

```

1  /*
2  * This is the .cpp file for the PMS7003 sensor
3  * This code was written exclusively by MECH 45X Team 26
4  */
5  #include "PM.h"
6
7  PM_7003::PM_7003() {
8      current_byte = 0;
9      packetdata.frame_length = MAX_FRAME_LENGTH;
10     frame_length = MAX_FRAME_LENGTH;
11 }
12
13 PM_7003::~~PM_7003() {
14 }
15
16 int PM_7003::getpm(void) {
17     return pm_avgpm2_5;
18 }
19
20 bool PM_7003::run_PM_sensor(void) {
21     /*
22     * run the PM sensor
23     * Start serial connection
24     *
25     * drain_serial() and read_sensor() until enough values have been read
26     * to take the average
27     */
28     Serial1.begin(9600);
29     read_count = 1;
30     done_reading = false;
31     frame_sync_count = 0;
32     pm_avgpm2_5 = 0;
33     while(!done_reading && frame_sync_count < MAX_FRAME_SYNC_COUNT) {
34         drain_serial();
35         delay(500);
36         read_sensor();
37     }
38
39     Serial1.end();
40
41     if(done_reading) {
42         Serial.println("-----");
43         Serial.println("Done reading from PM sensor");
44         Serial.println("-----");
45         Serial.println(" ");
46         return true;
47     }
48     else if(!done_reading && frame_sync_count >= MAX_FRAME_SYNC_COUNT){return false;}
49 }
50
51 void PM_7003::drain_serial(void) {
52     /*
53     * Drains serial buffer if there are more than 32 entries
54     * Reads entries to drain serial buffer
55     */
56     if (Serial1.available() > 32) {
57         drain = Serial1.available();
58         Serial.println("-- Draining buffer: ");
59         Serial.println(Serial1.available(), DEC);
60         for (int drain_index = drain; drain_index > 0; drain_index--) {Serial1.read();}
61     }
62 }
63
64 void PM_7003::frame_sync(void) {
65     /*
66     * syncs frames for PM sensor
67     * checks that frames are being read in correct order
68     * exits when it confirms that frames are being read correctly
69     */

```

```

70     sync_state = false;
71     frame_count = 0;
72     byte_sum = 0;
73
74     while (!sync_state && frame_sync_count < MAX_FRAME_SYNC_COUNT){
75         current_byte = Serial1.read();
76
77         if(current_byte == FIRST_BYTE && frame_count == 0) {
78             frame_buffer[frame_count] = current_byte;
79             packetdata.start_frame[0] = current_byte;
80             byte_sum = current_byte;
81             frame_count = 1;
82         }
83         else if(current_byte == SECOND_BYTE && frame_count == 1){
84             frame_buffer[frame_count] = current_byte;
85             packetdata.start_frame[1] = current_byte;
86             byte_sum = byte_sum + current_byte;
87             frame_count = 2;
88             sync_state = true;
89         }
90         else{
91             frame_sync_count++;
92             Serial.println("frame is syncing");
93             Serial.print("Current character: ");
94             Serial.println(current_byte, HEX);
95             Serial.print("frame count: ");
96             Serial.println(frame_sync_count);
97             delay(500);
98
99             if(frame_sync_count >= MAX_FRAME_SYNC_COUNT) {
100                 Serial.println("-----");
101                 Serial.println("Max frame count exceeded");
102                 Serial.println("-----");
103             }
104         }
105     }
106 }
107
108
109 void PM_7003::read_sensor(void) {
110     /*
111     * Sync the frames
112     * read bytes and fill frame_buffer
113     * use data_switch to calculate different parameters
114     * print_messages once all values have been read.
115     * done_reading = true if enough values have been read
116     */
117     frame_sync();
118
119     while(sync_state == true && Serial1.available() > 0) {
120         current_byte = Serial1.read();
121         frame_buffer[frame_count] = current_byte;
122         byte_sum = byte_sum + current_byte;
123         frame_count++;
124         uint16_t current_data = frame_buffer[frame_count-1]+(frame_buffer[frame_count-2]
125         ]<<8);
126         data_switch(current_data);
127
128         if (frame_count >= frame_length && read_count <= MAX_READ_COUNT) {
129             print_messages();
130             pm_avgpm2_5 = pm_avgpm2_5 + pm2_5;
131             read_count++;
132             break;
133         }
134     }
135
136     if (read_count > MAX_READ_COUNT) {
137         pm_avgpm2_5 = exp((pm_avgpm2_5/MAX_READ_COUNT + 109314)/15990)*10000;
138         done_reading = true;

