

Textul si imaginile din acest document sunt licentiate

Attribution-NonCommercial-NoDerivs  
CC BY-NC-ND



Codul sursa din acest document este licentiat

Public-Domain

Esti liber sa distribui acest document prin orice mijloace consideri (email, publicare pe website / blog, printare, sau orice alt mijloc), atat timp cat nu aduci nici un fel de modificari acestuia. Codul sursa din acest document poate fi utilizat in orice fel de scop, de natura comerciala sau nu, fara nici un fel de limitari.

## Raspberry PI si LCD Shield 20x4

Shield-ul LCD 20x4 functioneaza perfect impreuna cu placa Raspberry PI. Shield-ul se poate conecta direct la portul placii sau prin intermediul unei panglici cobbler. Placa expune in acelasi timp si toti pinii GPIO ai Raspberry PI (conectorul 2X13, nepopulat, de langa LCD). In acest mod, poti avea acces la oricare pin GPIO doresti si in acelasi timp poti folosi si LCD-ul.

In acest tutorial vei utiliza shield-ul LCD 20x4 pentru a afisa data/ora, adresa IP a placii, temperatura si nivelul de utilizare al procesorului. Pentru a scoate in evidenta utilitatea placii Raspberry PI, programul sintetizeaza vocal parametrii de mai sus.

Pentru acest tutorial ai nevoie de urmatoarele componente:

- O placa Raspberry PI.
- Un shield LCD 20x4.
- O boxa audio cu alimentare prin USB.
- O sursa de alimentare pentru Raspberry PI (5V).

### Cum procedez ?

1. Conecteaza shield-ul LCD la placa Raspberry PI.
2. Conecteaza mufa USB a boxei la unul din porturile USB ale placii.
3. Conecteaza mufa jack a boxei la iesirea audio a placii Raspberry.
4. Alimenteaza Raspberry PI si logheaza-te la placa prin Putty sau orice alt utilitar doresti.

### Pachetul Python RPi.GPIO 0.4.1a.

Inainte de a executa programul listat mai jos trebuie sa instalezi pachetul python RPi.GPIO 0.4.1a. Programul shield-ului depinde de acest pachet.

1. Descarca modulul prin comanda:

<http://www.robofun.ro/forum>

```
sudo wget  
https://pypi.python.org/packages/source/R/RPi.GPIO/RPi.GPIO-  
0.4.1a.tar.gz
```

2. Dezarhiveaza-l prin comanda:

```
sudo tar -xvf RPi.GPIO-0.4.1a.tar.gz
```

3. Schimba locatia in noul fisier:

```
cd RPi.GPIO-0.4.1a
```

4. Urmeaza sa il instalezi prin comanda:

```
sudo python setup.py install
```

## **Programul shield-ului.**

1. Creeaza un fisier cu numele lcd20x4.

```
sudo mkdir lcd20x4
```

2. Deschide editorul nano si copiaza codul sursa listat mai jos. Dupa ce l-ai copiat, salveaza-l cu CTRL X si Y:

```
sudo nano lcd.py
```

3. Executa programul cu comanda:

```
sudo python lcd.py
```

4. Pe afisaj vei obtine urmatoarele linii, ca in imaginile de mai jos, iar sintetizorul va reda vocal parametrii:



```
import os
import subprocess
import RPi.GPIO as GPIO
import time
from datetime import datetime
import socket
import fcntl
import struct
```

```
LCD_RS = 25
LCD_E  = 24
LCD_D4 = 23
LCD_D5 = 17
LCD_D6 = 18
LCD_D7 = 22
LED_ON = 15
```

```
LCD_WIDTH = 20
LCD_CHR = True
LCD_CMD = False
```

```
LCD_LINE_1 = 0x80
LCD_LINE_2 = 0xC0
LCD_LINE_3 = 0x94
LCD_LINE_4 = 0xD4
```

```
E_PULSE = 0.00005
E_DELAY = 0.00005
```

```
def main():
```

```
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)
    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)
    GPIO.setup(LED_ON, GPIO.OUT)

    lcd_init()

    while 1:
        lcd_byte(LCD_LINE_1, LCD_CMD)
        lcd_string("ROBOFUN.RO", 2)
        lcd_byte(LCD_LINE_2, LCD_CMD)
        lcd_string("Raspberry PI", 2)
        lcd_byte(LCD_LINE_3, LCD_CMD)
        lcd_string("LCD Shield", 2)
        lcd_byte(LCD_LINE_4, LCD_CMD)
        lcd_string("20x4", 2)
        os.system('espeak -v en "{0}"'.format('www.robofun.ro
introducing'))
        os.system('espeak -v en "{0}"'.format('the new Raspberry PI'))
        os.system('espeak -v en "{0}"'.format('20 characters by 4 lines
LCD Shield'))
```

```

for i in range(0,50):
    lcd_byte(LCD_LINE_1, LCD_CMD)
    textTime = datetime.now().strftime('%b %d %H:%M')
    lcd_string(textTime, 2)

    lcd_byte(LCD_LINE_2, LCD_CMD)
    textIP = get_ip_add('eth0')
    lcd_string(textIP, 2)

    lcd_byte(LCD_LINE_3, LCD_CMD)
    textTemp = 'CPU Temp %s C' % (get_cpu_temp())
    lcd_string(textTemp, 2)

    lcd_byte(LCD_LINE_4, LCD_CMD)
    textCpuUse = 'CPU Use %s' % (get_cpu_use()) + '%'
    lcd_string(textCpuUse, 2)

    os.system('espeak -v en "{0}"'.format('The temperature is %s
degrees Celsius' % (get_cpu_temp())))
    os.system('espeak -v en "{0}"'.format('IP adress is %s' %
textIP))
    os.system('espeak -v en "{0}"'.format('The procesor is being
used at %s percent' % get_cpu_use()))
    os.system('espeak -v en "{0}"'.format('The time is %s' %
textTime))

    time.sleep(0.2)

```

```

def get_cpu_use():
    return(str(os.popen("top -n1 | awk '/Cpu\(s\):/ {print $2}'").readline().strip()))

def get_cpu_temp():
    res = os.popen('vcgencmd measure_temp').readline()
    return (res.replace("temp=", " ").replace("'C\n", ""))

def get_ip_add(iframe):
    s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return
socket.inet_ntoa(fcntl.ioctl(s.fileno(), 0x8915, struct.pack('256s',
iframe[:15])))[20:24])

def lcd_init():
    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):
    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):
    GPIO.output(LCD_RS, mode)
    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)

    time.sleep(E_DELAY)
    GPIO.output(LCD_E, True)
    time.sleep(E_PULSE)
    GPIO.output(LCD_E, False)
    time.sleep(E_DELAY)
    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)

    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()

```