ÚVOD

Kolega astronom a polularizátor vědy a osvěty stran světelného znečištění, Jan Kondziolka, mne oslovil, zda bych mu pomohl naprogramovat Arduino a rozchodit senzor prachových částic PMS 7003. Po letmém pohledu na google a github, jsem nakonec kývnul, protože projekt je již velmi dobře popsán. Podělím se s vámi tedy o stručný návod jak na to.

V principu jde o měření prachových částic, které se běžně monitoruje i ČHMI. V naší aplikaci měříme pomocí laserového čidla PMS 7003, který svítí do proudu vzduchu laserem a vyhodnocuje odraz. Ze snímače lezou tyto veličiny

- PM 1.0 (ug/m3) – částice > 1 mikro metr
- PM 2.5 (ug/m3) – částice > 2,5 mikro metru
- PM 10 (ug/m3) – částice > 10 mikro metr

Jelikož nejde o přesně kalibrované zařízení a zobrazovány budou pouze aktuální hodnoty, bude se spíše jednat o jakýsi demonstrátor znečištění ovzduší. Nikoli o stanici dlouhodobě monitorující a vyhodnocující kvalitu ovzduší v delším časovém horizontu

HARDWARE

Přistál mi na stole notoricky známý papírový pytlík s bublinkovou výstelkou opatřen čínskými písmenky. Projekt je to velmi jednoduchý, stačí pouze tři komponenty a tyto jsem obdržel jako zadání.

- Arduino UNO
- LCD display 1602 na I2C
- Senzor prachových částic PMS 7003

LCD display se připojuje klasicky na I2C sběrnici (SDA,SCK) + napájení

Senzor, to byl trochu oříšek, v datasheetu píší, že je sice napájen 5V, ale logiku má 3,3V!

Komunikuje přes UART (RX, TX) nicméně pro základní funkci, kdy posílá jen data ze snímače, nám stačí jen TX vývod ze senzoru PMS 7003 a RX vstup v Arduinu. Nejprve jsem zkusil převodník úrovní a 3,3V ze snímače jsem převedl na 5V logiku, ale ani obraz ani zvuk. Nakonec jsem vyřadil převodník úrovní a připojil TX výstup z PMS 7003 přímo do Arduina na RX a voila…běželo na první dobrou. Ale to předbíhám. Hardware máme připojen a můžeme se vrhnout na programování. Jelikož UNO má HW piny pro sériovou komunikaci společné s USB převodníkem, nastavím si sériovou linku softwarově na jiných pinech. Jinak bych musel snímač neustále odpojovat při nahrávání programu z PC. Požiji tedy piny 2 – RX, (3 TX nepřipojen).

PROGRAM

Předem deklaruji, že nejsem žádný učený programátor. Základy jazyka wire ovládám, ale většinu kódů beru z příkladů v programu IDE, nebo postahuji z netu a pak akorát tak nějak učešu a dodělám podle sebe. Stáhl jsem si tedy základní kód na stránkách Adafruit přímo k tomuto snímači.

Jelikož zadání bylo aby zařízení zobrazovalo tři základní veličiny na jednoduchém displeji, bylo téměř hotovo. Nicméně nachystal jsem si kód pro případné pozdější rozšíření o modul reálného času RTC, a SD karty pro logování naměřených hodnot. V kódu mám rovněž zakomentovanou možnost připojení OLED displeje 128×64, který zobrazí tyto tři aktuální veličiny plus aktuální datum a čas. Při cvičném sestavení to vypadá na tomto malém OLED velmi pěkně.

Časem tedy zřejmě přibude ještě zmíněné logování na SD a případné další funkce.

