

Textul și imaginile din acest document sunt licențiate

Attribution-NonCommercial-NoDerivs

CC BY-NC-ND



Codul sursă din acest document este licențiat

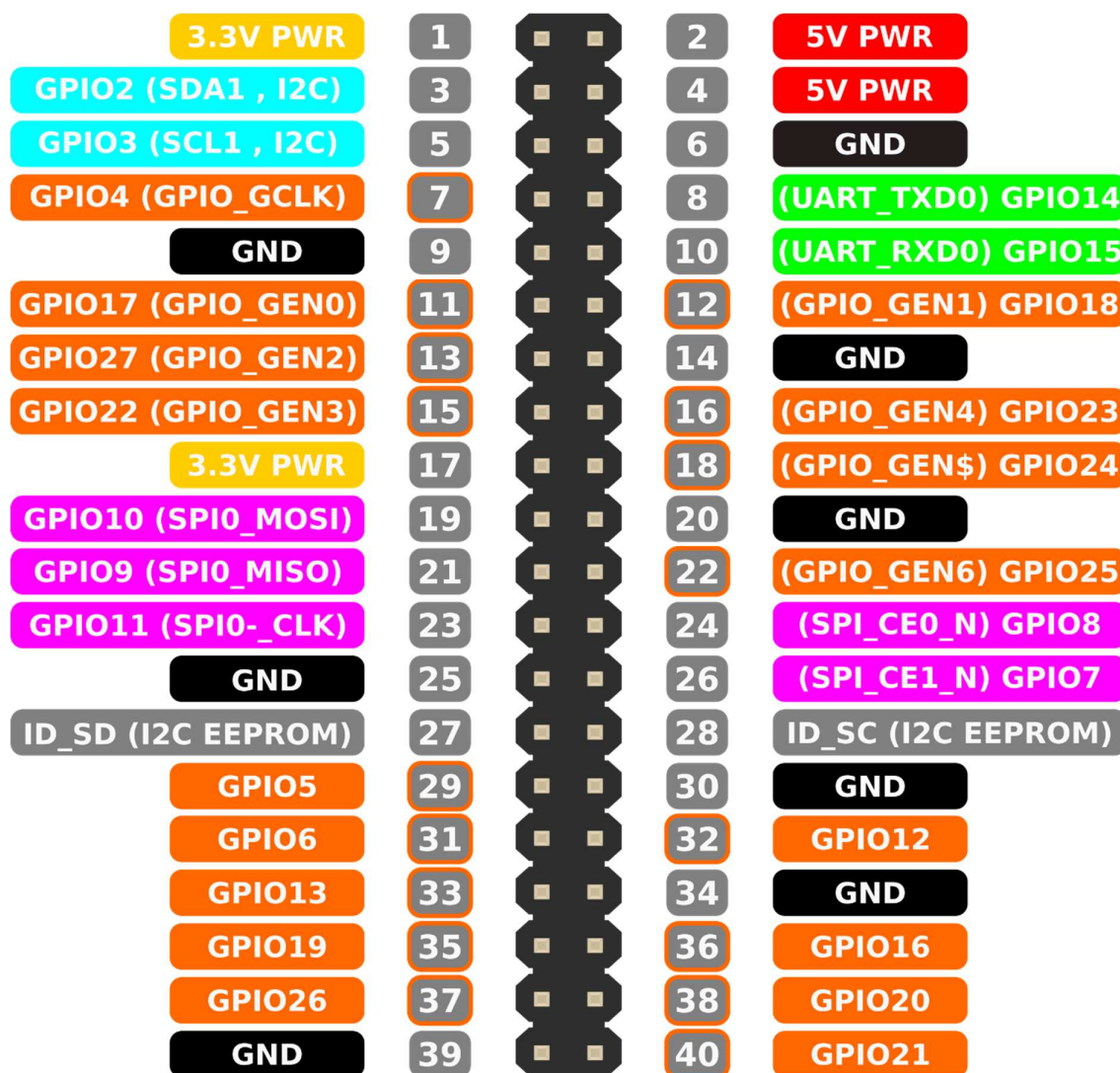
Public-Domain

Ești liber să distribui acest document prin orice mijloace consideri (email, publicare pe website / blog, tipărire, sau orice alt mijloc), atât timp cât nu aduci nici un fel de modificări acestuia. Codul sursă din acest document poate fi utilizat în orice fel de scop, de natură comercială sau nu, fără nici un fel de limitări dar autorii nu își asumă nici o răspundere pentru pagubele pricinuite de implementările realizate de utilizatori. Schemele și codul sursă au un rol educativ și nu sunt gândite pentru a fi utilizate în mediu de producție (industrial, casnic sau comercial).

Utilizarea pinilor GPIO la Raspberry Pi 3

Configurația pinilor GPIO la placa Raspberry Pi 3

Pe lângă resursele de calcul specifice unui sistem de calcul de uz general (microprocesor, memorie, interfața Ethernet și WiFi, porturi USB) placa Raspberry Pi 3 dispune și de un conector de 40 de pini ce expune o serie de pini digitali de intrare / ieșire (General-Purpose Input / Output). Diagrama conectorului și semnificația pinilor este prezentată în imaginea următoare.

