```
#!/usr/bin/python3
import struct
import smbus2
from smbus2 import i2c_msg
import numpy as np
import time
from collections import namedtuple

class SCD30():
    def __init__(self, measurement_intervalx, auto_calibrationx, pressurex):
        super(SCD30, self).__init__()

        self.i2c_address = 0x61
        self.bus_number = smbus2.SMBus(1)

        self.set_measurement_interval(measurement_intervalx)
        time.sleep(0.1)
        self.set_auto_calibration(auto_calibrationx)
        time.sleep(0.1)
        self.start_periodic_measurement(pressurex)
        time.sleep(0.1)

    # Set measurement interval
    def set_measurement_interval(self, interval):
        try:
            cmd_set_measurement_interval = self.create_cmd(interval, 0x00)
            #print ("cmd_set_measurement_interval=%s"%cmd_set_measurement_interval)
            self.bus_number.write_i2c_block_data(self.i2c_address, 0X46,
            cmd_set_measurement_interval)
        except OSError as e:
            print("set_measurement_interval: " + str(e))

    # Start periodic measurement
    def start_periodic_measurement(self, pressure):
        try:
            cmd_start_periodic_measurement = self.create_cmd(pressure, 0x10)
            #print ("cmd_start_periodic_measurement=%s"%cmd_start_periodic_measurement)
            self.bus_number.write_i2c_block_data(self.i2c_address, 0X00,
            cmd_start_periodic_measurement)
            time.sleep(0.1)
            #print("%7.2f hPa 2" % pressure)
        except OSError as e:
            print("start_periodic_measurement: " + str(e))

    # Set auto calibration
    def set_auto_calibration(self, on_off):
        try:
            cmd_set_auto_calibration = self.create_cmd(on_off, 0x06)
            #print ("cmd_set_auto_calibration=%s"%cmd_set_auto_calibration)
            self.bus_number.write_i2c_block_data(self.i2c_address, 0x53,
            cmd_set_auto_calibration)
        except OSError as e:
            print("set_auto_calibration: " + str(e))

    # Get Data ready
    def data_ready(self):
        try:
            self.bus_number.write_byte_data(self.i2c_address, 0x02, 0x02)
            time.sleep(0.1)
            data_ready = i2c_msg.read(self.i2c_address, 3)
            time.sleep(0.1)
            self.bus_number.i2c_rdwr(data_ready)

            return bool(np.array(list(data_ready))[1])
        except OSError as e:
            print("OSError in scd30 data_ready: " + str(e))
            return False

    # Read measurement buffer
    def get_scd30_measurements(self, ):
        try:
            self.bus_number.write_byte_data(self.i2c_address, 0x03, 0x00)
```

```
            time.sleep(0.1)

            measurement = i2c_msg.read(self.i2c_address, 18)
            self.bus_number.i2c_rdwr(measurement)

            co2_list = [0, 1, 2, 3, 4, 5]
            co2_measurement = self.get_measurement(measurement, co2_list)
            #print("co2_measurement=%s"%co2_measurement)

            temperature_list = [6, 7, 8, 9, 10, 11]
            temperature_measurement = self.get_measurement(measurement, temperature_list)
            #print("temperature_measurement=%s"%temperature_measurement)

            humidity_list = [12, 13, 14, 15, 16, 17]
            humidity_measurement = self.get_measurement(measurement, humidity_list)
            #print("humidity_measurement=%s"%humidity_measurement)

            if co2_measurement and humidity_measurement and temperature_measurement:
                return self.to_name_tuple(co2_measurement, humidity_measurement,
                temperature_measurement)
            else:
                return None

        except OSError as e:
            print("OSError in scd30 get_scd30_measurements: " + str(e))
            return None

    def extract_measurement(self, i, measurement):
        return np.array(list(measurement))[i]

    def pack_struct(self, i, i1, i2, i3, measurement):
        return struct.pack('4B', self.extract_measurement(i, measurement), self.
        extract_measurement(i1, measurement),
                           self.extract_measurement(i2, measurement), self.
                           extract_measurement(i3, measurement))

    def crc8_update(self, b, crc):
        crc = crc ^ b
        for i in range(8):
            if (crc & 0x80) == 0x80:
                crc = (crc << 1) ^ 0x131
            else:
                crc <<= 1
        return crc


    def calc_crc(self, d):
        crc = self.crc8_update((d >> 8) & 0x00FF, 0xff)
        crc = self.crc8_update((d & 0x00FF), crc)
        return crc

    def get_crc(self, i, j, measurement):
        crc = self.crc8_update(np.array(list(measurement))[i], 0xff)
        crc = self.crc8_update(np.array(list(measurement))[j], crc)
        return crc

    def get_measurement(self, measurement, val_list):
        crc = self.get_crc(val_list[0], val_list[1], measurement)
        unpacked_measurement = None

        if crc == self.extract_measurement(val_list[2], measurement):
            crc = self.get_crc(val_list[3], val_list[4], measurement)
            if crc == self.extract_measurement(val_list[5], measurement):
                tmp = self.pack_struct(val_list[0], val_list[1], val_list[3], val_list[4],
                measurement)
                unpacked_measurement = struct.unpack('>f', tmp)
        return unpacked_measurement

    def create_cmd(self, payload, x_):
        ff = 0x00FF
        i = 8
        return [x_, (payload >> i) & ff, (payload & ff), self.calc_crc(payload)]
```

```
    def to_name_tuple(self, co2_measurement, humidity_measurement, temperature_measurement):
        Data = namedtuple('Data', ['CO2', 'temperature', 'humidity'])
        return Data(co2_measurement[0], temperature_measurement[0], humidity_measurement[0])

scd30 = SCD30(5, 0, 1013)
while True:
    try:
        if scd30.data_ready():
            dados=scd30.get_scd30_measurements()
        time.sleep(5)
    except (KeyboardInterrupt, SystemExit):
        print("Stopped")
        exit("Tchau")
    except:
        print("Erro SCD30!")
        time.sleep(5)
    else:
        if dados is not None:
            print("SCD30: CO2=%d ppm, Temperatura=%0.1f ºC, Humidade=%0.1f %%"%(dados[0],
            dados[1],dados[2]))
```