Kód není třeba zcela popisovat. U klíčových věcí mám komentáře přímo v kódu. Jelikož je použit pouze malý display se 16 znaky na řádek, musel jsem osekat informace na minimum. Přemazání spodního řádku s hodnotami je řešeno velmi primitivně, a tedy před každým výpisem, se daná sekce přepíše mezerou. Vzorkovací frekvence zobrazování je řešena delay, ale ještě předělám na millis aby to bylo košér

```
#include <Wire.h>
#include <SoftwareSerial.h>
SoftwareSerial pmsSerial(2, 3);//nastavení softwareserial, pin 2 - RX arduina
- zde poslouchá ..propojit s TX senzoru, pin 3 nezapojen

//-------------------LCD 1602 varianta ------------------------------------
-----------------

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

//---------- modul RTC a SD card modul pro logování dat----------------------
----------------
/*
  #include <SPI.h>
  #include <SD.h>
  #include "RTClib.h"

  RTC_DS1307 DS1307;
*/
//---------------------OLED varianta-------------------------------------
/*
  #include "U8glib.h"

  U8GLIB_SSD1306_128X64 OLED(U8G_I2C_OPT_NONE); // inicializace OLED displeje
z knihovny U8glib  vytvoření objektu OLED
  long int prepis = 0; // proměnná pro uchování času poslední obnovy displeje

  char seznamDni[7][8] = {"nedele", "pondeli", "utery", "streda", "ctvrtek",
"patek", "sobota"}; // vytvoření pole seznamDni s názvy jednotlivých dní
*/
//------------------------------------------------------------------------



void setup() {
  // our debugging output
```

```
  Serial.begin(115200);

  // sensor baud rate is 9600
  pmsSerial.begin(9600);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(1, 0);
  lcd.print("Senzor castic");
  lcd.setCursor(1, 1);
  lcd.print("www.koppik.cz");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0, 0);

  //---------------nastavení RTC-----------------------------------------
---------
  /*
    // pro nastavení času v obvodu reálného času použijte jednorázově(!)
    // následující příkaz v pořadí rok, měsíc, den, hodina, minuta, vteřina
    // příklad: 26.4.2016 9:10:11
    DS1307.adjust(DateTime(2019, 10, 9, 23, 10, 11));
  */

}

struct pms5003data {
  uint16_t framelen;
  uint16_t pm10_standard, pm25_standard, pm100_standard;
  uint16_t pm10_env, pm25_env, pm100_env;
  uint16_t particles_03um, particles_05um, particles_10um, particles_25um,
particles_50um, particles_100um;
  uint16_t unused;
  uint16_t checksum;
};

struct pms5003data data;

void loop() {
```

```
  cteni();
  hodnoty();
 // zobraz();
}

//---------------------------------------------------------------------
//-------------- čtení hodnot ----------------------------------------

void cteni(){

if (readPMSdata(&pmsSerial)) {
    // reading data was successful!
    Serial.println();
    Serial.println("---------------------------------------");
    Serial.println("Concentration Units (standard)");
    Serial.print("PM 1.0: "); Serial.print(data.pm10_standard);
    Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_standard);
    Serial.print("\t\tPM 10: "); Serial.println(data.pm100_standard);
    Serial.println("---------------------------------------");
    Serial.println("Concentration Units (environmental)");
    Serial.print("PM 1.0: "); Serial.print(data.pm10_env);
    Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_env);
    Serial.print("\t\tPM 10: "); Serial.println(data.pm100_env);
    Serial.println("---------------------------------------");
    Serial.print("Particles > 0.3um / 0.1L air:");
Serial.println(data.particles_03um);
    Serial.print("Particles > 0.5um / 0.1L air:");
Serial.println(data.particles_05um);
    Serial.print("Particles > 1.0um / 0.1L air:");
Serial.println(data.particles_10um);
    Serial.print("Particles > 2.5um / 0.1L air:");
Serial.println(data.particles_25um);
    Serial.print("Particles > 5.0um / 0.1L air:");
Serial.println(data.particles_50um);
    Serial.print("Particles > 10.0 um / 0.1L air:");
Serial.println(data.particles_100um);
