

```

1  /*
2  *  Script_all.ino
3  *  This script runs the sensor package
4  *  Uses objects for each of the sensors
5  *  Prints information to Serial screen
6  *  Publishes data to ThingSpeak
7  */
8
9  #include "CALCULATE_MRT.h"
10 #include "MHZ19.h"
11 #include "CCS821.h"
12 #include "SHT35D.h"
13 #include "MRT.h"
14 #include "PM.h"
15 #include "Time.h"
16 #include <Wire.h>
17
18 // create instances of objects
19 PM_7003 myPM;
20 ClosedCube_Si7051 myMRT;
21 ClosedCube_SHT31D mySHT;
22 Adafruit_CCS811 myVOC;
23 MHZ19 myCO2;
24 mrt_and_ot my_MRT_OT;
25
26 /*
27 *  Boolean expressions
28 *  start_xxx indicate whether sensor has been read from properly
29 *  read_from_xxx indicate whether or not to read from sensor_xxx (changes throughout
    code)
30 *  finished_xxx indicates whether done reading from a sensor (read a good average)
31 */
32 bool start_co2 = false;
33 bool start_voc = false;
34 bool start_sht = false;
35 bool start_pm = false;
36 bool start_mrt = false;
37
38 bool read_from_co2 = true;
39 bool read_from_pm = false;
40
41 bool finished_co2 = false;
42 bool finished_pm = false;
43 bool finished_other_sensors = false;
44 bool finished_mrt_ot = false;
45 bool finished_voc = false;
46
47 // average reading values
48 int co2_ave = -1;
49 float sht_rh_ave = -1;
50 float sht_t_ave = -1;
51 float voc_eCO2_ave = -1;
52 float voc_TVOC_ave = -1;
53 int pm_ave = -1;
54 float T_g = -1;
55 float T_a = -1;
56 float T_mrt = -1;
57 float T_ot = -1;
58
59 bool publish_data = true; // should we publish data?
60
61 // pin numbers for pm and co2 sensors
62 int pm_transistor_control = A4;
63 int pm_tx_transistor_control = A5;
64 int co2_transistor_control = A3;
65
66 void setup() {
67     /*
68     *  Start Serial and Wire connections

```

```

69     * Initialize transistor control for CO2 and PM
70     * Turn CO2 sensor on (make_sensor_read())
71     * Test all I2C sensors (MRT, SHT, VOC)
72     * Stop and wait for 30 seconds (warm-up)
73     */
74     Serial.begin(9600);
75     Wire.begin();
76     Serial.println("Initializing");
77
78     myCO2.set_transistor(co2_transistor_control);
79     myPM.set_transistor(pm_transistor_control, pm_tx_transistor_control);
80
81     myCO2.make_sensor_read();
82
83     start_mrt = myMRT.start_mrt();
84     Serial.println("-----");
85
86     start_sht = mySHT.start_sht();
87     Serial.println("-----");
88
89     start_voc = myVOC.start_voc();
90     Serial.println("-----");
91     Serial.println("30 second delay");
92     Serial.println("-----");
93     delay(30000);
94
95 }
96
97 void loop() {
98     /*
99     * Wait for CO2 sensor to warm-up (PM sensor is off)
100    * Read from MRT, SHT, and VOC sensors while CO2 sensor warms-up
101    * After CO2 sensor warms-up, read from CO2 sensor and save average reading
102    * Save average value from MRT, SHT, and VOC sensors
103    * Turn off CO2 sensor, turn on PM sensor
104    * Read from MRT, SHT, and VOC sensors while PM sensor warms-up
105    * After PM sensor warms-up, read from PM sensor and push all data to ThingSpeak
106    * Repeat
107    */
108
109    // Decide which of CO2 or PM sensor to read from
110    if(read_from_co2) {
111        start_co2 = myCO2.make_sensor_read();
112        start_pm = false;
113
114        if(start_co2) {
115            read_from_co2 = false;
116            read_from_pm = true;
117            finished_co2 = true;
118        }
119    }
120    else if(read_from_pm) {
121        start_pm = myPM.make_sensor_read();
122        start_co2 = false;
123
124        if(start_pm) {
125            read_from_pm = false;
126            read_from_co2 = true;
127            finished_pm = true;
128        }
129    }
130
131    start_mrt = myMRT.run_mrt(); //read from MRT sensor
132
133    // Read from SHT sensor, or restart SHT sensor
134    if(start_sht) {
135        Serial.println("Reading from SHT Sensor");
136        Serial.println("-----");
137        start_sht = mySHT.run_sht();

