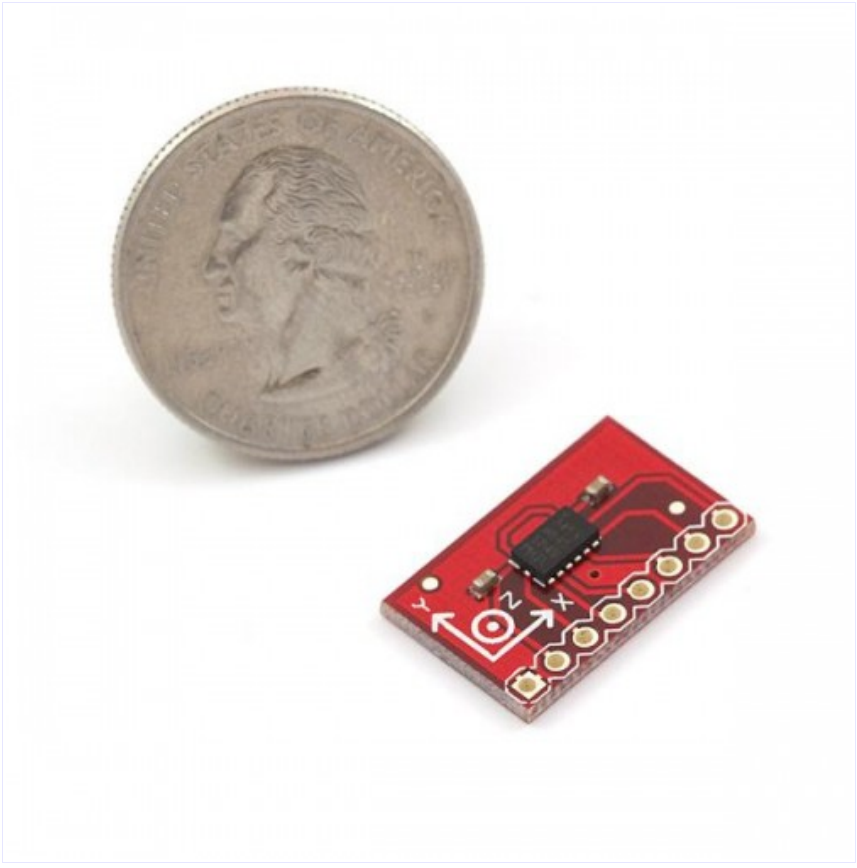
RaspberryPI si ADXL345

ADXL345 este un accelerometru de dimensiuni mici, are un consum redus de energie iar masuratorile efectuate pe cele 3 axe au o rezolutie mare (13 biti). Poate masura pana la ± 16g, iar datele sunt reprezentate in format digital, la alegere, prin interfata SPI sau I2C.

ADXL345 este foarte potrivit pentru masuratori ale acceleratiei statice a gravitatiei in aplicatii care sesizeaza bascularea, dar si acceleratia dinamica rezultata din miscare sau socuri. Accelerometrul are o rezolutie mare (4 mg/LSB) si permite masurarea schimbarilor de inclinatie mai mici de 1,0°.

Mai multe functii de sesizare speciale sunt furnizate. Sesizarea activitatii si inactivitatii depisteaza prezenta sau lipsa miscarii si daca acceleratia pe oricare axa excede un nivel setat de catre utilizator. Sesizarea batailor usoare depisteaza bataile simple sau duble. Sesizarea caderii libere depisteaza daca senzorul se afla se afla in cadere. Aceste functii pot fi mapate pe unul din doi pini de iesire de intrerupere. Un buffer first in, first out (FIFO) cu 32 de nivele integra poate fi folosit pentru a stoca datele pentru a minimiza interventia procesorului cu care comunica senzorul.

Senzorul poate functiona si in moduri cu consum redus de energie.



In acest tutorial vei programa placa Raspberry PI sa afiseze pe un shield LCD 20x4 acceleratiile corespunzatoare celor 3 axe.

Vei avea nevoie de urmatoarele componente:

- O placa Raspberry PI.
- Un shield LCD 20x4 pentru Raspberry PI.
- Un senzor ADXL345.
- Fire pentru conexiuni.
- Breadboard.
- O sursa de alimentare pentru Raspberry PI (5V)

Cum conectez shield-ul LCD si senzorul ADXL345 ?

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.

