

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.

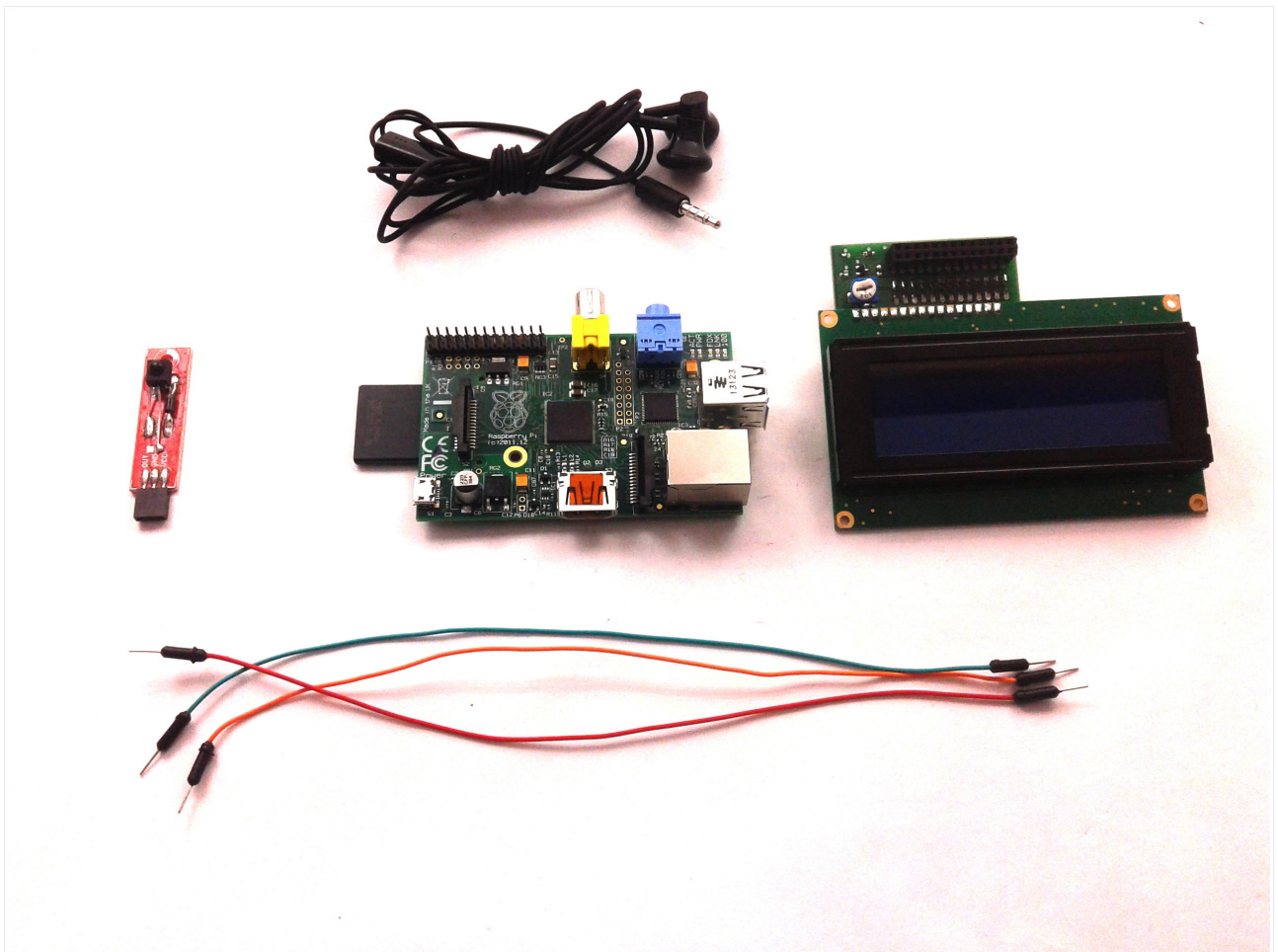
## Raspberry PI – Internet Radio cu shield LCD 20x4

În partea a doua a tutorialului, vei descoperi cum se poate adauga un shield LCD20x4 care va afisa informatii cu privire la postul de radio: nume, melodie, timpul de redare, volum, etc.

Informatiile sunt redade printr-un program Python care verifica, în același timp, datele postului de radio și actualizează informațiile pe ecranul LCD-ului.

