```
#!/usr/bin/python
import spidev
import time
import os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

# Define GPIO to LCD mapping
LCD_RS = 15
LCD_E  = 16
LCD_D4 = 7
LCD_D5 = 11
LCD_D6 = 12
LCD_D7 = 13

# Define sensor channels
temp_channel  = 0
'''
define pin for lcd
'''
# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005
delay = 1
```

```
GPIO.setup(LCD_E, GPIO.OUT)  # E
GPIO.setup(LCD_RS, GPIO.OUT) # RS
GPIO.setup(LCD_D4, GPIO.OUT) # DB4
GPIO.setup(LCD_D5, GPIO.OUT) # DB5
GPIO.setup(LCD_D6, GPIO.OUT) # DB6
GPIO.setup(LCD_D7, GPIO.OUT) # DB7

# Define some device constants
LCD_WIDTH = 16    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False
LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line

'''
Function Name :lcd_init()
Function Description : this function is used to initialized lcd by sending the different commands
'''
def lcd_init():
  # Initialise display
  lcd_byte(0x33,LCD_CMD) # 110011 Initialise
  lcd_byte(0x32,LCD_CMD) # 110010 Initialise
  lcd_byte(0x06,LCD_CMD) # 000110 Cursor move direction
  lcd_byte(0x0C,LCD_CMD) # 001100 Display On,Cursor Off, Blink Off
  lcd_byte(0x28,LCD_CMD) # 101000 Data length, number of lines, font size
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  time.sleep(E_DELAY)
'''
Function Name :lcd_byte(bits ,mode)
```

Fuction Name :the main purpose of this function to convert the byte data into bit and send to lcd port

'''

```
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
 lcd_toggle_enable()

 # 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
  lcd_toggle_enable()
'''
Function Name : lcd_toggle_enable()
Function Description:basically this is used to toggle Enable pin
'''
def lcd_toggle_enable():
  # Toggle enable
  time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  time.sleep(E_DELAY)
'''
Function Name :lcd_string(message,line)
Function  Description :print the data on lcd
'''
def lcd_string(message,line):
  # Send string to display
```

```
  message = message.ljust(LCD_WIDTH," ")

  lcd_byte(line, LCD_CMD)

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




# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
def ReadChannel(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0])
  data = ((adc[1]&3) << 8) + adc[2]
  return data


# Function to calculate temperature from
# TMP36 data, rounded to specified
# number of decimal places.
def ConvertTemp(data,places):

  # ADC Value
  # (approx)  Temp  Volts
  #    0      -50    0.00
  #   78      -25    0.25
  #  155        0    0.50
  #  233       25    0.75
  #  310       50    1.00
  #  465      100    1.50
```

```
 #  775      200    2.50
 # 1023      280    3.30

 temp = ((data * 330)/float(1023))
 temp = round(temp,places)
 return temp



# Define delay between readings
delay = 5
lcd_init()
lcd_string("welcome ",LCD_LINE_1)
time.sleep(2)
while 1:
 temp_level = ReadChannel(temp_channel)
 temp       = ConvertTemp(temp_level,2)

 # Print out results
 lcd_string("Temperature  ",LCD_LINE_1)
 lcd_string(str(temp),LCD_LINE_2)
 time.sleep(1)
```