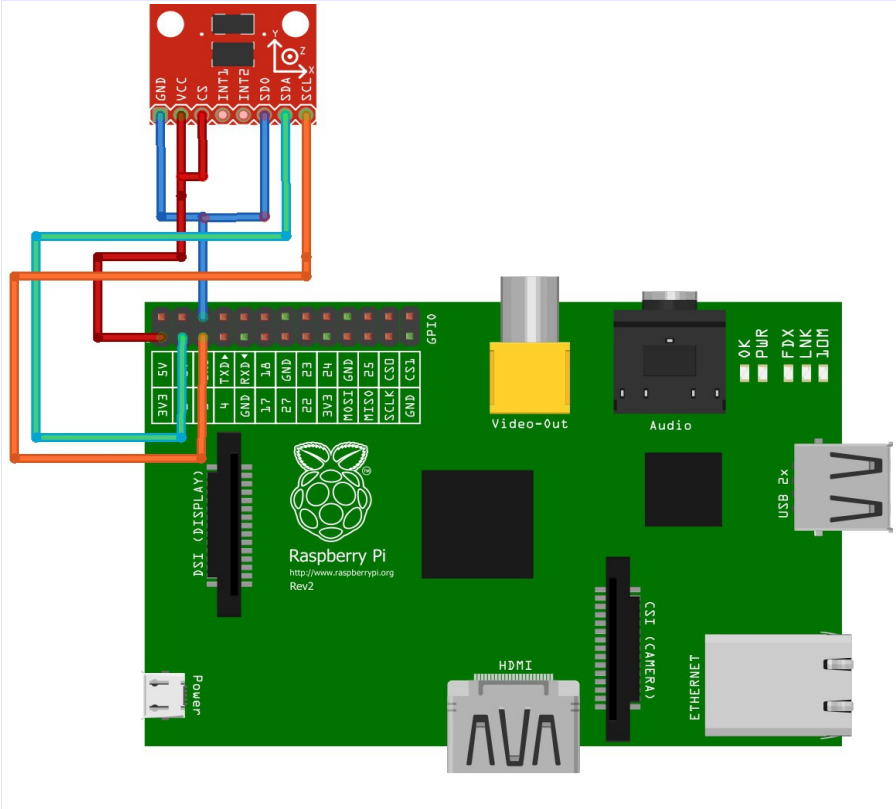
2. Asigura-te ca bareta mama 2x13 este lipita pe shield, altfel nu poti conecta senzorul ADXL345. La achizitionarea shield-ului poti opta pentru shield cu bareta lipita.

3. Conecteaza accelerometrul la placa Raspberry PI conform tabelului urmator:

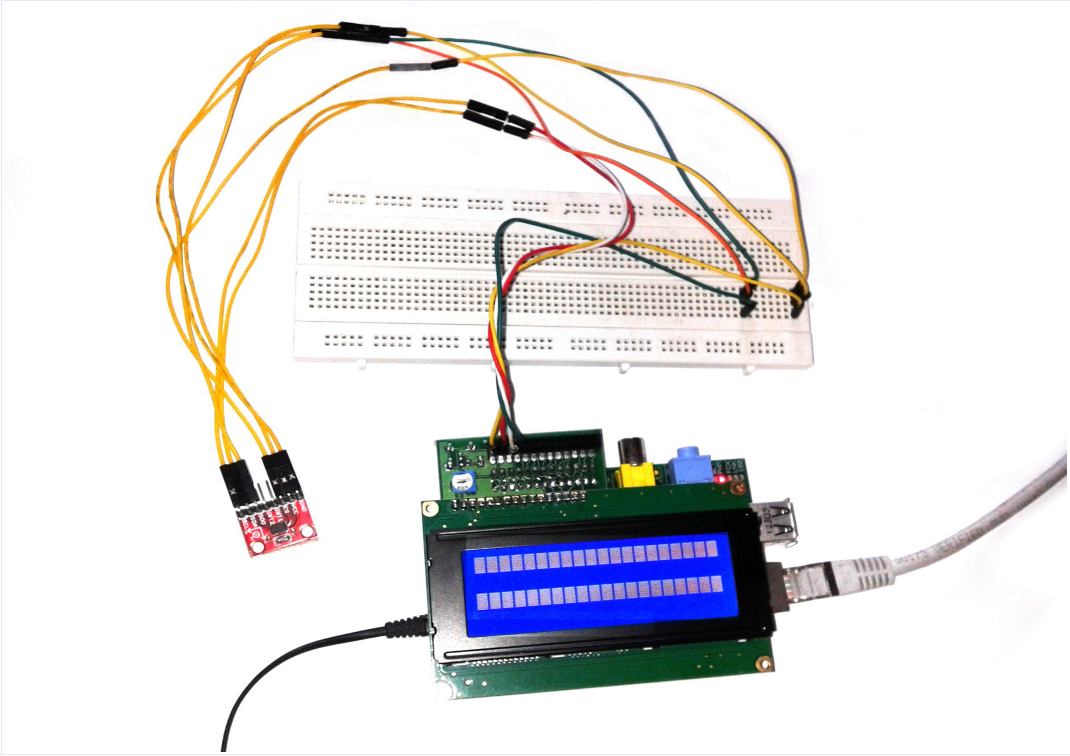
Raspberry PI 3.3V	ADXL345 VCC
Raspberry PI 3.3V	ADXL345 CS
Raspberry PI GND	ADXL345 GND
Raspberry PI GND	ADXL345 SDO
Raspberry PI SDA	ADXL345 SDA
Raspberry PI SCL	ADXL345 SCL