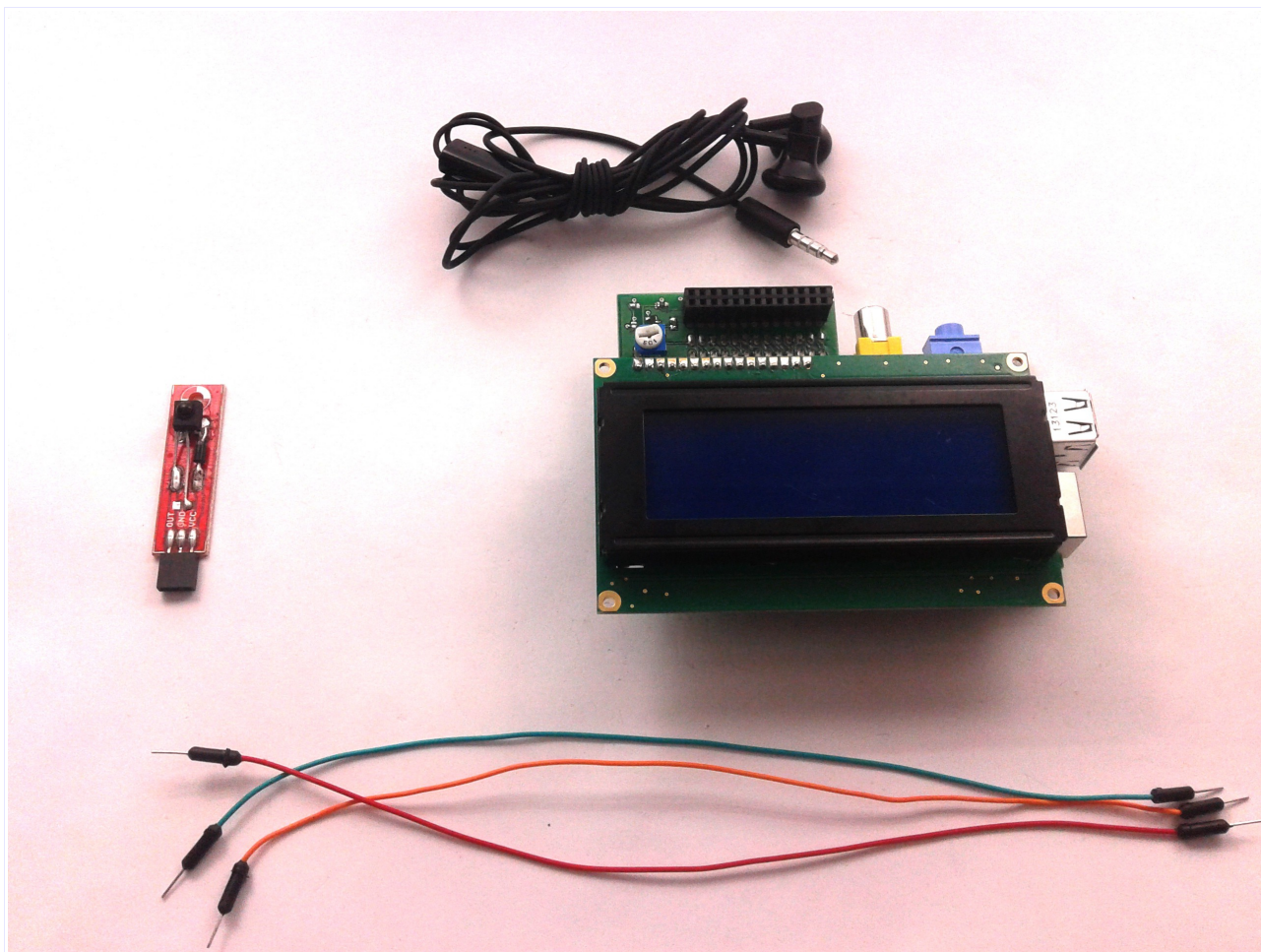
Vei avea nevoie de un shield LCD 20x4 pentru Raspberry PI:

[http://www.robofun.ro/shield-lcd-raspberry-pi-20x4?keyword=lcd&category\\_id=0](http://www.robofun.ro/shield-lcd-raspberry-pi-20x4?keyword=lcd&category_id=0)



Primul pas este să conectezi shield-ul LCD la portul GPIO al plăcii Raspberry PI. Shield-ul se înfige pur și simplu în port. Mufa 2x13 care permite conectarea este deja inclusă și lipită pe shield.