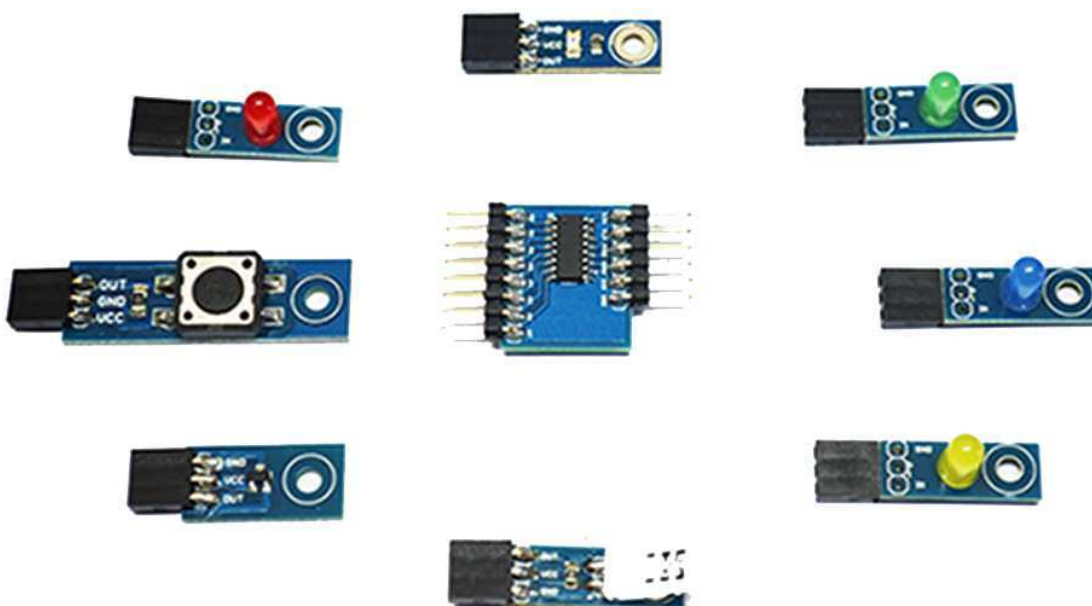


Din cei 40 de pini avem 2 pini de Power (alimentare) 3.3V – pinii 1 și 17, 2 pini de Power 5V – pinii 2 și 4, 8 pini de GND (masă) – pinii 6, 9, 14, 20, 25, 30, 34 și 39, 2 pini rezervați pentru identificarea plăcilor de extensie de tipul Pi HATS – pinii 27 și 28 și 24 de pini GPIO, unii dintre ei cu funcționalitate dublă, ca de exemplu:

- Pinii 3 și 5 (GPIO2 – SDA și GPIO3 – SCL) sunt și magistrală I2C;
- Pinii 8 și 10 (GPIO14 – UART_TX și GPIO15 – UART_RX) sunt și linii de comunicație serială UART;
- Pinii 19, 21, 23, 24 și 26 (GPIO10 – SPI0_MOSI, GPIO9 – SPI0_MISO, GPIO11 – SPI0_CLK, GPIO8 – SPI_CE0 și GPIO7 – SPI_CE1) – pini magistrală 0 SPI cu două semnale de chip select;

Toți pinii GPIO la Raspberry Pi suportă **maxim 3.3V** – conectarea de periferice cu un nivel logic "1" ce implică un nivel de tensiune mai mare de 3.3V va conduce la distrugerea definitivă a blocului GPIO al plăcii.