4. Foloseste, ca referinta, schema portului GPIO si schema electronica de mai jos:

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



Dupa realizarea conexiunilor, vei obtine urmatoarea imagine.



Cum programez placuta Raspberry PI ?

Mai jos sunt listate 2 programe, dintre care, primul se executa cu Python si se ocupa cu initializarea afisajului LCD si afisarea acceleratiilor pe fiecare rand iar cel de-al doilea program este scris in limbajul C si se ocupa cu initializarea interfetei I2C a senzorului si citirea acceleratiilor din el. In mod normal vei executa doar programul scris in C, deoarece acesta realizeaza apel automat catre programul Python.

Codul sursa este listat mai jos si il poti copia cu copy/paste, dar inainte de asta sunt necesare cateva configurari:

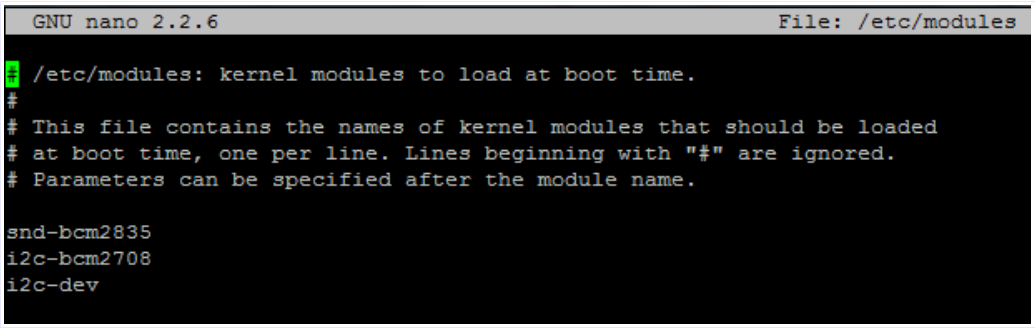
1. Deschide urmatorul fisier si asigurate ca ai cele 2 linii:

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

Liniile:

i2c-bcm2708

i2c-dev



2. Executa o restartare cu:

sudo reboot

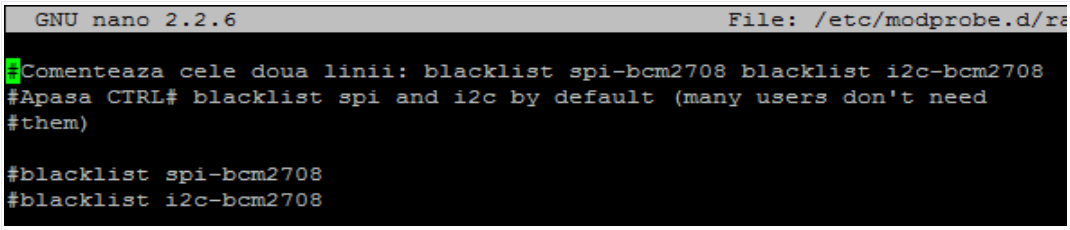
3. Dupa restartare, executa cele 2 comenzi:

sudo apt-get install python-smbus

sudo apt-get install i2c-tools

4. Deschide fisierul cu urmatoarea comanda si asigura-te ca este prezent cate un # pe fiecare linie:

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



5. Verifica daca senzorul este detectat corect prin comanda:

sudo i2cdetect -y 1

Comanda returneaza toate adresele detectate. Daca senzorul este detectat corect, atunci el va

raporta adresa 0x53.

6. Descarca si instaleaza pachetul python RPi.GPIO 0.4.1a. Programul Python depinde de acest pachet.

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

7. Dezarhiveaza-l prin comanda:

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

8. Schimba locatia in noul fisier:

```
cd Rpi.GPIO-0.4.1a
```

9. Executa instalarea:

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

Codul sursa ?

