```
//INCLUDES
#include <SPI.h> //Library for SPI communication with SD logging shield
#include <Wire.h> //Library for general I2C communication with CO2, p, T and rH
  sensors
#include "Adafruit_BMP085.h" //Library for communication with BMP10 p,T sensor
#include "RTClib.h" //Library for communication with the data loggers on board
  realt time clock (RTC)
#include <SD.h> //Library for communication with SD-card

#define HEADER "Date\tTime\tt[ms]\tCO2[ppm]\tSO2[ppm]\tT[C]\tPressure[Pa]
  \tHumidity[%]" //defines the header of the data file
#define CO2_ADDR 0x69 //adresses of all I2C devices
#define SHT_ADDR 0x40
#define BMP_ADDR 0x77
#define RTC_ADDR 0x77
#define SO2Pin A1 //analog input pin for reading of the SO2 sensors signal
#define GLEDPin 6 //digital output pins to control front panel LED's
#define RLEDPin 5

//CREATE NECESSARY CLASS INSTANCES
Adafruit_BMP085 bmp; //BMP180
File Data; //Logfile
RTC_DS1307 RTC; //Real-Time-Clock

//DATA VARIABLES
float Temperature, Humidity, SO2;
long Pressure;
int CO2;
long startTime = 0;
bool error = 0;

//INITIALIZATION
void setup(){

    pinMode(RLEDPin, OUTPUT);
    pinMode(GLEDPin, OUTPUT);
    pinMode(SO2Pin, INPUT);
    analogReference(INTERNAL);
    Serial.begin(9600);
    Wire.begin();

    if(!SD.begin(10)){ //Is the SD card inserted? If not, don't start
      measurements, instead make the red LED blink forever.
        Serial.print("SD-Card missing!");
        while(1){
            digitalWrite(RLEDPin, HIGH);
            delay(200);
            digitalWrite(RLEDPin, LOW);
            delay(300);
        }
    }

    RTC.begin();
```

```
    //RTC.adjust(DateTime(__DATE__, __TIME__)); //Uncomment this line and upload
       the sketch to set the RTC to the compilation time of the sketch

    //Initialize the data file on the SD card
    Data = SD.open("Data.txt", FILE_WRITE);
    Data.println(HEADER);

    bmp.begin();

    delay(50);

    //Initial check if all sensors are properly connected
    Wire.beginTransmission(CO2_ADDR);
    if(Wire.endTransmission()) error = 1;
    Wire.beginTransmission(SHT_ADDR);
    if(Wire.endTransmission()) error = 1;
    Wire.beginTransmission(BMP_ADDR);
    if(Wire.endTransmission()) error = 1;
    Wire.beginTransmission(RTC_ADDR);
    if(Wire.endTransmission()) error = 1;

    //if not, give a warning signal by shortly flashing the red front panel LED
    if(error){
        for(int i=0; i<4; i++){
            digitalWrite(RLEDPin, HIGH);
            delay(200);
            digitalWrite(RLEDPin, LOW);
            delay(200);
        }
    }
}

//ACTUAL MEASUREMENT ROUTINE
void loop(){

    startTime = millis();

    Temperature = bmp.readTemperature(); //read temperature from BMP180
    Pressure = bmp.readPressure(); //read pressure from BMP180
    Humidity = getSHTHumidity(); //read relative huimidity from SHT21
    CO2 = readCO2(); //read uncalibrated CO2 value from K30
    SO2 = getSO2(100); //read SO2 sensor uncalibrated analog signal, the
      parameter is the number analog readouts to average over
    printDataToSD(); //save data on SD card
    //printDataToSerial(); //optionally send data to arduino serial port

    //acknowledge a successful data retrieval by a flash of the green front
      panel LED
    digitalWrite(GLEDPin, HIGH);
    delay(40);
    digitalWrite(GLEDPin, LOW);
    //wait until it's time for the next datapoint
    while((millis()-startTime)<500);
```

```
}


//Procedure, which logs the data on SD card
void printDataToSD(){
    DateTime now = RTC.now();
    Data = SD.open("Data.txt", FILE_WRITE);
    Data.print(now.year(), DEC);
    Data.print('/');
    Data.print(now.month(), DEC);
    Data.print('/');
    Data.print(now.day(), DEC);
    Data.print(' ');
    Data.print(now.hour(), DEC);
    Data.print(':');
    Data.print(now.minute(), DEC);
    Data.print(':');
    Data.print(now.second(), DEC);
    Data.print('\t');
    Data.print(millis());
    Data.print('\t');
    Data.print(CO2);
    Data.print('\t');
    Data.print(SO2);
    Data.print('\t');
    Data.print(Temperature);
    Data.print('\t');
    Data.print(Pressure);
    Data.print('\t');
    Data.println(Humidity);
    Data.close();
}

//Procedure, which sends the sensor data to the Arduinos serial port
void printDataToSerial(){
    DateTime now = RTC.now();
    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.print('\t');
    Serial.print(millis());
    Serial.print('\t');
    Serial.print(CO2);
    Serial.print('\t');
```

```
    Serial.print(SO2);
    Serial.print('\t');
    Serial.print(Temperature);
    Serial.print('\t');
    Serial.print(Pressure);
    Serial.print('\t');
    Serial.println(Humidity);
}

//procedure to read relative humidity from the SHT21
float getSHTHumidity(){
    //initialize measurement
    Wire.beginTransmission(0x40);
    Wire.write(B11100101);
    Wire.endTransmission();
    Wire.requestFrom(0x40, 3);
    //wait for measurement
    delay(40);
    //read result
    long result = (Wire.read()<<8);
    result += Wire.read();
    Wire.read(); //read away the checksum
    result &= 0xFFFC; //mask status bits
    return 125*((float)result/pow(2,16))-6.0; //conversion to actual percent rH
      value
}


//procedure to read raw CO2 value from the K30 sensor
int readCO2(void)
{
    int CO2_value = 0;
    byte i = 0;
    byte buffer[4] = {0,0,0,0};
    byte sum = 0;

    do {
        //request value from sensor
        Wire.beginTransmission(CO2_ADDR);
        Wire.write(0x22);
        Wire.write(0x20);
        Wire.write(0x08);
        Wire.write(0x4A);
        Wire.endTransmission();
        //wait and receive response
        delay(50);
        Wire.requestFrom(CO2_ADDR, 4);
        delay(1);
        while(Wire.available()){
            buffer[i] = Wire.read();
            i++;
        }
        //convert digital value to
```

```
        CO2_value = 0;
        CO2_value |= buffer[1] & 0xFF;
        CO2_value = CO2_value << 8;
        CO2_value |= buffer[2] & 0xFF;
        sum = buffer[0] + buffer[1] + buffer[2]; //Byte addition utilizes
          overflow

    } while (sum != buffer[3]);
    return CO2_value;
}

//procedure to read analog signal of the SO2 sensor:
float getSO2(int meanOver){
    float SO2Value = 0;
    //calculate average over several readouts to minimize noise
    for(int i=0; i<meanOver; i++){
        SO2Value += analogRead(SO2Pin);
    }
    SO2Value /= meanOver;
    return SO2Value;
}
```