# Air Quality @ Meteo Data via LoRaWAN

DuPe (www.dupedup.cz)
Prototype Version: 11/2024-1

**Purpose:**

Measure Air Quality data (PM2.5 and PM10) and meteo data (temperature, humidity, pressure). Data is periodicaly sent via LoRaWAN network to data server at provider LoRaWAN. Received data is POSTed to web server into the CSV file. Data is readed by WEB browser at user and is displayed as graph.

**Hardware components:**

### 1) MCU Cube Cell HTTC AB02 + Antenna 868 MHz (Heltec Automation)

- Master Chip ASR6502 (48 MHz ARM® Cortex® M0+ MCU)
- Memory 128KB internal FLASH; 16KB internal SRAM
- LoRa Chipset SX1262
- Frequency 863~923 MHz
- Max TX Power 22 ± 1 dBm
- Receiving sensitivity -135 dBm
- Voltage Input Solar Energy 5.5～7V solar panel, Lithium Battery 3.7V, VIN 5V, VDD 3.3V
- UART x 2; SPI x 1; I2C x 1; SWD x 1; 12-bits ADC input x 3; 8–channel DMA engine; GPIO x 16
- Interface Micro USB x 1; LoRa Antenna interface(IPEX), Battery 3.7V Lithium (SH1.25 x 2 socket）
- Low Power Deep Sleep 3.5μA
- Onboard Display Size 0.96-inch OLED (controller SH110)
- USB to Serial Chip CP2102)
- Dimensions 51.9 x 25 x 8 mm

### 2) SENSOR DUST Laser PM2.5/10 SDS011 (Nova Fitness Co)

- Air quality sensor, designed to measure particulate matter in the air
- Measurement parameters PM2.5,PM10
- Range 0.0-999.9μg/m
- Digital interface: UART (9600/N/1)
- Analog interface
- Voltage 5V / 70mA±10mA / fan+laser sleep <4mA
- Size 71x70x23mm

### 3) SENSOR DIGITAL HUMIDITY, PRESSURE AND TEMPERATURE BME280

- Digital interface $I^2C$ (up to 3.4 MHz)
- Supply voltage VDD main supply voltage range: 1.71 V to 3.6 V
- Operating range temp: -40 +85°C, + 1°C, 0.01° C
- Operating range press: 300-1100hPa, ± 1hPa, 0.18Pa
- Operating range humi: 0 -100% RH, + 3% , 0,008%

### 4) DC-DC Step UP Modul ( MT3608)

- Input from Li-ion ACU Voltage 3.2 – 4.2V
- Output to senzor SHS011 Voltage 5V / min 250 mA

### 5) Li-ion Cylindrical Battery (GEB Model 18650)

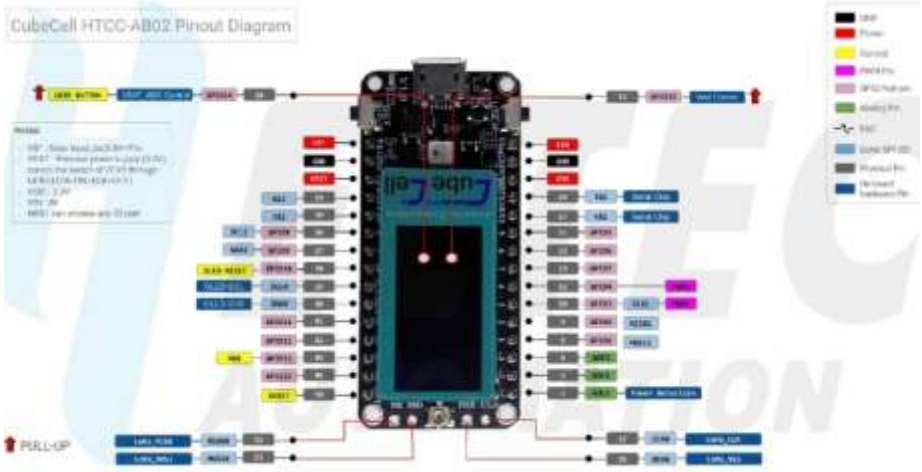
- Nominal Voltage: 3.7V
- Nominal Capacity : 3000mAh / 11.1 Wh
- Standard discharge current: 600 mA
- Size: d ≥18.5mm, h≥65.5 / 56