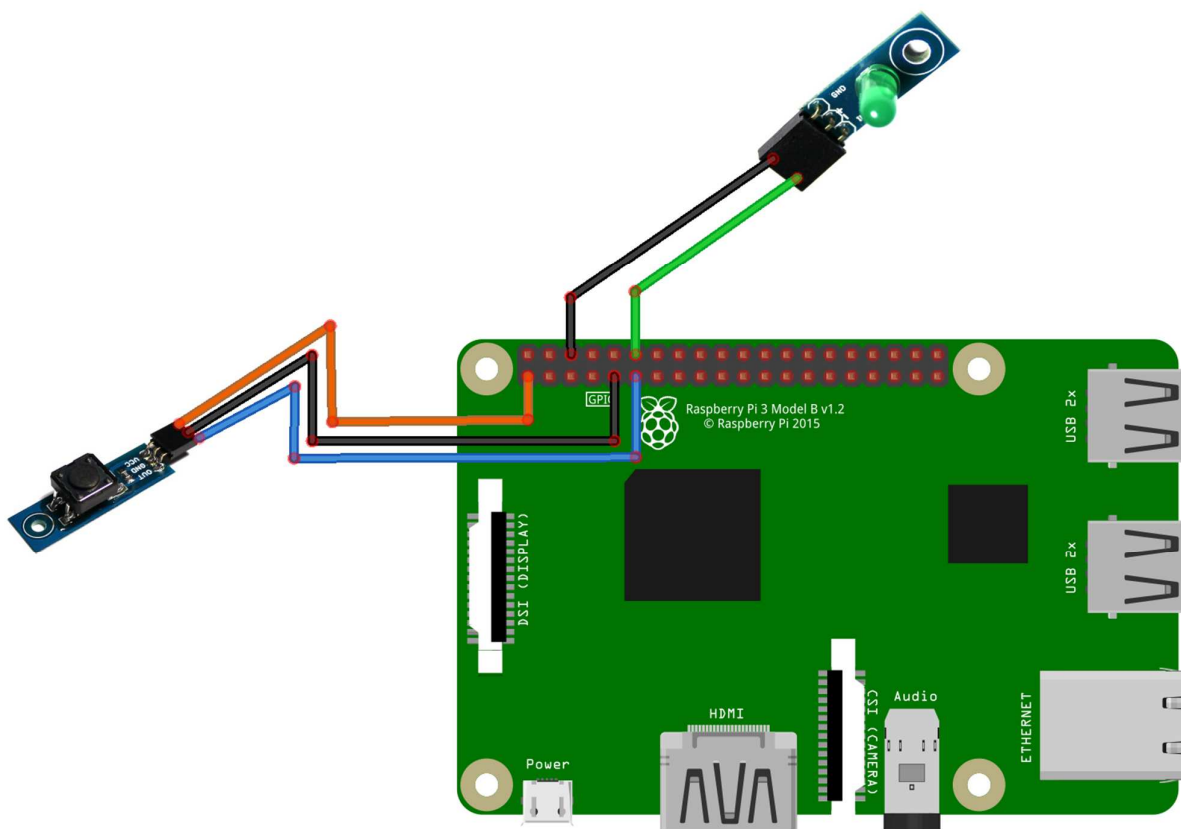
Pentru a ilustra utilizarea pinilor GPIO la placa Raspberry Pi 3 vom utiliza un kit de senzori pentru această placă ([1](#)) ce conține atât elemente digitale (brick-uri led și un brick buton) cât și analogice (senzori de temperatură, umiditate, lumină și un shield de achiziție analogică).



Utilizarea pinilor GPIO ca pini digitali

Utilizarea pinilor GPIO ca pini digitali (pini de intrare sau de ieșire cu valori logice "1" sau "0" – tensiune 3.3V sau 0V) se poate face din mai multe limbaje de programare. Vom exemplifica utilizarea pinilor GPIO utilizând două din cele mai utilizate limbaje de programare și anume: C ([2](#)) și Python ([3](#)).

Vom utiliza o schemă electrică compusă din placa Raspberry Pi 3, un brick led (element de ieșire) și un brick buton (element de intrare):



Brick-ul buton se va conecta la 3.3V (VCC), GND (GND) și pinul 11 (GPIO17). Brick-ul LED se va conecta la GND (GND) și la pinul 12 (GPIO18). Vom scrie, în ambele limbaje de programare, un program care va aprinde LED-ul când se va apăsa pe buton (se va activa o ieșire la activarea unei intrări).

Se va crea un fișier `exemplu1.py` (utilizând orice editor de text în linie de comandă,

nano de exemplu, sau în interfața grafică) care va conține următorul cod în limbajul Python:

```
import RPi.GPIO as GPIO
ledPin = 18
butPin = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin, GPIO.OUT)
GPIO.setup(butPin, GPIO.IN)
GPIO.output(ledPin, GPIO.LOW)
print("Press CTRL+C to exit")
try:
    while 1:
        if GPIO.input(butPin):
            GPIO.output(ledPin, GPIO.HIGH)
        else:
            GPIO.output(ledPin, GPIO.LOW)
except KeyboardInterrupt:
    GPIO.cleanup()
```

După crearea fișierului sursă programul poate fi executat imediat (limbajul Python nu necesită compilare) cu ajutorul comenzii (în *Terminal*):

```
python exemplul.py
```

Atâta timp cât programul rulează funcționalitatea descrisă anterior (LED-ul se va aprinde la apăsarea butonului) va merge. Pentru oprirea programului se va apăsa combinația de taste *CTRL+C*.

Elementele importante din program sunt importarea bibliotecii Python – *RPi.GPIO* – responsabilă cu funcțiile specifice accesului la pinii GPIO, funcțiile *GPIO.setmode* și *GPIO.setup* ce permit configurarea pinilor ca intrare sau ca ieșire și funcțiile *GPIO.output* și *GPIO.input* ce permit setarea și citirea pinilor de ieșire / intrare.