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

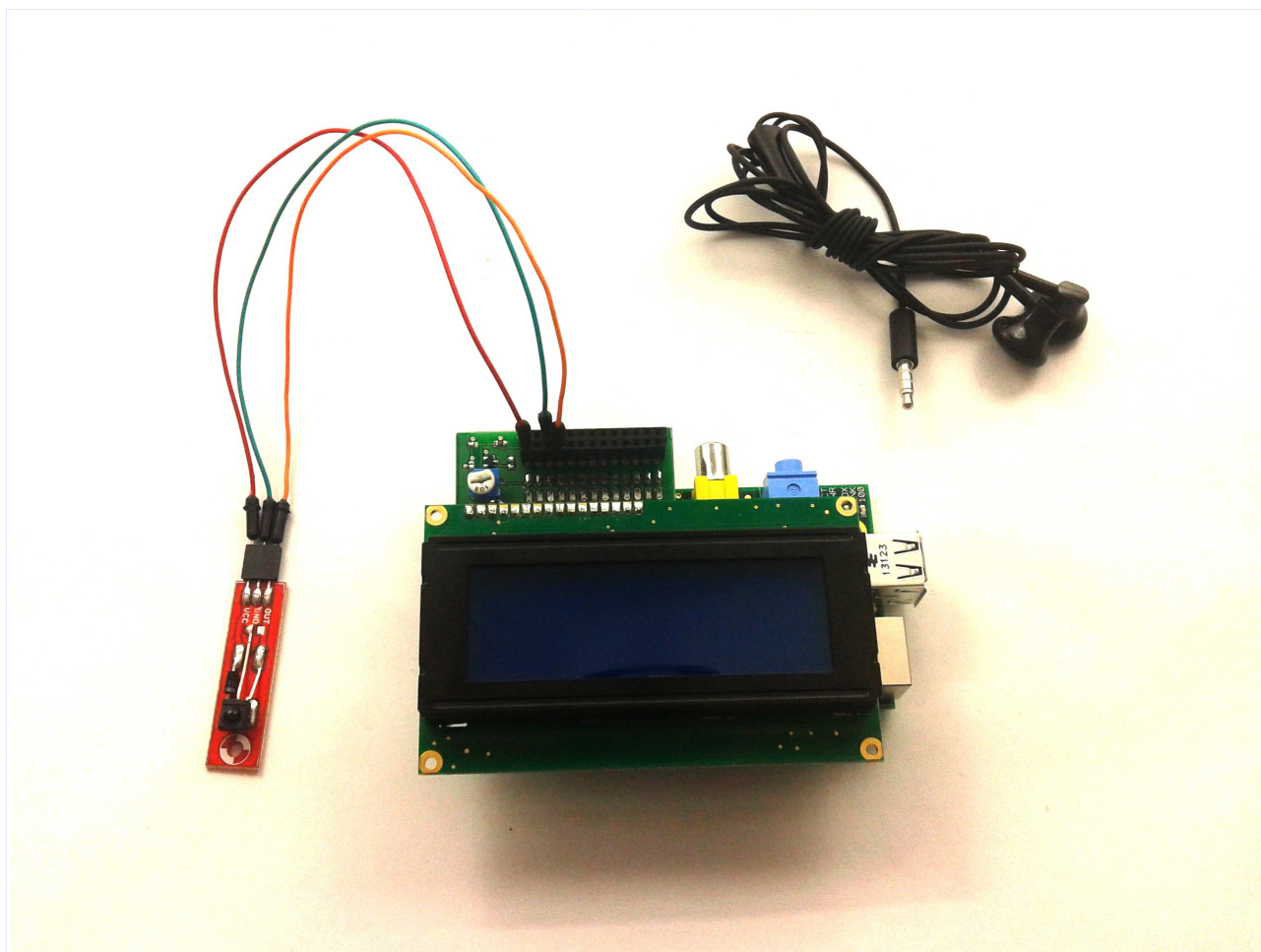


Placa expune in acelasi timp si toti pinii GPIO ai Raspberry PI (conectorul 2x13 care se populeaza la cerere). Este necesar sa ai acces la port, pentru ca in urmatorul pas vei conecta senzorul de telecomanda brick.

Ca si in tutorialul precedent, senzorul de telecomanda se conecteaza la portul GPIO in urmatoorii pini:

<b>Raspberry PI 3.3V</b>	<b>Senzor telecomanda brick VCC</b>
<b>Raspberry PI GND</b>	<b>Senzor telecomanda brick GND</b>
<b>Raspberry PI GPIO4</b>	<b>Senzor telecomanda brick OUT</b>

De aceasta data, vei conecta senzorul in mufa 2x13 de pe shield care respecta aceeasi ordine a pinilor ca si portul GPIO. Pentru a te orienta cat mai bine urmeaza imaginea de mai jos.



