**Arduino Code for PM2.5&Pm10 Senors**

1. // On Leonardo/Micro or others with hardware serial, use those!
2. // uncomment this line:
3. // #define pmsSerial Serial1
5. // For UNO and others without hardware serial, we must use software serial...
6. // pin #2 is IN from sensor (TX pin on sensor), leave pin #3 disconnected
7. // comment these two lines if using hardware serial
8. #include <SoftwareSerial.h>
9. SoftwareSerial pmsSerial(2, 3);
11. void setup() {
12. // our debugging output
13. Serial.begin(115200);
15. // sensor baud rate is 9600
16. pmsSerial.begin(9600);
17. }
19. struct pms5003data {
20. uint16\_t framelen;
21. uint16\_t pm10\_standard, pm25\_standard, pm100\_standard;
22. uint16\_t pm10\_env, pm25\_env, pm100\_env;
23. uint16\_t particles\_03um, particles\_05um, particles\_10um, particles\_25um, particles\_50um, particles\_100um;
24. uint16\_t unused;
25. uint16\_t checksum;
26. };
28. struct pms5003data data;
30. void loop() {
31. if (readPMSdata(&pmsSerial)) {
32. // reading data was successful!
33. Serial.println();
34. Serial.println("---------------------------------------");
35. Serial.println("Concentration Units (standard)");
36. Serial.print("PM 1.0: "); Serial.print(data.pm10\_standard);
37. Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25\_standard);
38. Serial.print("\t\tPM 10: "); Serial.println(data.pm100\_standard);
39. Serial.println("---------------------------------------");
40. Serial.println("Concentration Units (environmental)");
41. Serial.print("PM 1.0: "); Serial.print(data.pm10\_env);
42. Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25\_env);
43. Serial.print("\t\tPM 10: "); Serial.println(data.pm100\_env);
44. Serial.println("---------------------------------------");
45. Serial.print("Particles > 0.3um / 0.1L air:"); Serial.println(data.particles\_03um);
46. Serial.print("Particles > 0.5um / 0.1L air:"); Serial.println(data.particles\_05um);
47. Serial.print("Particles > 1.0um / 0.1L air:"); Serial.println(data.particles\_10um);
48. Serial.print("Particles > 2.5um / 0.1L air:"); Serial.println(data.particles\_25um);
49. Serial.print("Particles > 5.0um / 0.1L air:"); Serial.println(data.particles\_50um);
50. Serial.print("Particles > 10.0 um / 0.1L air:"); Serial.println(data.particles\_100um);
51. Serial.println("---------------------------------------");
52. }
53. }
55. boolean readPMSdata(Stream \*s) {
56. if (! s->available()) {
57. return false;
58. }
60. // Read a byte at a time until we get to the special '0x42' start-byte
61. if (s->peek() != 0x42) {
62. s->read();
63. return false;
64. }
66. // Now read all 32 bytes
67. if (s->available() < 32) {
68. return false;
69. }
71. uint8\_t buffer[32];
72. uint16\_t sum = 0;
73. s->readBytes(buffer, 32);
75. // get checksum ready
76. for (uint8\_t i=0; i<30; i++) {
77. sum += buffer[i];
78. }
80. /\* debugging
81. for (uint8\_t i=2; i<32; i++) {
82. Serial.print("0x"); Serial.print(buffer[i], HEX); Serial.print(", ");
83. }
84. Serial.println();
85. \*/
87. // The data comes in endian'd, this solves it so it works on all platforms
88. uint16\_t buffer\_u16[15];
89. for (uint8\_t i=0; i<15; i++) {
90. buffer\_u16[i] = buffer[2 + i\*2 + 1];
91. buffer\_u16[i] += (buffer[2 + i\*2] << 8);
92. }
94. // put it into a nice struct :)
95. memcpy((void \*)&data, (void \*)buffer\_u16, 30);
97. if (sum != data.checksum) {
98. Serial.println("Checksum failure");
99. return false;
100. }
101. // success!
102. return true;
103. }