```
/*
 * Script _all.ino
 * This script runs the sensor package
 * Uses objects for each of the sensors
 * Prints information to Serial screen
 * Publishes data to ThingSpeak
 */

#include "CALCULATE_MRT.h"
#include "MHZ19.h"
#include "CCS821.h"
#include "SHT35D.h"
#include "MRT.h"
#include "PM.h"
#include <Wire.h>

// create instances of objects
PM_7003 myPM;
ClosedCube_Si7051 myMRT;
ClosedCube_SHT31D mySHT;
Adafruit_CCS811 myVOC;
MHZ19 myCO2;
mrt_and_ot my_MRT_OT;

/*
 * Boolean expression
 * Indicate whether sensor has
 * been started successfully
 */
bool start_co2 = false;
bool start_voc = false;
bool start_sht = false;
bool start_pm = false;
bool start_mrt = false;

// average reading values
int co2_ave = -1;
float sht_rh_ave = -1;
float sht_t_ave = -1;
float voc_eCO2_ave = -1;
float voc_TVOC_ave = -1;
float pm_ave = -1;
float T_g = -1;
float T_a = -1;
float T_mrt = -1;
float T_ot = -1;

bool publish_data = true; // should we publish data?

// pin numbers for pm and co2 sensors
int pm_transistor_control = A4;
int co2_transistor_control = A3;

void setup() {
    /*
     * start Serial and wire connections
     * try to start each sensor
     * assign true false to each of the relevant booleans
     */
    Serial.begin(9600);
    Wire.begin();
    pinMode(pm_transistor_control,OUTPUT);
    pinMode(co2_transistor_control,OUTPUT);
    Serial.println("Initializing");

    Serial.println("Trying to start CO2 sensor");
    delay(1000);
    digitalWrite(co2_transistor_control, HIGH);
    start_co2 = myCO2.start_sensor();
```

```
70        Serial.println("----------------------");
71        digitalWrite(co2_transistor_control, LOW);
72
73        start_mrt = myMRT.start_mrt();
74        Serial.println("----------------------");
75        start_sht = mySHT.start_sht();
76        Serial.println("----------------------");
77
78        start_voc = myVOC.start_voc();
79        Serial.println("----------------------");
80
81        Serial.println("Trying to start PM sensor");
82        digitalWrite(pm_transistor_control, HIGH);
83        delay(1000);
84        start_pm = myPM.run_PM_sensor();
85        digitalWrite(pm_transistor_control, LOW);
86
87        if(start_pm){Serial.println("Successfully started PM sensor");}
88        else if(!start_pm){Serial.println("Failed to start PM sensor");}
89        Serial.println("----------------------");
90    }
91
92    void loop() {
93        /*
94         * Run each sensor if it has been started
95         * If the sensor has not been started, print error message
96         * After all values have been read, prepare to publish data
97         */
98        if(start_co2) {
99            digitalWrite(co2_transistor_control, HIGH);
100           delay(1800);
101           start_co2 = myCO2.run_sensor();
102           digitalWrite(co2_transistor_control, LOW);
103           delay(1000);
104       }
105
106       if(start_pm) {
107           Serial.println("Reading from PMS Sensor");
108           Serial.println("----------------------");
109           digitalWrite(pm_transistor_control, HIGH);
110           delay(10000);
111           start_pm = myPM.run_PM_sensor();
112           digitalWrite(pm_transistor_control, LOW);
113           delay(500);
114       }
115       else if(!start_pm) {
116           Serial.println("Not reading from PMS Sensor");
117           Serial.println("--------------------------");
118           delay(500);
119       }
120
121       if(start_mrt) {
122           myMRT.run_mrt();
123           delay(500);
124       }
125       else if(!start_mrt) {
126           Serial.println("Not reading from MRT Sensor");
127           Serial.println("--------------------------");
128           delay(500);
129       }
130
131       if(start_sht) {
132           Serial.println("Reading from SHT Sensor");
133           Serial.println("----------------------");
134           mySHT.run_sht();
135           Serial.println("----------------------");
136       }
137       else if(!start_sht) {
138           Serial.println("Not reading from SHT Sensor");
```

```
        Serial.println("--------------------------");
        delay(500);
    }

    if(start_voc) {
        Serial.println("Reading from VOC Sensor");
        Serial.println("----------------------");
        myVOC.run_voc();
        Serial.println("----------------------");
    }
    else if(!start_voc) {
        Serial.println("Not reading from VOC Sensor");
        Serial.println("--------------------------");
        delay(500);
    }

    if(publish_data) {
        char data[1000];
        if(start_mrt && start_sht){
            T_g = myMRT.get_MRT_ave();
            T_a = mySHT.get_t_ave();
            sht_rh_ave = mySHT.get_rh_ave();
            my_MRT_OT.calculate_mrt_and_ot(T_g, T_a);
            T_mrt = my_MRT_OT.get_mrt();
            T_ot = my_MRT_OT.get_ot();
        }
        else if(start_mrt && !start_sht) {
            T_g = myMRT.get_MRT_ave();
            T_a = -1;
            sht_rh_ave = -1;
            T_mrt = -1;
            T_ot = -1;
        }
        else if(!start_mrt && start_sht) {
            T_g = -1;
            T_a = mySHT.get_t_ave();
            sht_rh_ave = mySHT.get_rh_ave();
            T_mrt = -1;
            T_ot = -1;
        }
        else {
            T_g = -1;
            T_a = -1;
            sht_rh_ave = -1;
            T_mrt = -1;
            T_ot = -1;
        }

        if(start_pm){pm_ave = myPM.getpm();}
        else {pm_ave = -1;}

        if(start_co2){co2_ave = myCO2.get_co2_ave();}
        else{co2_ave = -1;}

        if(start_voc){
            voc_eCO2_ave = myVOC.get_eCO2_ave();
            voc_TVOC_ave = myVOC.get_TVOC_ave();
        } else {
            voc_eCO2_ave = -1;
            voc_TVOC_ave = -1;
        }


        sprintf(data,"{ \"Mean Radiant Temperature\": \"%f\", \"Operating
        Temperature\": \"%f\", \"Globe Temperature\": \"%f\", \"CO2 Concentration\":
        \"%i\", \"TVOC\": \"%f\",\"PM 2.5 (Counts/m^3)\": \"%f\", \"Air Temperature\":
        \"%f\",\"Relative Humidity of Air\": \"%f\"}" , T_mrt, T_ot, T_g, co2_ave,
        voc_TVOC_ave, pm_ave, T_a, sht_rh_ave);
        Serial.println(data);
```

```
204             Particle.publish("IEQ Data", data, PRIVATE);
205
206         }
207     }
208
209
```