```
# -*- coding: utf-8 -*-
import MySQLdb as mdb
import sys
import time
import os
import lcddll
from time import sleep
from smbus import SMBus

# weerrobot6.py
# measuring Temperature, barometric pressure, light and hymidity
# schrijf gegevens naar mysql database weerrobot op 192.168.1.1
# max. 7 data waarden met bijbehorende tijdwaarneming

#=========================================
#========== init some values/parameters ============
#=======================================
b = SMBus(1)
var = [0,0,0,0]
# This is our array for the sensor data
ansa = bytearray()

os.environ['TZ'] = 'Europe/Amsterdam'
now = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
db = "true"
#try:
#    con = mdb.connect('192.168.1.1', 'weerrobot', 'weerrobot', 'weerrobot')
#     db="true"
#     print "Contact met JVS1..Mysql"
#    print now
#except ValueError:
#    print 'Er is geen verbinding met JVS'
#   db="false"

var1= str(1)
var2= str(2)
var3= str(3)
var4= str(4)
var5= str(5)
var6= str(6)
var7= str(7)

#=========================================
#============== connect mysql database     ===========
#=========================================

def writeTodatabase():
   now = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
   try:
     con = mdb.connect('192.168.1.1', 'weerrobot', 'weerrobot', 'weerrobot')
     db="true"
     with con:
```

```
            cur = con.cursor()
            cur.execute("INSERT INTO metingen(tijd,Data1,Data2,Data3,Data4,Data5,Data6, Data7) \
                        VALUES (%s,%s,%s,%s,%s,%s,%s,%s)" , ( now,
var1,var2,var3,var4,var5,var6,var7))
            print now, "Data weggeschreven JVS1 in database weerrobot"
            cur.close()
   except ValueError:
    print 'Er is geen verbinding met JVS'
    db="false"


#=========================================
#==============lcd display     =
#=========================================
# zie datasheet
# you can check the address with $ i2cdetect -y 1
lcd1 = lcddll.lcd(0x27,1)
lcd1.lcd_write(0x01);
s0 = "ZO weerrobot 6!"
s1= "  Temperature"
s2= "   Air Pressure"
s3= "     Light"
s4= "   Humidity"
s5="     "             #invoegen tab
s6= "              "   #lege regel 16 spaties

lcd1.lcd_puts("Hallo Jesse",1) #display  Pi" on line 1
lcd1.lcd_puts(s0,2) #display "Take a byte!" on line 2
#lcd1.lcd_puts(" works on all 4",3) #display "Raspberry Pi" on line 1
#lcd1.lcd_puts("Lines of the display",4) #display "Take a byte!" on line 2
lcd1.lcd_backlight(1)
lcd1.lcd_backlight(1)
sleep(0.5)
#lcd1.lcd_backlight(0)

#=========================================
#==============HYT221 humidity and temperature sensor     =
#=========================================
# zie datasheet HYT221
# you can check the address with $ i2cdetect -y 1
add = 0x28
# Create a smbus object on I2C bus #1
hyt = SMBus(1)

# Init the array just to make sure we have 4 bytes available -
# I'm paranoid!
ansa.append(0x30)
ansa.append(0x31)
ansa.append(0x32)
ansa.append(0x33)

# Init HYT 221 for reading, ignore answer
```

```
ans = hyt.read_byte_data(add,0)

def getHumidity():
         # Read 4 bytes (even more) of data from HYT221
        ansa = hyt.read_i2c_block_data(add,4)
         # Now we have all data from sensor in the first 4 bytes of 'ansa'
         # Look for HYT221 doc to see how to
        # extract temperature and humidity from bytes
       #Calc humidity in rel.%
        hum = ansa[0]<<8 | ansa[1]
        hum = hum & 0x3FFF
        var5 = 100.0*hum/(2**14)
        return var5

def getTHumidity():
         # Read 4 bytes (even more) of data from HYT221
        ansa = hyt.read_i2c_block_data(add,4)
        # Calc temperature in °C
        ansa[3] = ansa[3] & 0x3F
        temp = ansa[2] << 6 | ansa[3]
        var6 = 165.0*temp/(2**14)-40
        return var6

#=========================================
#==============TSL2561 light/lux sensor     ============
#=========================================
# zie datasheet TSL2561
TSLAddress = 0x39
b.write_byte_data(TSLAddress,0x80,0x03) # enable
#b.write_byte_data(TSLAddress,0x81,0x02) # set low gain
b.write_byte_data(TSLAddress,0x81,0x12) # set highgain

def getLight():
        var = b.read_i2c_block_data(0x39, 0xAC,4)
        var3 = ((var[1]<<8) + var[0])  # light
        return var3

def getLightIR():
        var = b.read_i2c_block_data(0x39, 0xAC,4)
        var4 = ((var[3]<<8) + var[2])  # IR
        return var4
#=========================================