```

```

138     Serial.println("-----");
139 }
140 else if(!start_sht) {
141     Serial.println("-----");
142     Serial.println("Not reading from SHT Sensor");
143     Serial.println("-----");
144     Serial.println("Tryng to start SHT");
145     start_sht = mySHT.start_sht();
146     Serial.println("-----");
147 }
148 // Read from VOC sensor, or restart VOC sensor
149 if(start_voc) {
150     Serial.println("Reading from VOC Sensor");
151     Serial.println("-----");
152     start_voc = myVOC.run_voc();
153     Serial.println("-----");
154 }
155 else if(!start_voc) {
156     start_voc = myVOC.start_voc();
157     Serial.println("Reading from VOC Sensor");
158     Serial.println("-----");
159     start_voc = myVOC.run_voc();
160     Serial.println("-----");
161 }
162
163 // If done reading from CO2 sensor, save CO2, MRT, SHT, and VOC readings
164 if(finished_co2 && !finished_other_sensors) {
165     finished_other_sensors = true;
166
167     if(!finished_mrt_ot) {
168         if(start_mrt && start_sht){
169             T_g = myMRT.get_MRT_ave();
170             T_a = mySHT.get_t_ave();
171             sht_rh_ave = mySHT.get_rh_ave();
172             my_MRT_OT.calculate_mrt_and_ot(T_g, T_a);
173             T_mrt = my_MRT_OT.get_mrt();
174             T_ot = my_MRT_OT.get_ot();
175             finished_mrt_ot = true;
176         }
177         else if(start_mrt && !start_sht) {
178             T_g = myMRT.get_MRT_ave();
179             T_a = -1;
180             sht_rh_ave = -1;
181             T_mrt = -1;
182             T_ot = -1;
183         }
184         else if(!start_mrt && start_sht) {
185             T_g = -1;
186             T_a = mySHT.get_t_ave();
187             sht_rh_ave = mySHT.get_rh_ave();
188             T_mrt = -1;
189             T_ot = -1;
190         }
191         else {
192             T_g = -1;
193             T_a = -1;
194             sht_rh_ave = -1;
195             T_mrt = -1;
196             T_ot = -1;
197         }
198     }
199
200     if(start_voc && !finished_voc){
201         voc_eCO2_ave = myVOC.get_eCO2_ave();
202         voc_TVOC_ave = myVOC.get_TVOC_ave();
203         finished_voc = true;
204     } else {
205         voc_eCO2_ave = -1;
206         voc_TVOC_ave = -1;

```

```

207     }
208
209     co2_ave = myCO2.get_co2_ave();
210
211     if(finished_mrt_ot && finished_voc) {
212         finished_other_sensors = true;
213     }
214 }
215
216 // If done reading from PM and CO2 sensors, save PM reading and push to ThingSpeak
217 if(finished_co2 && finished_pm) {
218     pm_ave = myPM.get_pm_ave();
219     finished_co2 = false;
220     finished_pm = false;
221     finished_mrt_ot = false;
222     finished_voc = false;
223     finished_other_sensors = false;
224
225     if(publish_data) {
226         char data[1000];
227         sprintf(data, "{ \"Mean Radiant Temperature\": \"%3.2f\", \"Operating
228             Temperature\": \"%3.2f\", \"CO2 Concentration\": \"%i\", \"eCO2\":
229             \"%4.2f\", \"TVOC\": \"%4.2f\", \"PM 2_5\": \"%i\", \"Air Temperature\":
230             \"%3.2f\", \"Relative Humidity of Air\": \"%3.2f\"}" , T_mrt, T_ot, co2_ave,
231             voc_eCO2_ave, voc_TVOC_ave, pm_ave, T_a, sht_rh_ave);
232         Serial.println("-----");
233         Serial.print("Data:");
234         Serial.println(data);
235         Serial.println("-----");
236
237         Particle.publish("IEQ Final Prototype", data, PRIVATE);
238
239         myCO2.reset_co2_ave();
240         myPM.reset_pm_ave();
241     }
242 }
243
244 }
245
246 }

```