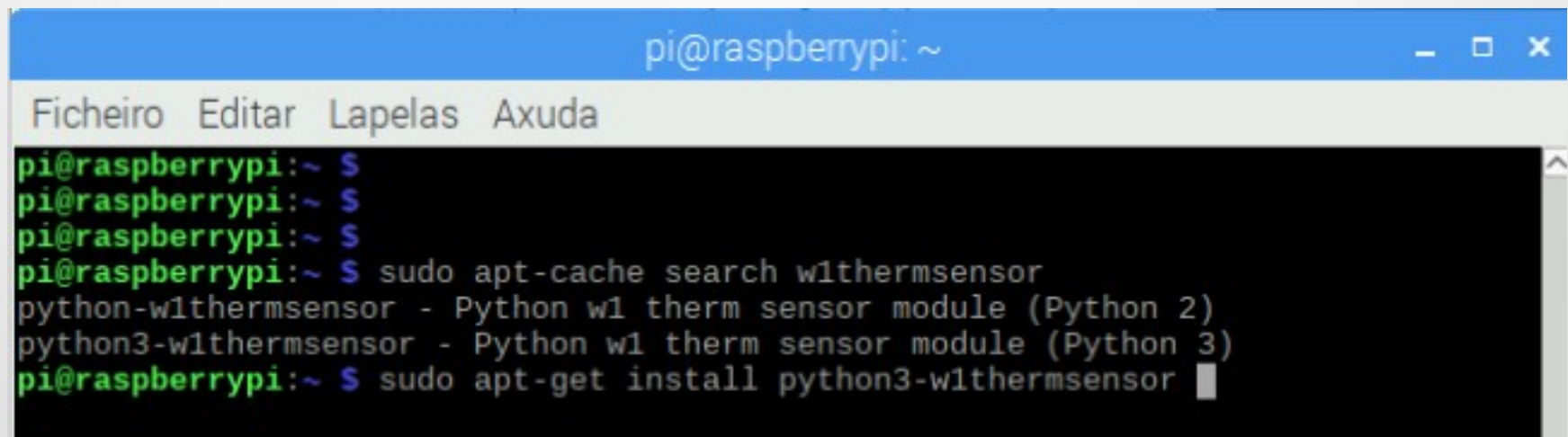
https://www.itead.cc/wiki/DS18B20_1_-_Wire_Digital_Thermometer_Module

- Precisa ademais dunha resistencia de Pull-up (~4.7 k Ω) que se pode simular por software



GPIO en Python - Sensor DS18B20

- Para activar o bus 1-wire:
 - executamos nun terminal: `sudo raspi-config`
 - escollemos '5. Interfacing options'
 - no seguinte menú 'P7.1-wire' seleccionamos 'yes'
- Para cargar a librería python que nos permite acceder facilmente ao sensor:

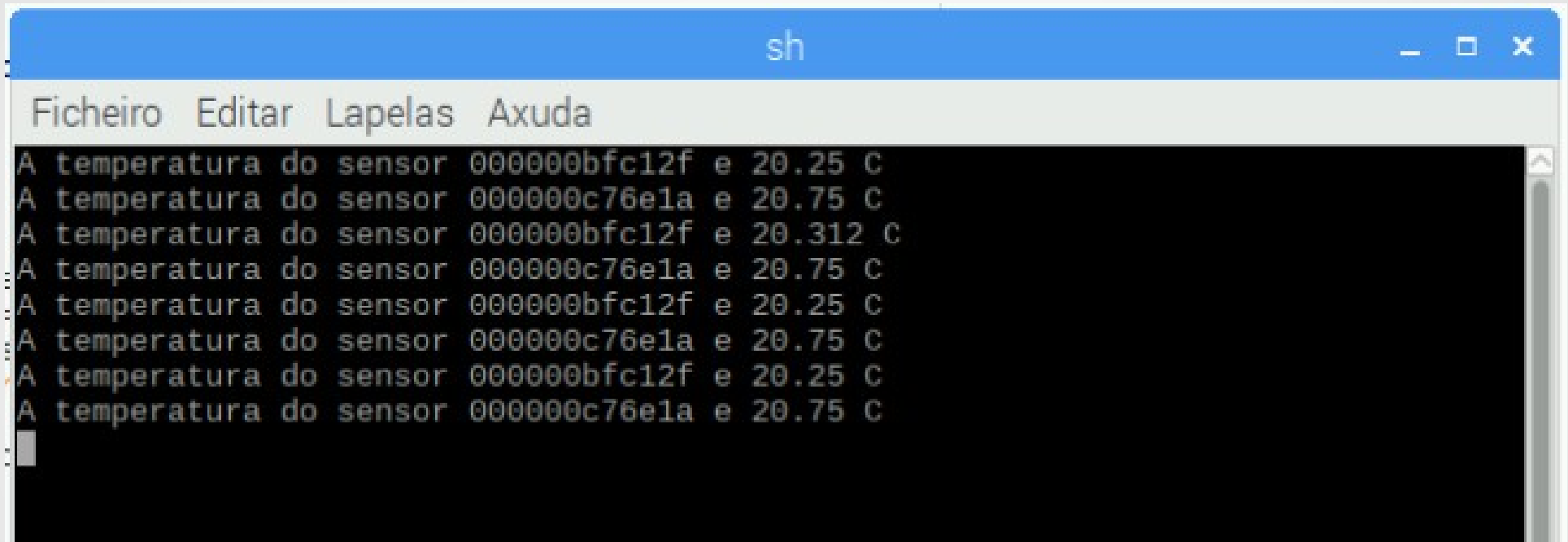


```
pi@raspberrypi: ~  
Ficheiro  Editar  Lapelas  Axuda  
pi@raspberrypi:~$  
pi@raspberrypi:~$  
pi@raspberrypi:~$  
pi@raspberrypi:~$ sudo apt-cache search w1thermsensor  
python-w1thermsensor - Python w1 therm sensor module (Python 2)  
python3-w1thermsensor - Python w1 therm sensor module (Python 3)  
pi@raspberrypi:~$ sudo apt-get install python3-w1thermsensor
```

GPIO en Python - Sensor DS18B20

```
Ola.mundo.py x led.py x sensor.temperatura.py x
1  #!/usr/bin/python3
2
3  from w1thermsensor import W1ThermSensor    #Importamos as librerias
4  import time
5
6  descanso = .5
7
8  while(True):
9      for sensor in W1ThermSensor.get_available_sensors():
10         identificador = sensor.id
11         temperatura = sensor.get_temperature()
12         print('A temperatura do sensor %s e %s C' % \
13               |(identificador, temperatura))
14         time.sleep(descanso)
15
16
17
```

GPIO en Python - Sensor DS18b20



```
sh
Ficheiro  Editar  Lapelas  Axuda
A temperatura do sensor 000000bfc12f e 20.25 C
A temperatura do sensor 000000c76e1a e 20.75 C
A temperatura do sensor 000000bfc12f e 20.312 C
A temperatura do sensor 000000c76e1a e 20.75 C
A temperatura do sensor 000000bfc12f e 20.25 C
A temperatura do sensor 000000c76e1a e 20.75 C
A temperatura do sensor 000000bfc12f e 20.25 C
A temperatura do sensor 000000c76e1a e 20.75 C
```

- Esta é a saída con dous sensores DS18b20 conectados aos pins: 07 (1wire), 01 (3.3 v) e 06 (ground)