#==============T5403 barometer en temperature sensor ===
#=========================================
slaveAddress = 0x77
#registers
T5403_COMMAND_REG = 0xf1
T5403_DATA_REG_LSB = 0xf5
T5403_DATA_REG_MSB = 0xf6
#commands
COMMAND_GET_TEMP = 0x03
#definitions for pressure reading commands with accuracy modes
```

```
MODE_LOW = 0x00
MODE_STANDARD = 0x01
MODE_HIGH = 0x10
MODE_ULTRA = 0x11


global c1,c2,c3,c4,c5,c6,c7,c8
def uint16Toint16(data):
    if data > 32767:
        return data - 0x10000
    else:
        return data
def sendCommand(cmd):
    return (b.write_byte_data(slaveAddress,T5403_COMMAND_REG,cmd))
def getData():
    return (b.read_byte_data(slaveAddress,T5403_DATA_REG_MSB)<<8) +
b.read_byte_data(slaveAddress,T5403_DATA_REG_LSB)
def begin():
    global c1,c2,c3,c4,c5,c6,c7,c8
    c1 = (b.read_byte_data(slaveAddress,0x8f)<<8) + b.read_byte_data(slaveAddress,0x8e)
    c2 = (b.read_byte_data(slaveAddress,0x91)<<8) + b.read_byte_data(slaveAddress,0x90)
    c3 = (b.read_byte_data(slaveAddress,0x93)<<8) + b.read_byte_data(slaveAddress,0x92)
    c4 = (b.read_byte_data(slaveAddress,0x95)<<8) + b.read_byte_data(slaveAddress,0x94)
    c5 = (b.read_byte_data(slaveAddress,0x97)<<8) + b.read_byte_data(slaveAddress,0x96)
    c6 = (b.read_byte_data(slaveAddress,0x99)<<8) + b.read_byte_data(slaveAddress,0x98)
    c7 = (b.read_byte_data(slaveAddress,0x9b)<<8) + b.read_byte_data(slaveAddress,0x9a)
    c8 = (b.read_byte_data(slaveAddress,0x9d)<<8) + b.read_byte_data(slaveAddress,0x9c)
    c5 = uint16Toint16(c5)
    c6 = uint16Toint16(c6)
    c7 = uint16Toint16(c7)
    c8 = uint16Toint16(c8)
def getTemperature():
    sendCommand(COMMAND_GET_TEMP)
    sleep(0.006)
    temp_raw = uint16Toint16(getData())
    temp_actual = ((( c1 * temp_raw)/ 0x100) + ( c2 * 0x40)) * 100) / 0x10000;
    return float(temp_actual)/100.0
def getPressure(precision):
    sendCommand(COMMAND_GET_TEMP)
    sleep(0.006)
    temp_raw = uint16Toint16(getData())
    #Load measurement noise level into command along with start command bit.
    precision = (precision << 3)|(0x01)
    # Start pressure measurement
    sendCommand(precision)
    if precision == MODE_LOW:
        sleep(0.005)
    elif precision == MODE_STANDARD:
        sleep(0.011)
    elif precision == MODE_HIGH:
        sleep(0.019)
    elif precision == MODE_ULTRA:
```

```
            sleep(0.067)
        else:
            sleep(0.1)
        pressure_raw = getData()
        # calculate pressure
        s = ((((( c5 * temp_raw) >> 15) * temp_raw) >> 19) + c3 + (( c4 * temp_raw) >> 17);
        o = ((((( c8 * temp_raw) >> 15) * temp_raw) >> 4) + (( c7 * temp_raw) >> 3) + (c6 * 0x4000);
        return ((s * pressure_raw + o) >> 14)
#================================================
def displayOnLcd():
    #lcd1.lcd_write(0x01);
    displayClear()
    lcd1.lcd_puts(s2,1) #display "pressure" on line 1
    lcd1.lcd_puts(s5 + str(var2),2)
    sleep(1)
    displayClear()
    lcd1.lcd_puts(s3,1) #display "light" on line 1
    lcd1.lcd_puts(s5 + str(var3),2)
    sleep(1)
    displayClear()
    lcd1.lcd_puts(s4,1) #display "humidity" on line 1
    lcd1.lcd_puts(s5 + str(round(var5,1)),2)
    sleep(1)
    displayClear()
    lcd1.lcd_puts(s1,1) #display "temperature" on line 1
    lcd1.lcd_puts(s5 + str(var1),2)
    sleep(1)
    return

def displayTimeOnLCD():
    displayClear()
    lcd1.lcd_puts("Hello world",1) #display on line 1
    t0 = time.strftime('%H:%M:%S %d-%m-%Y', time.localtime())
    lcd1.lcd_puts(t0,2)   #display date/time  on line 2
    sleep(2)
    return

def displayErrorOnLcd():
    displayClear()
    lcd1.lcd_puts("!@!@ ERROR @!@!",1) #display on line 1
    lcd1.lcd_puts("Database server not present",2)
    sleep(2)
    return

def displayClear():
    lcd1.lcd_puts(s6,1) #display "pressure" on line 1
    lcd1.lcd_puts(s6,2)
    sleep(2)
    return

#================== MAIN LOOP ==================
var = 1
```

```
t = time.time()

while var == 1:
    begin()

    displayTimeOnLCD():

    var1=getTemperature()
    var2=getPressure(MODE_ULTRA)
    var3=getLight()
    var4=getLightIR()
    var5=getHumidity()
    var6=getTHumidity()
    print 'Temperature=',var1
    print 'Air Pressure=',var2 # mBar
    print 'Light=',var3,' IR=', var4
    print 'Humidity=',round(var5,2)
    #print 'Temperature=',var6

    displayOnLcd()

    if db == "false":
       displayErrorOnLcd()

    if time.time() - t > 60*10
       t=0
       writeTodatabase()
```