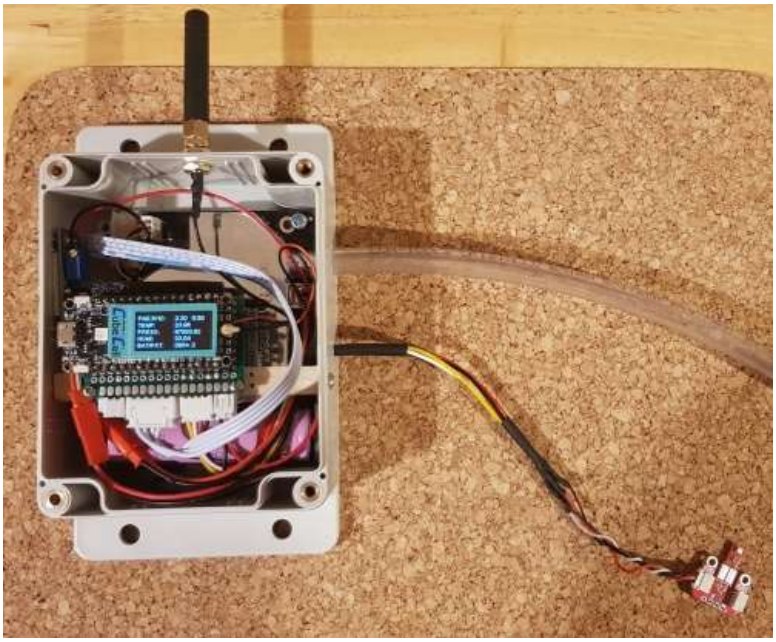
### 6) Solar Panel 6V

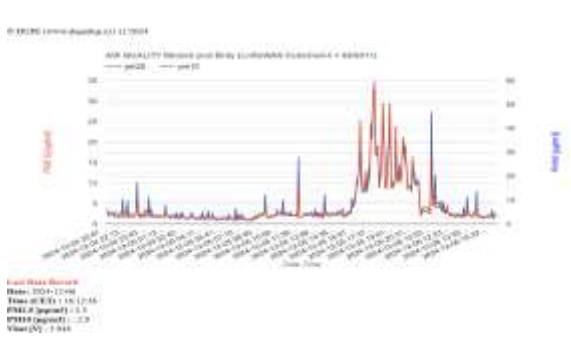
- Output min 300mA (2W)
- Size 110x136x3mm / 52g

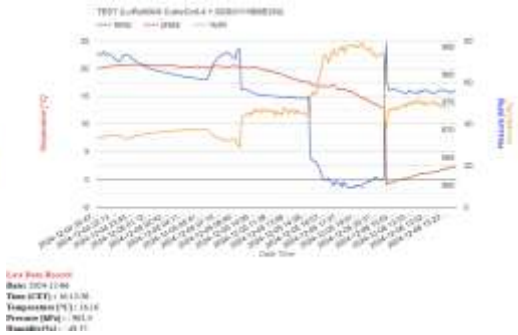
**Circuit:**

| **Cube Cell Modul** (PIN 5V tolerant) | **DUST SENZOR** (UART) | **BME SENZOR** (I2C) | **STEP UP** | **Other** |
|---|---|---|---|---|
| 29 UART BUS RX2 | TX | | | |
| 30 UART BUS TX2 | RX | | | |
| 36 I2C BUS SCL1 | | SCL | | |
| 37 I2C BUS SDA1 | | SDA | | |
| VDD (3.3 V OUT) | | VCC | | |
| Battery Connector | | | INPUT | ACU LiION 1S |
| VS ( 5.5 - 7V) | | | | Solar Panel 6V |
| GND | GND | GND | | GND ACU/Solar |
| | 5V | | OUTPUT 5V | |







Current consumption of the device is about 150 mA in the working mode and 20 mA in the sleeping mode. In case of steady operation of the device (e.g during nightime without solar light on the panel) is required interruption voltage of the dust senzor during the sleeping time. It can be used a DC StepUp modul with a digital switch controlled by level of GPIO MCU.

