Raspberry PI si senzorul BMP085 Blue Edition

BMP085 Blue Edition este un senzor de inalta precizie pentru presiune atmosferica si temperatura. Senzorul poate masura de la 300 la 1100 hPa cu o precizie absoluta de pana la 0.03 hPa. Senzorul se alimenteaza cu o tensiune cuprinsa intre 1.8 si 3.6V si se conecteaza direct la portul I2C al placii Raspberry PI.

In acest tutorial vei programa placa Raspberry PI sa afiseze pe un shield LCD 16x2 temperatura si presiunea atmosferica.

Vei avea nevoie de urmatoarele componente:

- O placa Raspberry PI.
- Un shield LCD 16x2 pentru Raspberry PI.
- Un senzor BMP085 Blue Edition.
- Fire pentru conexiuni.
- O sursa de alimentare pentru Raspberry (5V).

Cum conectez shield-ul LCD si senzorul BMP085 ?

1. Conecteaza shield-ul la portul GPIO al placii Raspberry PI. Shield-ul poate fi infipt direct in portul GPIO sau il poti conecta prin intermediul panglicii de tip cobbler.

2. Asigura-te ca bareta mama 2x13 este lipita pe shield, altfel nu poti conecta senzorul de temperatura si presiune atmosferica. La achizitionarea shield-ului poti opta pentru shield cu bareta lipita.

3. Conecteaza senzorul BMP085 conform tabelului urmator:

Raspberry PI 3.3V	BMP085 pin VCC
Raspberry PI GND	BMP085 pin GND
Raspberry PI pin SDA	BMP085 pin SDA
Raspberry PI pin SCL	BMP085 pin SCL

4. Foloseste aceasta schema a portului GPIO ca referinta:

3.3V	1	2	5V
I2C0 SDA	3	4	DNC
I2C0 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
DNC	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 21	13	14	DNC
GPIO 22	15	16	GPIO 23
DNC	17	18	GPIO 24
SP10 MOSI	19	20	DNC
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
DNC	25	26	SP10 CE1 N

Configurarea portului I2C.

Inainte de a instala programul este necesara configurarea portului I2C al Raspberry PI.

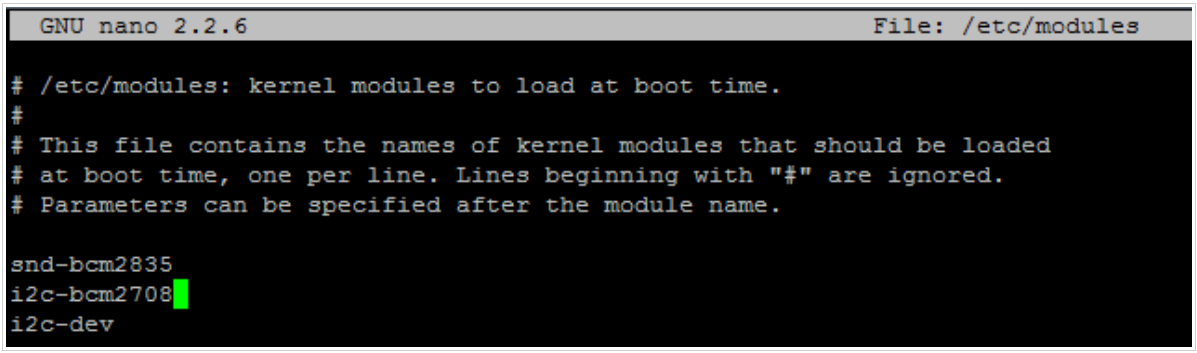
1. Deschide urmatorul fisier prin comanda:

```
sudo nano /etc/modules
```