Codul echivalent în limbajul C este:

```
#include <stdio.h>
#include <wiringPi.h>

const int ledPin = 18;
const int butPin = 17;

int main(void)
{
    wiringPiSetupGpio();
    pinMode(ledPin, OUTPUT);
    pinMode(butPin, INPUT);
    printf("Press CTRL+C to quit.\n");
    while(1)
    {
        if (digitalRead(butPin))
        {
            digitalWrite(ledPin, HIGH);
        }
        else
        {
            digitalWrite(ledPin, LOW);
        }
    }
    return 0;
}
```

Programul trebuie compilat înainte de execuție cu ajutorul comenzii:

```
gcc -o exemplu1 exemplu1.c -l wiringPi
```

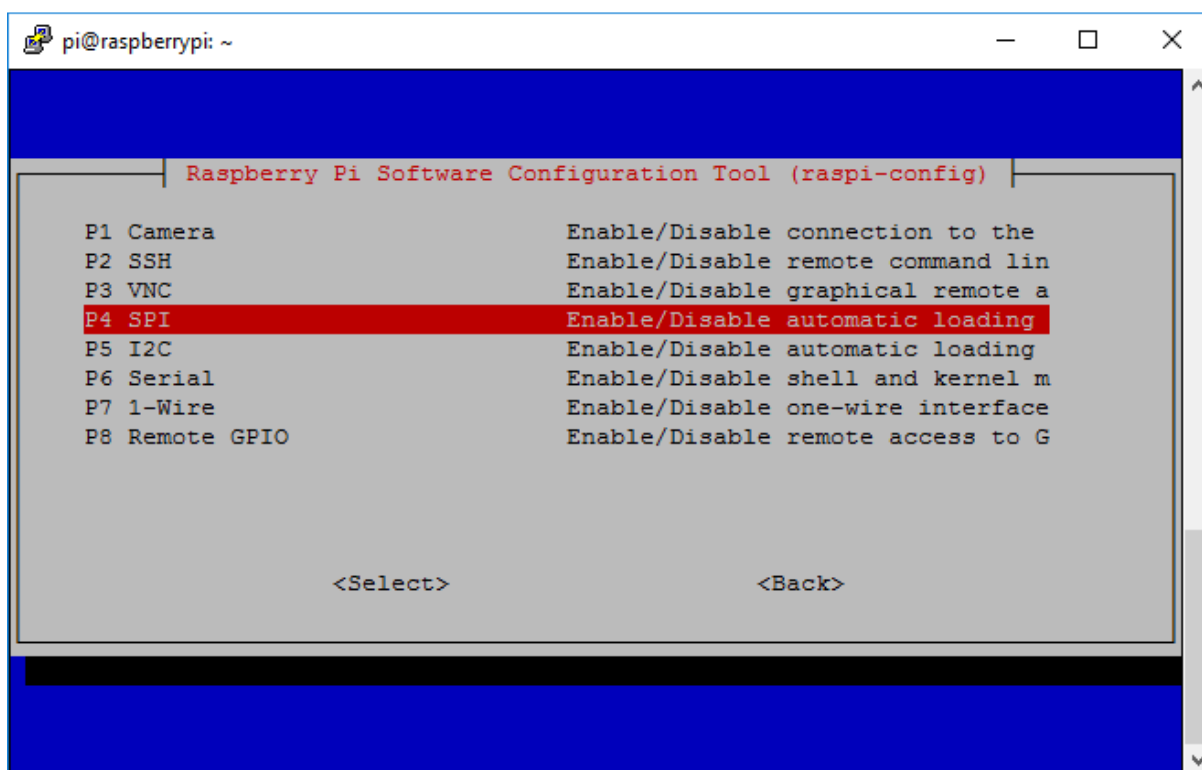
După compilare poate fi executat cu ajutorul comenzii:

```
./exemplul
```