## Processing:

AIR PM/TEMPERATURE/HUMIDITY/PRESSURE - > SENSOR -> SENSOR REGISTRY -> SENSOR BUS (I2C) -> MCU DATA BUS (I2C) -> MCU (STM32 ) -> MCU DATA BUS -> OLED (I2C) -> LORA RADIO chip (SPI) -> RADIO 868MHz -> LORAWAN GW (UP TO 10 km) -> LORAWAN SERVER REST API -> DATA POST -> WWW SERVER (PHP CODE) -> DATA FILE
DATA FILE -> HTML/JS CODE ( API GOOGLE CHARTS ) - > WWW USER BROWSER

Data : 26 B
Cycle: 300s
LoRa: 868 MHz , SF 12, BW 125, DR about 300 bps
LoRaWAN: OTTA parameters (devEUI, apEUI, appKEY) were generated on The Things Network
LoRaWAN Provider: CeskeRadiokomunikace Datové centrum, přenos dat, Cloud a IoT | České Radiokomunikace

DUST SENZOR: REPORTING mode (query report)- workng time: 30 sec (delay MCU), sleeping time 300 sec (MCU is sleeping)
BME SENZOR: FORCED mode (perform one measurement, store results and return to sleep mode)

Dust Senzor SDS011 Digital Protocol via UART (Number of bytes Name Content):
0 Messageheader AA
1 CommanderNo. C0
2 DATA1 PM2.5Lowbyte
3 DATA2 PM2.5Highbyte
4 DATA3 PM10Lowbyte
5 DATA4 PM10Highbyte
6 DATA5 IDbyte1
7 DATA6 IDbyte2
8 Check-sum Check-sum
9 Messagetail AB

## Software Components:

- C/C++ VSCode / Platform IO
- GitHub - HelTecAutomation/CubeCell-Arduino: Heltec CubeCell Series (based on ASR6501, ASR6502 chip) Arduino support.
- https://github.com/adafruit/Adafruit_BME280_Library
- https://github.com/lewapek/sds-dust-sensors-arduino-library Library for Nova Fitness SDS dust sensors family (SDS011, SDS021) - If used software serial API - must be replaced lib Arduino **SofwareSerial** in lewapek all header or source cpp files with Heltec lib **softSerial** . Can be used hardware serial API also.

## Code:

**platformio.ini**
[env:cubecell_board_plus]
platform = heltec-cubecell

```
board = cubecell_board_plus
framework = arduino
monitor_speed = 115200
lib_deps = adafruit/Adafruit BME280 Library@^2.2.4
build_flags =
   -D ACTIVE_REGION=LORAMAC_REGION_EU868
   -D REGION_EU868
   -D CubeCell_BoardPlus
   -D LORAWAN_CLASS=CLASS_A
   -D LORAWAN_DEVEUI_AUTO=0
   -D LORAWAN_NETMODE=true
   -D LORAWAN_ADR=true
   -D LORAWAN_UPLINKMODE=true
   -D LORAWAN_Net_Reserve=false
   -D LORAWAN_AT_SUPPORT=0
   -D LORAWAN_PREAMBLE_LENGTH=8
   -D LORAWAN_DebugLevel=2
   -D RGB=1
```

