```cpp
/*
 * This is the .cpp file for the MH-Z19 CO2 Sensor
 * This code was exclusively written by MECH 45X Team 26
 */

#include "MHZ19.h"

MHZ19::MHZ19() {
}

MHZ19::~MHZ19() {
}

bool MHZ19::start_sensor(void) {
    /*
     * Start sequence for MHZ19
     * returns true if sensor on, false if sensor off
     * uses run_sensor() function
     */
    return (run_sensor());
}

bool MHZ19::run_sensor(void) {
    /*
     * Run the MHZ19 sensor
     * Set ppm to zero
     * clear the frame_buffer
     * drain the serial buffer
     * read from the sensor
     * print reading
     * add the reading to the average value buffer
     * calculate average value
     */
    co2_ppm = -1;
    co2_ppm_average = 0;
    is_average_taken = false;
    does_sensor_work = true;
    reading_count = 1;

    serial_drain();
    start_countdown(STARTUP_TIME);

    while(is_average_taken == false && does_sensor_work == true) {
        memset(frame_buffer, 0, 9);
        read_sensor();
        print_current_reading();
        add_to_ave_buf();
        calculate_average_reading();
        print_average_reading();
        co2_ppm = -1;
    }
    if(is_average_taken == true) {
        return(true);
    } else {return(false);}
}

void MHZ19::start_countdown(int start_time) {
    /*
     * Countdown so that users can visualize how long before the sensor starts
     */
    while(start_time > 0) {
        Serial.print("Starting CO2 Sensor in: ");
        Serial.print(start_time);
        Serial.println("s");
        delay(1000);
        start_time = start_time - 1;
    }
}

```

```
70  void MHZ19::print_current_reading(void) {
71      /*
72       * Prints current reading if reading is valid (i.e. co2_ppm > 0)
73       * and if the maximum number of readings haven't been exceeded
74       */
75      if(co2_ppm > 0 && reading_count > DISCARD_VALUES) {
76        Serial.print("MHZ19 CO2 PPM Reading ");
77        Serial.print(reading_count);
78        Serial.print(": ");
79        Serial.println(co2_ppm);
80      }
81      else if(co2_ppm > 0 && reading_count <= DISCARD_VALUES) {
82          Serial.print("DISCARD - MHZ19 CO2 PPM Reading ");
83          Serial.print(reading_count);
84          Serial.print(": ");
85          Serial.println(co2_ppm);
86      }
87      else {
88        Serial.println("Error reading CO2 PPM from MHZ19");
89      }
90  }
91
92  void MHZ19::add_to_ave_buf(void) {
93      /*
94       * IF a valid value of co2 is read and the number of reading is less than the max,
95       * THEN add current value to buffer
96       */
97      if(co2_ppm > 0 && reading_count <= NUMBER_OF_VALUES) {
98          mhz19_buffer[reading_count - 1] = co2_ppm;
99          reading_count += 1;
100     }
101 }
102
103 void MHZ19::calculate_average_reading(void) {
104     /*
105      * IF the number of readings exceeds the number of values to be read,
106      * THEN calculate the average
107      */
108     if(reading_count > NUMBER_OF_VALUES) {
109         for(int k = DISCARD_VALUES; k < NUMBER_OF_VALUES; k++) {co2_ppm_average +=
        mhz19_buffer[k];}
110
111         co2_ppm_average = co2_ppm_average / ( NUMBER_OF_VALUES - DISCARD_VALUES );
112
113         is_average_taken = true;
114     }
115 }
116
117 void MHZ19::print_average_reading(void) {
118     /*
119      * IF the average has been taken (co2_ppm_average > 0)
120      * THEN print the average
121      */
122     if(co2_ppm_average > 0) {
123         Serial.print("CO2 PPM Average Reading: ");
124         Serial.println(co2_ppm_average);
125     }
126 }
127
128 void MHZ19::read_sensor(void) {
129     /*
130      *  Start Serial1 connection
131      *  Send command to read from sensor to the sensor
132      *  Read from the sensor (fill_from_buffer();)
133      *  Calculate PPM for CO2
134      *  End Serial connection
135     */
136
137     Serial1.begin(9600);
```

```
138       Serial1.write(mhz19_read_command, 9);
139       delay(1000);
140       fill_frame_buffer();
141       co2_ppm = 256*frame_buffer[2] + frame_buffer[3];
142       Serial1.end();
143   }
144
145   void MHZ19::serial_drain(void) {
146     /*
147      * Drains serial buffer when sensor is turned on
148      */
149       while (Serial1.available() > 0) {
150           drain = Serial1.available();
151           Serial.print("-- Draining buffer: ");
152       }
153   }
154
155   void MHZ19::frame_sync(void) {
156       /*
157        * Sync frames so that frames are added to the frame_buffer in the correct order
158        * IF correct byte is read, THEN add to buffer and move on to next byte
159        * ELSE read byte and discard
160        * IF no bytes are available to read and the frames have not been synced, THEN send
          command to read from sensor again
161        *
162        * frame_sync_count keeps track of how many frames are added to frame_buffer
163        * frame_read_count keeps track of how many frames are read but not added to buffer
          (fails if too many frames read)
164        */
165       sync_state = false;
166       frame_sync_count = 0;
167       frame_read_count = 0;
168       byte_sum = 0;
169
170       while (!sync_state && Serial1.available() > 0 && frame_read_count <
          MAX_FRAME_READ_COUNT) {
171           current_byte = Serial1.read();
172
173           if (current_byte == MHZ19_ZEROTH_BYTE && frame_sync_count == 0) {
174               frame_buffer[frame_sync_count] = current_byte;
175               byte_sum = current_byte;
176               frame_sync_count = 1;
177           }
178           else if (current_byte == MHZ19_FIRST_BYTE && frame_sync_count == 1) {
179               frame_buffer[frame_sync_count] = current_byte;
180               byte_sum += current_byte;
181               sync_state = true;
182               frame_sync_count = 2;
183           }
184           else {
185               Serial.print("-- Frame syncing... ");
186               Serial.println(current_byte, HEX);
187               frame_read_count ++;
188           }
189
190           if (!sync_state && !(Serial1.available() > 0) && frame_read_count <
              MAX_FRAME_READ_COUNT) {
191               Serial1.write(mhz19_read_command, 9);
192               Serial.println("Read command has been sent to CO2 sensor");
193               delay(500);
194           }
195       }
196   }
197
198   void MHZ19::fill_frame_buffer(void) {
199     /*
200      * Sync frames
201      * Read byte into frame_buffer
202      */
```

```
203         frame_sync();
204 
205         while(sync_state && Serial1.available() > 0 && frame_sync_count < MAX_FRAME_LEN) {
206             current_byte = Serial1.read();
207             frame_buffer[frame_sync_count] = current_byte;
208             byte_sum += current_byte;
209             frame_sync_count++;
210         }
211     }
212 
213     // getter functions
214     int MHZ19::get_co2_ave(void) {return co2_ppm_average;}
215     int MHZ19::get_co2_reading(void) {return co2_ppm;}
216 
```