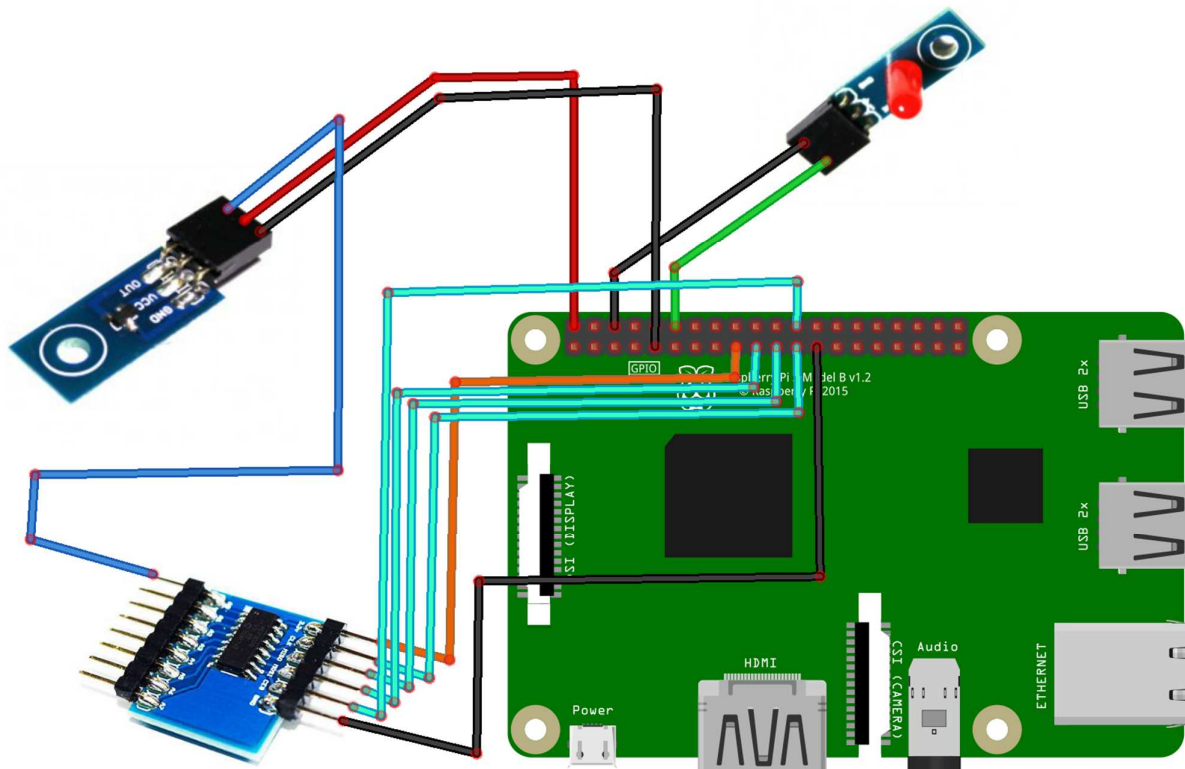
Se poate observa ușor echivalența elementelor principale din programul scris în limbajul C și cel scris în Python. Biblioteca utilizată se numește *wiringPi* (4) iar funcțiile utilizate sunt *pinMode*, *digitalRead* și *digitalWrite* (funcții similare se regăsesc și în mediul Arduino IDE).

Achiziția analogică și comunicația SPI

Placa Raspberry Pi nu dispune de un convertor analog-numeric, nu poate citi valori intermediare în intervalul 0 – 3.3V ci doar valori digitale de "0" logic și "1" logic. Din acest motiv dacă dorim să utilizăm senzori analogici trebuie să utilizăm un convertor extern. Shield-ul din kit-ul de senzori conține un astfel de convertor, mai exact un circuit MCP3008 (5) cu opt canale de intrare (se pot conecta până la opt senzori analogici). Acest circuit va comunica cu placa Raspberry Pi prin intermediul interfeței SPI, cu alte cuvinte achiziția analogică se va realiza prin intermediul interfeței SPI. Înainte de conectarea shield-ului (6) se va activa comunicația SPI cu ajutorul utilitarului *raspi-config*:



Pentru ilustrarea achiziției analogice vom utiliza următoarea schemă electrică (placă Raspberry Pi 3, un brick LED roșu, shield-ul de senzori (6) și un brick sensor analogic de temperatură (7)):



Conexiunile din schemă sunt următoarele:

- Brick-ul LED este conectat ca și în schema precedentă: GND – GND (pin 6), IN – GPIO18 (pin 12);
- Shield-ul de senzori se conectează la placa Raspberry Pi astfel: 3.3V – 3.3V (pin 17), CLK – SPI0_CLK (pin 23), MISO – SPI0_MISO (pin 21), MOSI – SPI0_MOSI (pin 19), CE – SPI0_CE0 (pin 24), GND – GND (pin 25);
- Brick-ul sensor de temperatură se conectează GND – GND (pin 9), VCC – 5V (pin 2) și OUT (ieșirea analogică) se conectează la canalul 0 al shield-ului de senzori.

Scopul schemei este să citim valoarea măsurată de senzorul analogic de temperatură (senzor LM50 (8)) prin intermediul shield-uri de senzori SPI, să afișăm valoarea măsurată (valoarea tensiunii achiziționate și valoarea temperaturii echivalente) și, în cazul în care temperatura este mai mare de 30 de grade Celsius, să aprindem LED-ul –