1. Creeaza un fisier cu editorul nano si copiaza codul Python. Inchide editorul si salveaza-l cu CTRL X si Y.

```
sudo nano lcd.py
```

2. Creeaza un fisier cu acelasi editor si copiaza codul in limbajul C. Inchide editorul in aceeasi maniera:

```
sudo nano readingADXL345.c
```

3. Compileaza programul C cu comanda:

```
gcc -o read_adxl readingADXL345.c
```

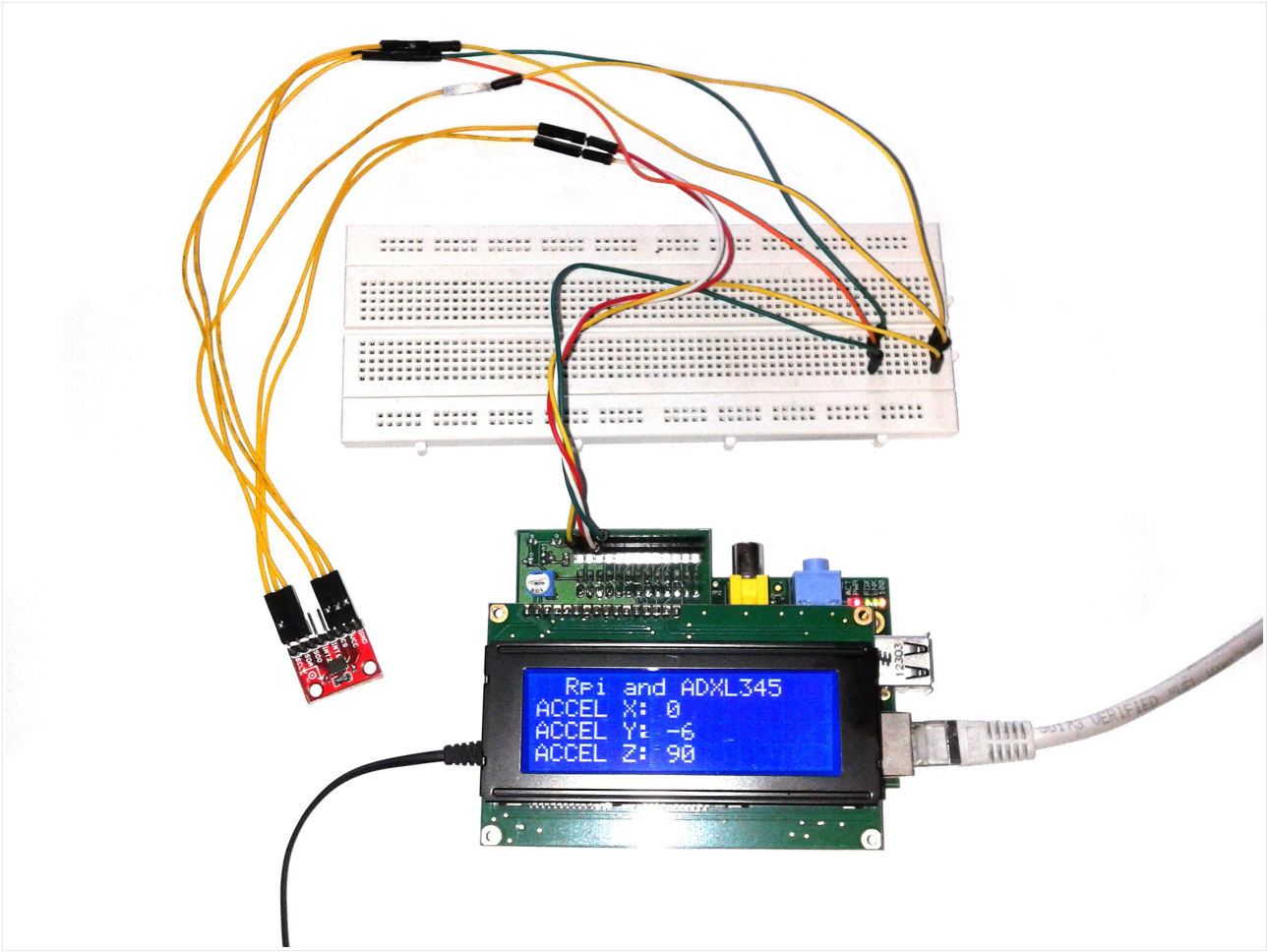
4. Dupa compilare vei obtine un executabil denumit read_adxl.

```
pi@raspberrypi ~/adxl345 $ ls -l
total 16
-rwxr-xr-x 1 root root 2759 Aug 13 19:01 lcd.py
-rwxr-xr-x 1 pi   pi   7303 Aug 13 19:03 read_adxl
-rw-r--r-- 1 root root 2293 Aug 13 19:03 readingADXL345.c
pi@raspberrypi ~/adxl345 $
```