**setup-lora.h**

```
#define AT_SUPPORT  0
#define RGB 1
#define DELAYTIME 1000
#define ACTIVE_REGION LORAMAC_REGION_EU868
#define CLASS_MODE CLASS_A
#define LORAWAN_DEVEUI_AUTO 0
#define CubeCell_BoardPlus
#define REGION_EU868
#define LORAWAN_NETMODE true
#define LORAWAN_ADR true
#define LORAWAN_UPLINKMODE true
#define LORAWAN_Net_Reserve false
#define LORAWAN_AT_SUPPORT 0
#define LORAWAN_PREAMBLE_LENGTH 8
#define LORAWAN_DebugLevel 2
// OTAA parameters – generate !!
uint8_t devEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
uint8_t appEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
uint8_t appKey[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
/* ABP para*/
uint8_t nwkSKey[] = { 0x15, 0xb1, 0xd0, 0xef, 0xa4, 0x63, 0xdf, 0xbe, 0x3d, 0x11, 0x18, 0x1e, 0x1e, 0xc7, 0xda,0x85
};
uint8_t appSKey[] = { 0xd7, 0x2c, 0x78, 0x75, 0x8c, 0xdc, 0xca, 0xbf, 0x55, 0xee, 0x4a, 0x77, 0x8d, 0x16, 0xef,0x67
};
uint32_t devAddr =  ( uint32_t )0x007e6ae1uint16_t userChannelsMask[6]={
0x00FF,0x0000,0x0000,0x0000,0x0000,0x0000 };
LoRaMacRegion_t loraWanRegion = ACTIVE_REGION;
DeviceClass_t  loraWanClass = LORAWAN_CLASS;
uint32_t appTxDutyCycle = 300000;
bool overTheAirActivation = LORAWAN_NETMODE;   // OTAA
bool loraWanAdr = LORAWAN_ADR;
bool keepNet = LORAWAN_NET_RESERVE;
bool isTxConfirmed = LORAWAN_UPLINKMODE;
uint8_t appPort = 2;
uint8_t confirmedNbTrials = 4;
```

**MCU Source Code**

```
/*
DUPE 12/2024
HELTEC CUBECELL-4 (AB02 OLED), LORA 868 MHz, LORAWAN CRO
SENZOR SDS011 UART BUS RX2:29 TX2:30 (AIR QUALITY), SENZOR WORKING TIME: 30 sec, SLEEPING 300 sec
SENZOR BME280 I2C BUS SDA1:37 SCL1:36 (TEMP/PRESS/HUMI)
CO: https://heltec-automation-docs.readthedocs.io/en/latest/index.html
CO :https://github.com/lewapek/sds-dust-sensors-arduino-library
```

