#include <SoftwareSerial.h>

SoftwareSerial pmsSerial(2,3);

void setup() {

// our debugging output

Serial.begin(115200);

// sensor baud rate is 9600

pmsSerial.begin(9600);

}

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() {

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;

}