2. Adauga in acest fisier urmatoarele linii:

```
i2c-bcm2708

i2c-dev
```



3. Salveaza fisierul cu CTRL X si Y si restarteaza Raspberry PI prin:

```
reboot
```

4. Dupa restartare executa urmatoarele 2 comenzi:

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

5. Deschide urmatorul fisier prin comanda:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

6. Comenteaza cele 2 linii ca in imagine:

```
blacklist spi-bcm2708
blacklist i2c-bcm2708
```

```
 GNU nano 2.2.6                                File: /etc/modprobe.d/raspi-blacklist.conf

#Comenteaza cele doua linii: blacklist spi-bcm2708 blacklist i2c-bcm2708
#Apasa CTRL# blacklist spi and i2c by default (many users don't need
#them)

#blacklist spi-bcm2708
#blacklist i2c-bcm2708
```

7. Asigura-te ca senzorul este detectat corect. Tasteaza urmatoarea comanda:

```
sudo i2cdetect -y 1
```

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
```

8. Senzorul este detectat corect la adresa 0x77.

Programul senzorului.

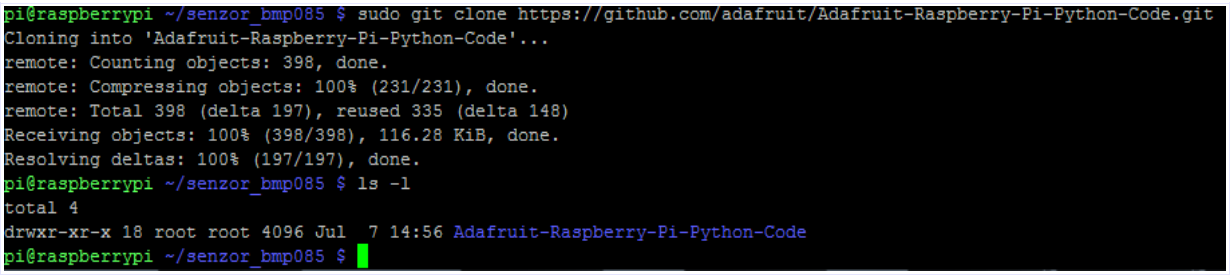
Exista o librarie special construita pentru acest senzor. Instalarea este structurata pe urmatorii pasi:

1. In primul pas vei creea un fisier. Acolo se vor regasi libraria si programul senzorului.

```
sudo mkdir senzor_bmp085
```

2. Executa urmatoarea comanda:

sudo git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git



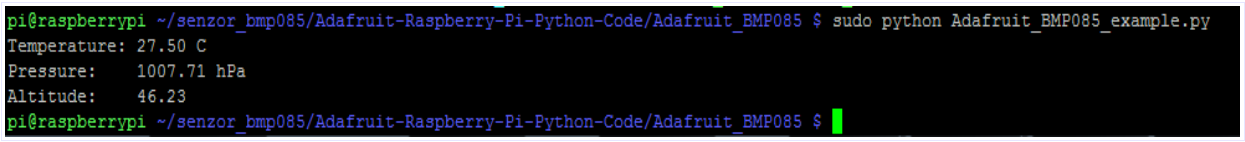
3. Schimba locatia:

cd Adafruit-Raspberry-Pi-Python-Code

cd Adafruit_BMP085

4. Executa urmatoarea comanda:

```
sudo python Adafruit_BMP085_example.py
```



5. Acest program citeste datele senzorului si le afiseaza in terminal. Urmeaza sa modifici acest program pentru ca datele sa fie afisate pe shield-ul LCD.

Cum afisez temperatura si presiunea pe un LCD?

In continuare este listat codul sursa modificat. Copiaza-l si executa-l cu python.