CO: https://github.com/adafruit/Adafruit_BME280_Library
RGB red means sending;
RGB purple means joined done;
RGB blue means RxWindow1;
RGB yellow means RxWindow2;
RGB green means received done;
*/

```
#include "Arduino.h"
#include <softSerial.h>
#include <Wire.h>
#include "LoRaWan_APP.h"
#include "HT_SH1107Wire.h"
#include "SdsDustSensor.h"
#include "Adafruit_BME280.h"
#include "setup-lora.h"
#define SDAI2CPIN SDA1
#define SCLI2CPIN SCL1
#define BoardRxPin UART_RX2
#define BoardTxPin UART_TX2
#define BAUD 9600
#define BYTES 26
#define BME_ADDR 0x77
softSerial softwareSerial(BoardTxPin, BoardRxPin);
SdsDustSensor sds(softwareSerial);
SH1107Wire oled_disply(0x3c, 500000, SDA, SCL ,GEOMETRY_128_64, GPIO10); // addr, freq, sda, scl, resolution, rst
Adafruit_BME280 bme;
int count = 0;
static void prepareTxFrame( uint8_t port)
{
	count++;
	appDataSize = BYTES;
	char payload[BYTES];
	uint16_t vbat = getBatteryVoltage();
	String payloads = "";
	Serial.println("Initialize BMP Sensor ");
	Wire.begin(SDAI2CPIN, SCLI2CPIN);
	if ((bme.begin(BME_ADDR))) {
		delay(500);
		bme.takeForcedMeasurement();
		delay(1000);
		float t = bme.readTemperature();
		float p = bme.readPressure();
		float h = bme.readHumidity();
		Serial.print("Teplota=");
		Serial.println((t));
		Serial.print("Tlak=");
		Serial.println(p);
		Serial.print("Vlhkost=");
		Serial.println(h);
		Serial.print("BatteryVoltage:");
		Serial.println(vbat);
		Wire.end();
		oled_disply.init();
		oled_disply.setFont(ArialMT_Plain_10);
		oled_disply.clear();
		oled_disply.drawString(0, 0, "PM2.5/10: ");
		oled_disply.drawString(0, 12, "TEMP: ");
		oled_disply.drawString(0, 24, "PRESS: ");
		oled_disply.drawString(0, 36, "HUMI: ");
		oled_disply.drawString(0, 48, "BAT/PKT: ");
		oled_disply.drawString(65, 12, String(t));
		oled_disply.drawString(65, 24, String(p));
		oled_disply.drawString(65, 36, String(h));
```

```
                oled_disply.drawString(65, 48, String(vbat));
                oled_disply.display();
        } else {
                Serial.print("Could not find BME sensor: ");
                Serial.println(pm.statusToString());
                oled_disply.clear();
                oled_disply.drawString(0, 0, "NO BME DATA");
                oled_disply.display();
                float t,p,h = 0;
        }
        Serial.println("Initialize Dust Sensor ");
        sds.begin();
        delay(200);
        Serial.println("WakeUP Dust Sensor");
        sds.wakeup();
        Serial.println(sds.queryFirmwareVersion().toString());
        Serial.println(sds.setQueryReportingMode().toString()); // ensures sensor is in 'query' reporting mode
        Serial.println("Working Dust Sensor");
        delay(30000); // working 30 seconds
        PmResult pm = sds.queryPm();
        if (pm.isOk()) {
                float pm25 = pm.pm25;
                float pm10 = pm.pm10;
                int p1 = int(10*(pm25+10));
                int p2 = int(10*(pm10+10));
                int p3 = int(100*(t+50));
                int p4 = int (p+20000);
                int p5 = int(100*(h+10));
                payloads.concat(p1); // 4B
                payloads.concat(p2); // 4B
                payloads.concat(vbat); // 4B
                payloads.concat(p3); // 4B
                payloads.concat(p4); // 6B
                payloads.concat(p5); // 4B
                payloads.toCharArray(payload,BYTES);
                // appData = (unsigned char *)payload;
                strcpy ((char*)appData, payload);
                Serial.println(pm.toString());
                Serial.println(payloads);
                oled_disply.drawString(65, 0, String(pm25));
                oled_disply.drawString(95, 0, String(pm10));
                oled_disply.drawString(95, 48, String(count));
                oled_disply.display();
        } else {
                Serial.print("Could not read from dust sensor: ");
                Serial.println(pm.statusToString());
                oled_disply.clear();
                oled_disply.drawString(0, 30, "NO DUST DATA");
                oled_disply.display();
        }
        WorkingStateResult state = sds.sleep();
        if (state.isWorking()) {
                Serial.println("Problem with sleeping dust sensor.");
        } else {
                Serial.println("Dust sensor is sleeping");
        }
}
void VextON(void)
{
        pinMode(Vext,OUTPUT);
        digitalWrite(Vext, LOW);
}
void VextOFF(void) //Vext default OFF
```

```
{
        pinMode(Vext,OUTPUT);
        digitalWrite(Vext, HIGH);
}
void setup() {
        Serial.begin(115200);
        VextON();
        boardInitMcu();
        deviceState = DEVICE_STATE_INIT;
        LoRaWAN.ifskipjoin();
        delay(500);
}
void loop()
// HELTEC
{
        switch( deviceState )
        {
                case DEVICE_STATE_INIT:
                {
 #if(AT_SUPPORT)
                        Enable_AT()
                        getDevParam();
 #endif
                        printDevParam();
                        LoRaWAN.init(loraWanClass,loraWanRegion);
                        deviceState = DEVICE_STATE_JOIN;
                        break;
                }
                case DEVICE_STATE_JOIN:
                {
                        LoRaWAN.join();
                        break;
                }
                case DEVICE_STATE_SEND:
                {
                        prepareTxFrame( appPort );
                        LoRaWAN.send();
                        deviceState = DEVICE_STATE_CYCLE;
                        break;
                }
                case DEVICE_STATE_CYCLE:
                {
                        // Schedule next packet transmission
                        txDutyCycleTime = appTxDutyCycle + randr( 0, APP_TX_DUTYCYCLE_RND );
                        LoRaWAN.cycle(txDutyCycleTime);
                        deviceState = DEVICE_STATE_SLEEP;
                        break;
                }
                case DEVICE_STATE_SLEEP:
                {
                        LoRaWAN.sleep();
                        break;
                }
                default:
                {
                        deviceState = DEVICE_STATE_INIT;
                        break;
                }
        }
}
```