funcționalitate de alarmă termică vizuală. Codul utilizat este scris în limbajul Python:

```
import spidev
import time
import RPi.GPIO as GPIO
ledPin = 18
spi = spidev.SpiDev()
spi.open(0,0)
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin, GPIO.OUT)
GPIO.output(ledPin, GPIO.LOW)
def readadc(adcnum):
    if ((adcnum > 7) or (adcnum < 0)):
        return -1
    r = spi.xfer2([1, (8+adcnum)<<4,0])
    adcout = ((r[1]&3) << 8) + r[2]
    return adcout

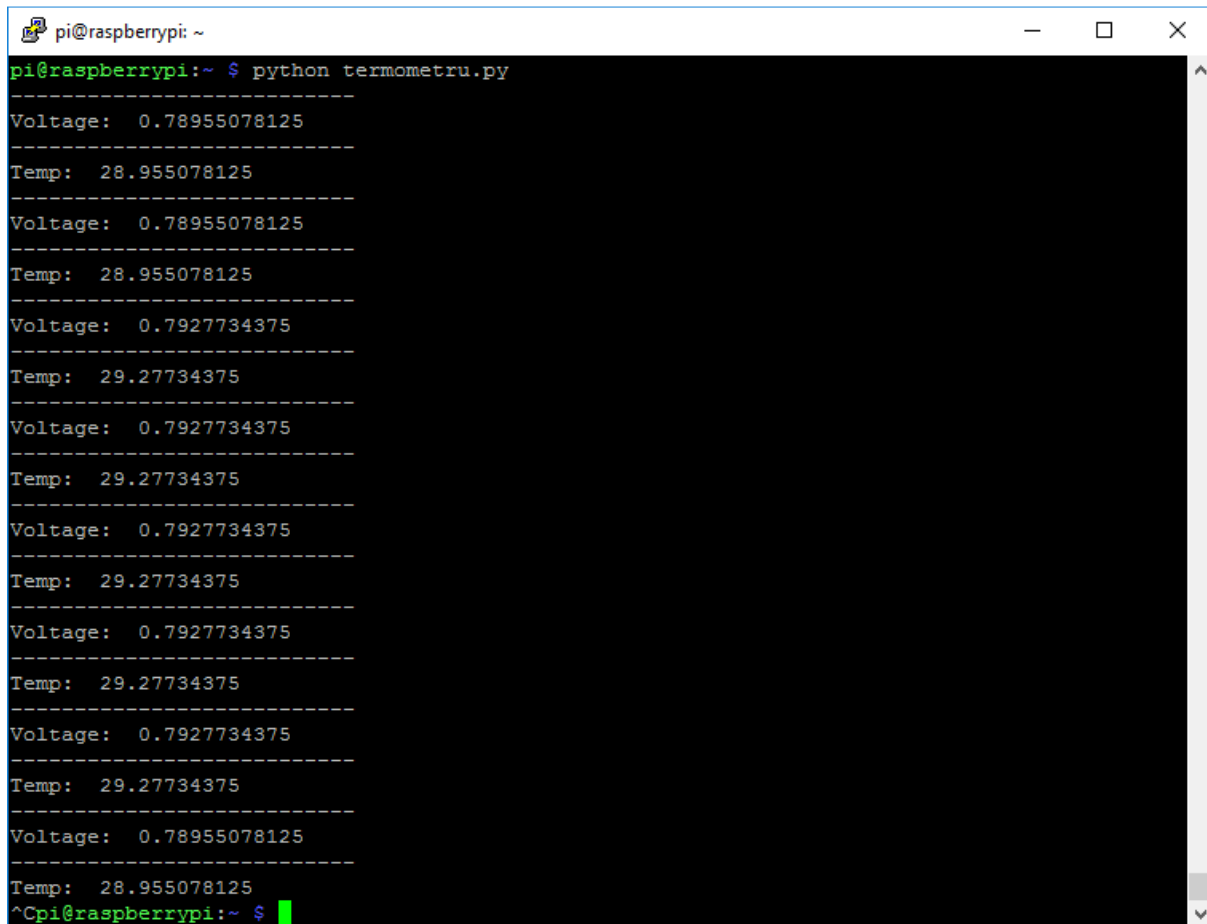
try:
    while True:
        value = readadc(0)
        voltage = value * 3.3
        voltage /= 1024.0
        tempCelsius = (voltage-0.5)*100
        if (tempCelsius>30):
            GPIO.output(ledPin, GPIO.HIGH)
        else:
            GPIO.output(ledPin, GPIO.LOW)
        print "-----"
        print "Voltage: ", voltage
        print "-----"
        print "Temp: ", tempCelsius
```

```
        time.sleep(1)
except KeyboardInterrupt:
    GPIO.cleanup()
```

Comanda LED-ului în program se face identic cu exemplul precedent. Pentru partea de comunicație SPI utilizăm biblioteca *spidev* (9) care, prin intermediul funcției *readadc*, ne permite citirea valorii achiziționate de la circuitul extern de conversie analog-numerice. Tensiunea măsurată de circuitul de conversie este convertită în temperatură (grade Celsius) conform documentației senzorului LM50 (8).

Ca și în cazul precedent programul se execută direct cu comanda (presupunem că am salvat programul în fișierul *termometru.py*):

```
python termometru.py
```