Tot ce iti ramane este sa conectezi castile audio in mufa jack a placii, alimentatorul de 5V si cablul de retea. Poti conecta o pereche de casti (nivelul audio este suficient de mare) sau o pereche de boxe.

La aceasta data, versiunea de Raspbian nu necesita nicio modificare cu privire la setarile placii de sunet. Setarile sunt default si sunetul functioneaza fara probleme.

Daca doresti mobilitate poti opta pentru un acumulator cu incarcare solara:

<http://www.robofun.ro/raspberry-pi-si-componente/acumulator-incarcator-usb-solar>

si un conector WI-PI:

<http://www.robofun.ro/raspberry-pi-si-componente/oficial-wifi-raspberry-pi-wi-pi>

Partea de conectare hardware este gata, urmeaza sa copiezi si sa executi codul Python listat mai jos. Daca doresti ca programul sa se execute automat la pornire, trebuie sa deschizi cu editorul nano urmatorul fisier:

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



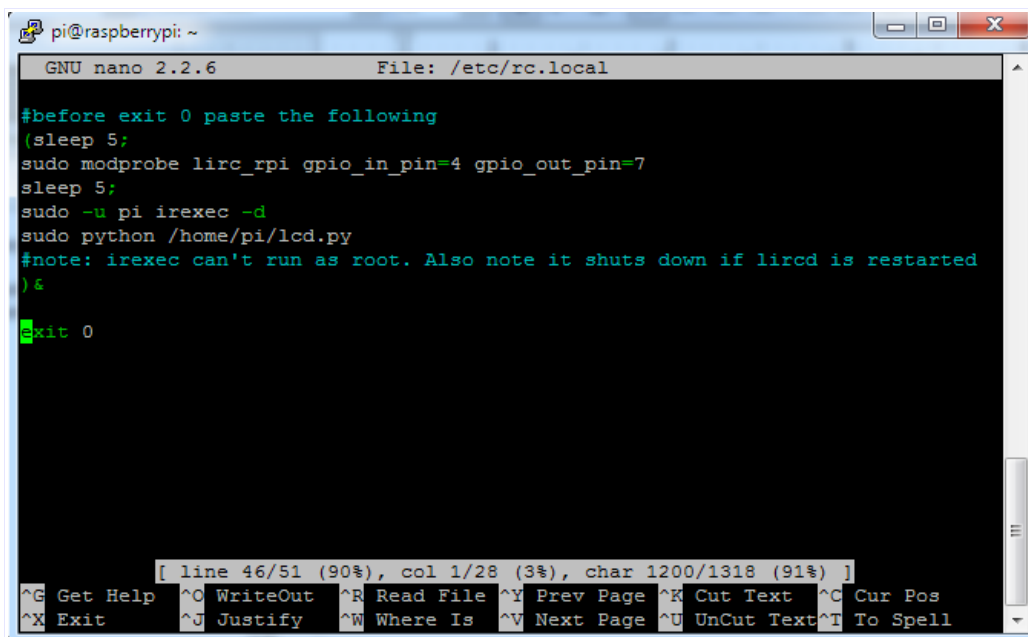
```
sudo nano /etc/rc.local
```

Adauga urmatoarea linie:

```
sudo python /home/pi/lcd.py
```

imediat dupa linia:

```
sudo -u pi irexec -d
```



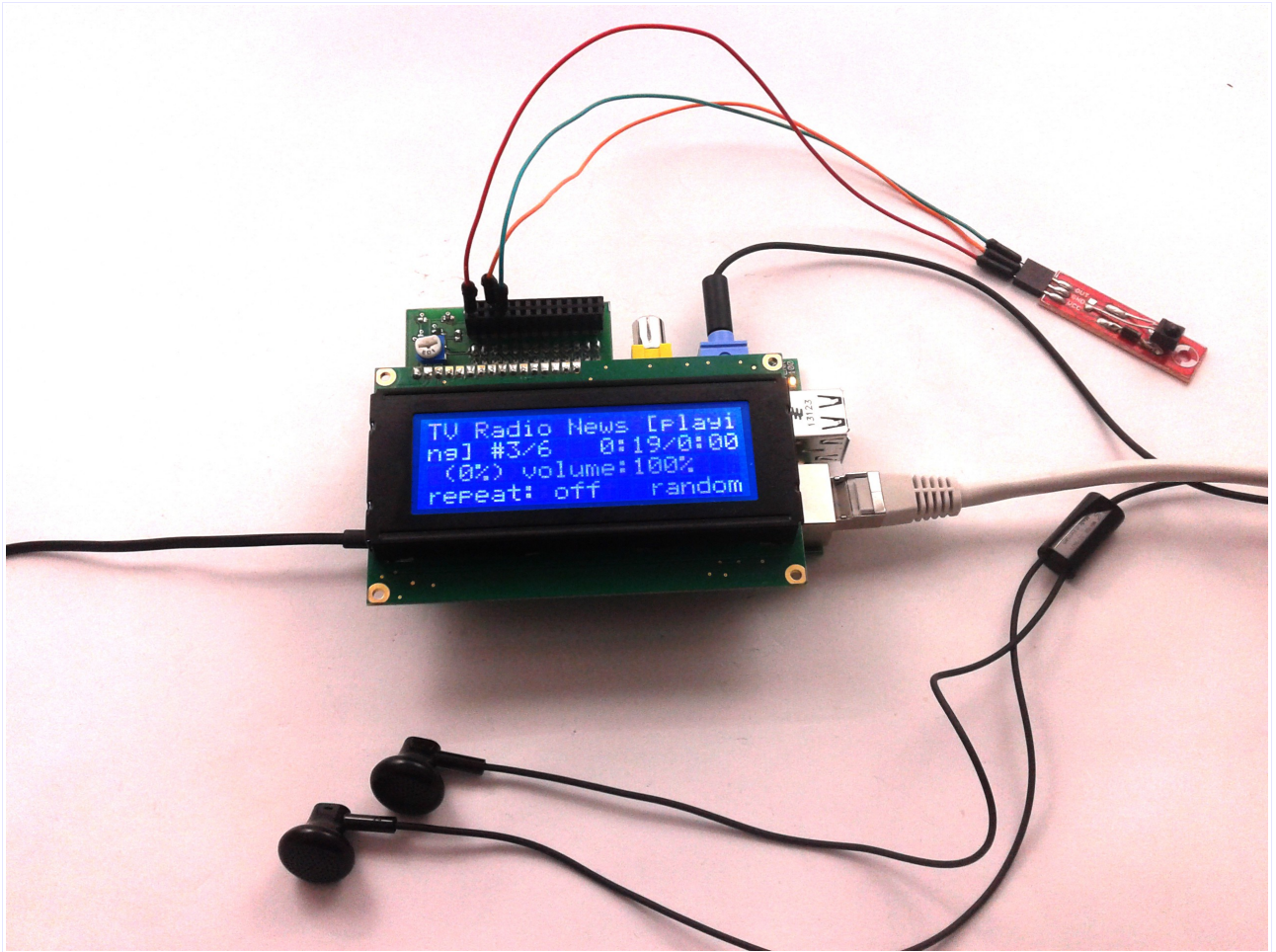
```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/rc.local

#before exit 0 paste the following
(sleep 5;
sudo modprobe lirc_rpi gpio_in_pin=4 gpio_out_pin=7
sleep 5;
sudo -u pi irexec -d
sudo python /home/pi/lcd.py
#note: irexec can't run as root. Also note it shuts down if lircd is restarted
)&

exit 0

[ line 46/51 (90%), col 1/28 (3%), char 1200/1318 (91%) ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Executa un restart, iar dupa cateva momente iti va aparea pe ecranul lcd-ului postul de radio detaliat. Textul va defila in asa fel incat sa poti citi mai multe informatii (in mod normal acestea nu incap pe toata suprafata lcd-ului).



```
#!/usr/bin/python

import RPi.GPIO as GPIO
import time
import sys
from subprocess import check_output

stringToDivide = "Hello"

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.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:
        output = check_output(["mpc", "play"])
        output = output.replace('\n', ' ')
        dim = len(output)
        for x in range(0, dim-80):
            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string(output[x+0:x+20],1)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string(output[x+20:x+40],1)
            lcd_byte(LCD_LINE_3, LCD_CMD)
            lcd_string(output[x+40:x+60],1)
            lcd_byte(LCD_LINE_4, LCD_CMD)
            lcd_string(output[x+60:x+80],1)

            checkOutput = check_output(["mpc", "play"])
            checkOutput = checkOutput.replace('\n', ' ')

            if checkOutput[0:10] == output[0:10]:
                print "ok"
            else:
                break
            time.sleep(0.4)

        time.sleep(1)

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):
    # 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()
```