```
#!/usr/bin/python
from Adafruit_BMP085 import BMP085
import RPi.GPIO as GPIO
import time

LCD_RS = 22
LCD_E  = 18
LCD_D4 = 16
LCD_D5 = 11
LCD_D6 = 12
LCD_D7 = 15

LCD_WIDTH = 16
LCD_CHR = True
LCD_CMD = False

LCD_LINE_1 = 0x80
LCD_LINE_2 = 0xC0


E_PULSE = 0.00005
E_DELAY = 0.00005
```

```
#
================================================================
========
# Example Code
#
================================================================
========

# Initialise the BMP085 and use STANDARD mode (default value)
# bmp = BMP085(0x77, debug=True)
def main():
 lcd_init()
 lcd_byte(LCD_LINE_1, LCD_CMD)
 lcd_string("ROBOFUN.RO",2)
 lcd_byte(LCD_LINE_2, LCD_CMD)
 lcd_string("Raspberry PI",2)
 time.sleep(5)


 bmp = BMP085(0x77)

# To specify a different operating mode, uncomment one of the
following:
# bmp = BMP085(0x77, 0)  # ULTRALOWPOWER Mode
# bmp = BMP085(0x77, 1)  # STANDARD Mode
# bmp = BMP085(0x77, 2)  # HIRES Mode
# bmp = BMP085(0x77, 3)  # ULTRAHIRES Mode
 while 1:
  temp = bmp.readTemperature()
  pressure = bmp.readPressure()
  altitude = bmp.readAltitude()
```

```
  print "Temperatura: %.2f C" % temp
  print "Presiune:    %.2f hPa" % (pressure / 100.0)
  print "Altitudine:    %.2f" % altitude

  textLCDOne = "Temp:%s C" % temp
  textLCDTwo = "Pres:%s hPa" % (pressure / 100.0)

  lcd_byte(LCD_LINE_1, LCD_CMD)
  lcd_string(textLCDOne,1)
  lcd_byte(LCD_LINE_2, LCD_CMD)
   lcd_string(textLCDTwo,1)
  time.sleep(1)

def lcd_init():
  GPIO.setwarnings(False)
  GPIO.setmode(GPIO.BOARD)
  GPIO.setup(LCD_E, GPIO.OUT)
  GPIO.setup(LCD_RS, GPIO.OUT)
  GPIO.setup(LCD_D4, GPIO.OUT)
  GPIO.setup(LCD_D5, GPIO.OUT)
  GPIO.setup(LCD_D6, GPIO.OUT)
  GPIO.setup(LCD_D7, GPIO.OUT)


  lcd_byte(0x33,LCD_CMD)
  lcd_byte(0x32,LCD_CMD)
  lcd_byte(0x28,LCD_CMD)
  lcd_byte(0x0C,LCD_CMD)
  lcd_byte(0x06,LCD_CMD)
  lcd_byte(0x01,LCD_CMD)


def lcd_string(message,style):
  # style=1 Left justified
  # style=2 Centred
  # style=3 Right justified


  if style==1:
    message = message.ljust(LCD_WIDTH," ")
  elif style==2:
    message = message.center(LCD_WIDTH," ")
  elif style==3:
    message = message.rjust(LCD_WIDTH," ")
```

```
  for i in range(LCD_WIDTH):
    lcd_byte(ord(message[i]),LCD_CHR)

def lcd_byte(bits, mode):
  # Send byte to data pins
  # bits = data
  # mode = True  for character
  #        False for command


  GPIO.output(LCD_RS, mode) # RS

  # High bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x10==0x10:
    GPIO.output(LCD_D4, True)
  if bits&0x20==0x20:
    GPIO.output(LCD_D5, True)
  if bits&0x40==0x40:
    GPIO.output(LCD_D6, True)
  if bits&0x80==0x80:
    GPIO.output(LCD_D7, True)


  # Toggle 'Enable' pin
  time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  time.sleep(E_DELAY)


  # Low bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x01==0x01:
    GPIO.output(LCD_D4, True)
  if bits&0x02==0x02:
    GPIO.output(LCD_D5, True)
  if bits&0x04==0x04:
    GPIO.output(LCD_D6, True)
  if bits&0x08==0x08:
    GPIO.output(LCD_D7, True)
```

```
  # Toggle 'Enable' pin
  time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  time.sleep(E_DELAY)


if __name__ == '__main__':
  main()
```