5. Porneste programul care citeste si afiseaza senzorul prin comanda:

`./read_adxl`

Vei obtine urmatoarele imagini:



```
pi@raspberrypi ~/adxl345 $ ls -l
total 16
-rwxr-xr-x 1 root root 2759 Aug 13 19:01 lcd.py
-rwxr-xr-x 1 pi   pi   7303 Aug 13 19:03 read_adxl
-rw-r--r-- 1 root root 2293 Aug 13 19:03 readingADXL345.c
pi@raspberrypi ~/adxl345 $ ./read_adxl
   0    0    0
   5   -2   91
   6   -1   92
   7   -1   92
   6   -1   91
   5   -2   90
   6   -2   91
   4   -2   92
   4   -2   91
   5   -2   91
   5   -2   92
   4   -1   91
   0   -6   91
 -15   -4  117
   0   -6   91
  -0   -7   91
```

Programul Python.

```
from datetime import datetime
import RPi.GPIO as GPIO
import time
import os
import subprocess
import sys

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():

 textLineTwo = "ACCEL X: " + str(int(float(sys.argv[1])))
 textLineThree = "ACCEL Y: " + str(int(float(sys.argv[2])))
 textLineFour = "ACCEL Z: " + str(int(float(sys.argv[3])))

 lcd_init()
 lcd_byte(LCD_LINE_1, LCD_CMD)
 lcd_string("Rpi and ADXL345",2)
 lcd_byte(LCD_LINE_2, LCD_CMD)
 lcd_string(textLineTwo,1)
 lcd_byte(LCD_LINE_3, LCD_CMD)
 lcd_string(textLineThree,1)
 lcd_byte(LCD_LINE_4, LCD_CMD)
 lcd_string(textLineFour,1)

def lcd_init():
  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)

  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()
```

Programul C.

```
// gcc -o read_adxl readingADXL345.c

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <linux/i2c-dev.h>

#define ADXL345_I2C_ADDR 0x53
```

```
void selectDevice(int fd, int addr, char * name)
{
   if (ioctl(fd, I2C_SLAVE, addr) < 0)
   {
      fprintf(stderr, "%s not present\n", name);
      //exit(1);
   }
}

void writeToDevice(int fd, int reg, int val)
{
   char buf[2];
   buf[0]=reg; buf[1]=val;
   if (write(fd, buf, 2) != 2)
   {
      fprintf(stderr, "Can't write to ADXL345\n");
      //exit(1);
   }
}

int main(int argc, char **argv)
{
   unsigned int range;
   int count, b;
   short x, y, z;
   float xa, ya, za;
   int fd;
   unsigned char buf[16];
   char str[80] = {};
   char system_str[256] = {};

   if ((fd = open("/dev/i2c-1", O_RDWR)) < 0)
   {
      // Open port for reading and writing
      fprintf(stderr, "Failed to open i2c bus\n");
      exit(1);
   }
   /* initialise ADXL345 */

   selectDevice(fd, ADXL345_I2C_ADDR, "ADXL345");

   writeToDevice(fd, 0x2d, 0);
   writeToDevice(fd, 0x2d, 16);
   writeToDevice(fd, 0x2d, 8);
   writeToDevice(fd, 0x31, 0);
   writeToDevice(fd, 0x31, 11);

   while (1) {
```

```
      /* select ADXL345 */

      selectDevice(fd, ADXL345_I2C_ADDR, "ADXL345");

      buf[0] = 0x32;
      if ((write(fd, buf, 1)) != 1)
      {
         // Send the register to read from
         fprintf(stderr, "Error writing to i2c slave\n");
         //exit(1);
      }
      if (read(fd, buf, 6) != 6)
      {
         //  X, Y, Z accelerations

         fprintf(stderr, "Unable to read from ADXL345\n");
         //exit(1);
      }
      else
      {
         x = buf[1]<<8| buf[0];
         y = buf[3]<<8| buf[2];
         z = buf[5]<<8| buf[4];
         xa = (90.0 / 256.0) * (float) x;
         ya = (90.0 / 256.0) * (float) y;
         za = (90.0 / 256.0) * (float) z;
         //printf("x=%d, y=%d, z=%d\n", x, y, z);
         printf("%4.0f %4.0f %4.0f\n", xa, ya, za);
         //for (b=0; b<6; b++) printf("%02x ",buf[b]);
printf("\n");
         sprintf(system_str, "sudo python lcd.py %f %f %f", xa, ya,
za);
         system( system_str );

      }
//      usleep(9000);
   }
   return 0;
}
```