    Serial.println("---------------------------------------");
  }
```

```
}
//-------------------------------------------------------------------------
---
boolean readPMSdata(Stream *s) {
  if (! s->available()) {
    return false;
  }

  // Read a byte at a time until we get to the special '0x42' start-byte
  if (s->peek() != 0x42) {
    s->read();
    return false;
  }

  // Now read all 32 bytes
  if (s->available() < 32) {
    return false;
  }

  uint8_t buffer[32];
  uint16_t sum = 0;
  s->readBytes(buffer, 32);

  // get checksum ready
  for (uint8_t i = 0; i < 30; i++) {
    sum += buffer[i];
  }

  /* debugging
    for (uint8_t i=2; i<32; i++) {
    Serial.print("0x"); Serial.print(buffer[i], HEX); Serial.print(", ");
    }
    Serial.println();
  */

  // The data comes in endian'd, this solves it so it works on all platforms
  uint16_t buffer_u16[15];
  for (uint8_t i = 0; i < 15; i++) {
    buffer_u16[i] = buffer[2 + i * 2 + 1];
```

```
    buffer_u16[i] += (buffer[2 + i * 2] << 8);
  }

  // put it into a nice struct :)
  memcpy((void *)&data, (void *)buffer_u16, 30);

  if (sum != data.checksum) {
    Serial.println("Checksum failure");
    return false;
  }
  // success!
  return true;
}

//----------------ZOBRAZENÍ HODNOT, 2 VARIANTY DISPLEJŮ----------------------
-


//------------------------LCD 1602-----------------------------------------
-----------------------
void hodnoty() {

//  lcd.setCursor(0, 1);
//  lcd.print("               ");

  lcd.setCursor(0, 0);
  lcd.print("PM1");
  lcd.setCursor(0, 1);
  lcd.print("     ");
  lcd.setCursor(0, 1);
  lcd.print(data.pm10_standard);


  lcd.setCursor(5, 0);
  lcd.print("P2.5");
  lcd.setCursor(5, 1);
  lcd.print("     ");
  lcd.setCursor(5, 1);
  lcd.print(data.pm25_standard);
```

```
  lcd.setCursor(11, 0);
  lcd.print("P10");
  lcd.setCursor(11, 1);
  lcd.print("     ");
  lcd.setCursor(11, 1);
  lcd.print(data.pm100_standard);
  delay(50);

}

//----------------------OLED 128x63 ----------------------------------------
----------------------------
/*
  void zobraz()
  {

  if (millis() - prepis > 100) { // následující skupina příkazů obnoví obsah
OLED displeje
    OLED.firstPage();
    do {
      hodnoty1();


    } while ( OLED.nextPage() );
    prepis = millis(); // uložení posledního času obnovení
  }


  }


  void hodnoty1()
  {

  OLED.setFont(u8g_font_helvB08);
  OLED.setPrintPos(0, 10);
  OLED.print("PM 1.0 (ug/m3):  ");
  OLED.print(data.pm10_standard);
```

```
  OLED.setFont(u8g_font_helvB08);
  OLED.setPrintPos(0, 22);
  OLED.print("PM 2.5 (ug/m3):  ");
  OLED.print(data.pm25_standard);

  OLED.setFont(u8g_font_helvB08);
  OLED.setPrintPos(0, 34);
  OLED.print("PM 10.0 (ug/m3):  ");
  OLED.print(data.pm100_standard);

  DateTime datumCas = DS1307.now();
  OLED.setFont(u8g_font_helvB08);
  OLED.setPrintPos(0, 48);
  OLED.print("Cas:  ");
  OLED.print(datumCas.hour());
  OLED.print(':');
  OLED.print(datumCas.minute());
  OLED.print(':');
  OLED.print(datumCas.second());

  OLED.setPrintPos(0, 60);
  //OLED.print("Datum:  ");
  OLED.print(seznamDni[datumCas.dayOfTheWeek()]);
  OLED.print(" ");
  OLED.print(datumCas.day());
  OLED.print('.');
  OLED.print(datumCas.month());
  OLED.print('.');
  OLED.print(datumCas.year());
  OLED.println();
  }

*/
```

ZÁVĚR

Komponenty mám tedy „zadrátované“ k sobě. Předám to kolegovi, který s tím již naloží dle svého. První úskalí vidím asi v možnosti kalibrace a možnosti vnášení korekcí do měření.

Nejprve uvidíme jak to měří, porovnáme s nějakou meteorologickou stanicí a a pak případně upravíme kód, podle toho jaké budou požadavky na funkce. Zatím to není ošetřeno nijak uživatelsky. Pokud bude nutnost korekcí, bude se muset změnit kód. Do budoucna mě napadlo, že při použití SD modulu, by byl na kartě nějaký konfigurační soubor, který by se dal načíst v PC, přes poznámkový blok změnit hodnoty korekcí a Arduino už by si je při startu natáhlo a započítalo tuto opravu pro další měření. Uvidíme co přinese provoz.