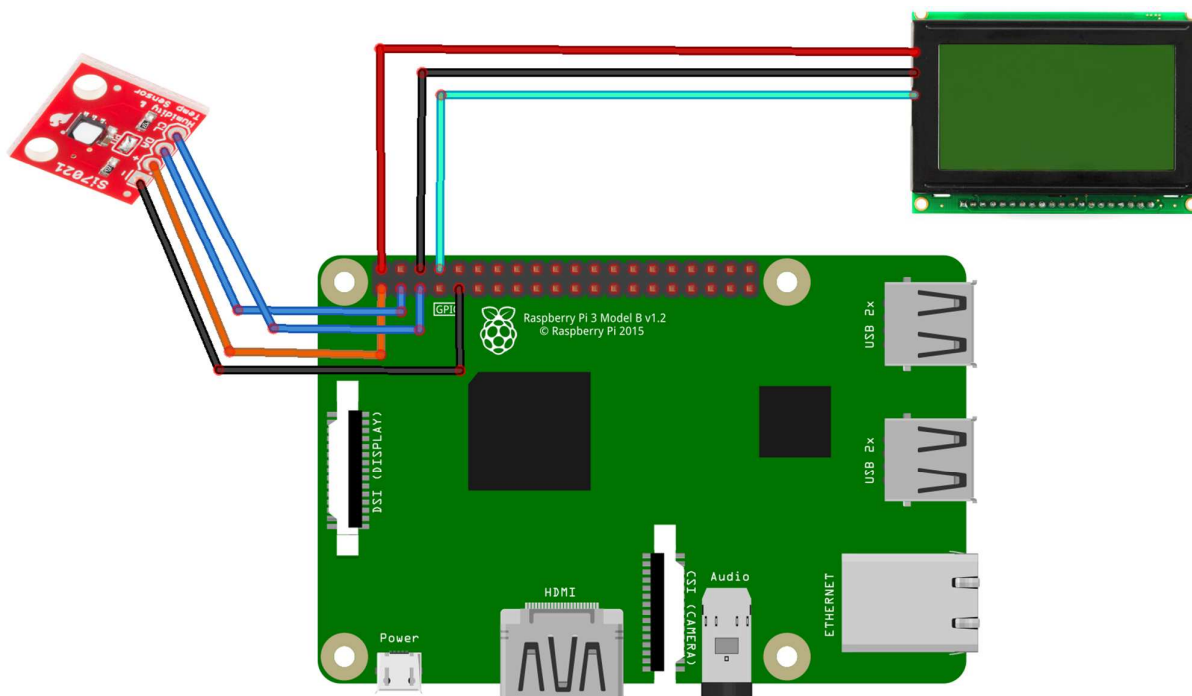
```
pi@raspberrypi: ~
pi@raspberrypi:~ $ python termometru.py
-----
Voltage:  0.78955078125
-----
Temp:    28.955078125
-----
Voltage:  0.78955078125
-----
Temp:    28.955078125
-----
Voltage:  0.7927734375
-----
Temp:    29.27734375
-----
Voltage:  0.7927734375
-----
Temp:    29.27734375
-----
Voltage:  0.7927734375
-----
Temp:    29.27734375
-----
Voltage:  0.7927734375
-----
Temp:    29.27734375
-----
Voltage:  0.78955078125
-----
Temp:    28.955078125
^Cpi@raspberrypi:~ $
```

Execuția programului se oprește cu combinația de taste *CTRL+C*.

Comunicația I2C și comunicația serială UART

Alte două instrumente puse la dispoziție de pinii plăcii Raspberry Pi 3 sunt pinii de magistrală I2C (10) și portul de comunicație serială UART (11). Ambele metode de comunicație pot permite conectarea la placa Raspberry Pi de diverse module electronice suplimentare: senzori digitali I2C, ecrane LCD / TFT I2C sau seriale, module GPS seriale etc. Vom exemplifica utilizarea celor două canale de comunicație utilizând un senzor digital de temperatură și umiditate I2C – Si7021 breakout (12) – și un ecran LCD grafic serial 128x64 (13). La fel ca și în cazul magistralei SPI, este necesară activarea comunicațiilor I2C și seriale utilizând utilitarul *raspi-config*.

Schema de interconectare între placa Raspberry Pi și cele două componente este următoarea:



Senzorul Si7021 se va conecta la cei doi pini I2C ai plăcii: DA senzor – pin SDA / GPIO2 placă, CL senzor – pin SCL / GPIO3 placă iar alimentarea la 3.3V: + senzor – pin 1 placă, - senzor – pin 9 placă.

Afișajul LCD serial se alimentează la 5V: Vin LCD – pin 2 placă, GND LCD – pin 6 placă iar comunicația se va realiza unidirecțional din motive de niveluri logice de funcționare (LCD-ul funcționează la 5V iar placa la 3.3V) și din cauză că nu este nevoie ca LCD-ul să trimită date către placa ci doar placa să trimită date către LCD: pinul RX al LCD-ului se va conecta la pinul 8 (UART_TX) al plăcii.

Programul necesar funcționării sistemului este scris tot în limbajul de programare Python și utilizează bibliotecile [smbus \(14\)](#) (necesară comunicației I2C) și [serial \(15\)](#) (necesară comunicației seriale).

```
import smbus
import time
import serial
```

Următoarea secțiune deschide și configurează portul serial și inițializează comunicația I2C:

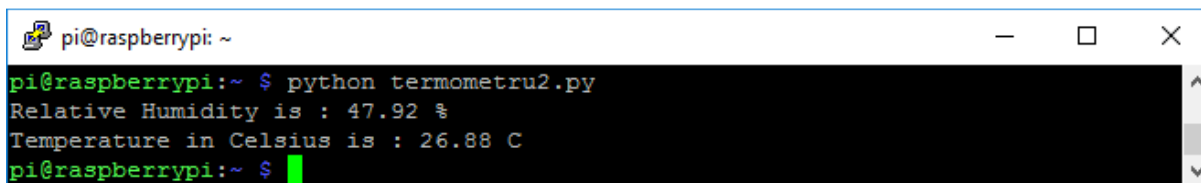
```
port = serial.Serial('/dev/ttyS0', 115200, timeout=3.0)
bus = smbus.SMBus(1)
```