```

```

138     }
139 }
140
141
142 void PM_7003::data_switch(uint16_t current_data) {
143     /*
144      * data_switch uses current data and frame_count
145      * to assign values to parameters
146      */
147     switch (frame_count) {
148     case 4:
149         packetdata.frame_length = current_data;
150         frame_length = current_data + frame_count;
151         break;
152     case 6:
153         packetdata.concPM1_0_factory = current_data;
154         break;
155     case 8:
156         packetdata.concPM2_5_factory = current_data;
157         break;
158     case 10:
159         packetdata.concPM10_0_factory = current_data;
160         break;
161     case 12:
162         packetdata.concPM1_0_ambient = current_data;
163         break;
164     case 14:
165         packetdata.concPM2_5_ambient = current_data;
166         break;
167     case 16:
168         packetdata.concPM10_0_ambient = current_data;
169         break;
170     case 18:
171         packetdata.countPM0_3um = current_data;
172         break;
173     case 20:
174         packetdata.countPM0_5um = current_data;
175         break;
176     case 22:
177         packetdata.countPM1_0um = current_data;
178         break;
179     case 24:
180         packetdata.countPM2_5um = current_data;
181         break;
182     case 26:
183         packetdata.countPM5_0um = current_data;
184         break;
185     case 28:
186         packetdata.countPM10_0um = current_data;
187         break;
188     case 29:
189         current_data = frame_buffer[frame_count-1];
190         packetdata.version = current_data;
191         break;
192     case 30:
193         current_data = frame_buffer[frame_count-1];
194         packetdata.error = current_data;
195         break;
196     case 32:
197         packetdata.checksum = current_data;
198         byte_sum -= ((current_data>>8)+(current_data&0xFF));
199         break;
200     default:
201         break;
202     }
203 }
204
205 void PM_7003::print_messages(void) {
206     /*

```

```

207     * Print messages to string and Serial screen
208     */
209     Serial.println("-----");
210     Serial.print("PMS 7003 - Reading #");
211     Serial.println(read_count);
212     Serial.println("-----");
213     sprintf(print_buffer, ", %02x, %02x, %04x, ",
214             packetdata.start_frame[0], packetdata.start_frame[1], packetdata.frame_length);
215     sprintf(print_buffer, "%s%04d, %04d, %04d, ", print_buffer,
216             packetdata.concPM1_0_factory, packetdata.concPM2_5_factory, packetdata.
                concPM10_0_factory);
217     sprintf(print_buffer, "%s%04d, %04d, %04d, ", print_buffer,
218             packetdata.concPM1_0_ambient, packetdata.concPM2_5_ambient, packetdata.
                concPM10_0_ambient);
219     sprintf(print_buffer, "%s%04d, %04d, %04d, %04d, %04d, %04d, ", print_buffer,
220             packetdata.countPM0_3um, packetdata.countPM0_5um, packetdata.countPM1_0um,
221             packetdata.countPM2_5um, packetdata.countPM5_0um, packetdata.countPM10_0um);
222     sprintf(print_buffer, "%s%02d, %02d, ", print_buffer,
223             packetdata.version, packetdata.error);
224
225     pm2_5 = packetdata.countPM1_0um - packetdata.countPM2_5um + packetdata.countPM0_5um
226             - packetdata.countPM1_0um + packetdata.countPM0_3um - packetdata.countPM0_5um;
227     Serial.println(print_buffer);
228     Serial.println("-----");
229     delay(500);
230 }
231

```