Comunicația cu senzorul Si7021 se realizează cu ajutorul comenzilor directe de citire a registrelor interne a acestuia. Pentru mai multe detalii puteți consulta și materialul *Build a Hygrometer at Home Using Raspberry Pi and SI7021* ([16](#)).

```
bus.write_byte(0x40, 0xF5)
time.sleep(0.3)
data0 = bus.read_byte(0x40)
data1 = bus.read_byte(0x40)
humidity = ((data0 * 256 + data1) * 125 / 65536.0) - 6
time.sleep(0.3)
bus.write_byte(0x40, 0xF3)
time.sleep(0.3)
data0 = bus.read_byte(0x40)
data1 = bus.read_byte(0x40)
cTemp = ((data0 * 256 + data1) * 175.72 / 65536.0) - 46.85
print "Relative Humidity is : %.2f %" %humidity
print "Temperature in Celsius is : %.2f C" %cTemp
```

Programul va afișa și în consola de comenzi temperatura și umiditatea citită de la

senzor:

A terminal window titled 'pi@raspberrypi: ~' with standard window controls. It shows the command 'python termometru2.py' being executed. The output is: 'Relative Humidity is : 47.92 %' and 'Temperature in Celsius is : 26.88 C'. The prompt 'pi@raspberrypi:~ \$' is followed by a green cursor.

```
pi@raspberrypi:~ $ python termometru2.py
Relative Humidity is : 47.92 %
Temperature in Celsius is : 26.88 C
pi@raspberrypi:~ $
```

Ultima secțiune a programului va realiza afișarea pe LCD. Comenzile de inițializare, ștergere a ecranului și poziționare a cursorului sunt specifice protocolului serial al LCD-ului. Pentru funcționalități suplimentare, inclusiv parte de afișare grafică, consultați manualul oficial (17).

```
port.write(b'\x7C\x00')
port.write(b'\x7C\x18\x00')
port.write(b'\x7C\x19\x00')
time.sleep(1)
port.write('Temperatura: ')
port.write('%.2f' % cTemp)
port.write('oC\r\n')
time.sleep(1)
port.write('Umiditate: ')
port.write('%.2f' % humidity)
port.write('%')
port.close()
```



Programul se va executa ca și în cazurile precedente cu ajutorul comenzii (presupunând că programul a fost salvat sub denumirea *termometru2.py*):

```
python termometru2.py
```

Referințe on-line

(1) Kit Senzori Raspberry PI

<https://www.robofun.ro/raspberry-pi-si-componente/Kit-RASPBERRY-PI-B-pentru-incepatori>

(2) C (programming language)

[https://en.wikipedia.org/wiki/C_\(programming_language\)](https://en.wikipedia.org/wiki/C_(programming_language))

(3) Python (programming language)

[https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))

(4) WiringPi

<http://wiringpi.com/>

(5) MCP3008 - Mixed Signal - Successive Approximation Register A/D Converters

<http://www.microchip.com/wwwproducts/en/mcp3008>

(6) Sensor Shield V4 pentru Raspberry Pi

<https://www.robofun.ro/sensor-brick-pentru-raspberry-pi>

(7) Senzor Temperatura Brick

<https://www.robofun.ro/senzor-temperatura-brick>

(8) LM50 Single-Supply Centigrade Temperature Sensor

<http://www.ti.com/lit/ds/symlink/lm50.pdf>

(9) SpiDev Documentation

http://tightdev.net/SpiDev_Doc.pdf

(10) I2C - learn.sparkfun.com

<https://learn.sparkfun.com/tutorials/i2c>

(11) Serial Communication - learn.sparkfun.com

<https://learn.sparkfun.com/tutorials/serial-communication>

(12) Senzor Umiditate si Temperatura - Si7021

<https://www.robofun.ro/senzori/vreme/enzor-umiditate-si-temperatura-si7021>

(13) LCD Grafic 128x64 Serial

https://www.robofun.ro/lcd/lcd_grafic_serial_128x64

(14) Using the I2C Interface – Raspberry Pi Projects

<http://www.raspberrypi-projects.com/pi/programming-in-python/i2c-programming-in-python/using-the-i2c-interface-2>

(15) Serial port programming - eLinux.org

http://www.elinux.org/Serial_port_programming

(16) Build a Hygrometer at Home Using Raspberry Pi and SI7021: 6 Steps

<http://www.instructables.com/id/Build-a-Hygrometer-at-Home-Using-Raspberry-Pi-and-/>

(17) Graphic LCD Serial Backpack Datasheet

<https://www.sparkfun.com/datasheets/LCD/Monochrome/Corrected-SFE-0016-DataSheet-08884-SerialGraphicLCD-v2.pdf>