```
/*
 * This is the .h file for the PMS7003 sensor
 * This code was written exclusively by MECH 45X Team 26
 */
#include <stdint.h>
#include "WProgram.h"
#include "Time.h"

#define LIB_PM_H
#define FIRST_BYTE 0x42
#define SECOND_BYTE 0x4D
#define SENSOR_OUTPUT_PIN A0
#define MAX_FRAME_LENGTH 64

#define START_TIME 6000
#define SAMPLING_TIME 280
#define SLEEP_TIME 912
#define MAX_READ_COUNT 5
#define MAX_FRAME_SYNC_COUNT 40
#define PMS_START_UP_TIME 120
#define MAX_FUNCTION_CALL_COUNT 3

class PM_7003 {
public:
    PM_7003();
    virtual ~PM_7003();
    float get_pm_ave(void);
    void set_transistor(int ground_pin, int tx_pin);
    bool make_sensor_read(void);
    void calibrate_sensor(void);
    void reset_pm_ave(void);

private:
    int current_byte;
    bool sync_state;
    char print_buffer[256];
    uint16_t byte_sum;
    int drain;
    uint16_t current_data;
    int pm_ground_control;
    int pm_tx_control;
    char frame_buffer[MAX_FRAME_LENGTH];
    int frame_count;
    int frame_length;

    bool debug = false;

    float pm_avgpm2_5;
    float pm_avgpm1_75;
    float pm_avgpm0_75;
    float pm_avgpm0_4;
    float pm1_75_buf[MAX_READ_COUNT];
    float pm0_75_buf[MAX_READ_COUNT];
    float pm0_4_buf[MAX_READ_COUNT];

    bool done_reading;
    int read_count;
    int function_call_count;
    int frame_sync_count;
    bool first_time;

    bool run_PM_sensor(void);
    void drain_serial(void);
    void frame_sync(void);
    void read_sensor(void);
    void data_switch(uint16_t current_data);
    void print_messages(void);

    //time
    void begin_timer(void);
    bool check_begin_reading(void);
    time_t start_time;
    time_t current_time;
    time_t duration;


    struct PMS7003data {
        uint8_t  start_frame[2];
        uint16_t frame_length;
        uint16_t concPM1_0_factory;
        uint16_t concPM2_5_factory;
        uint16_t concPM10_0_factory;
        uint16_t concPM1_0_ambient;
        uint16_t concPM2_5_ambient;
        uint16_t concPM10_0_ambient;
        uint16_t countPM0_3um;
        uint16_t countPM0_5um;
        uint16_t countPM1_0um;
        uint16_t countPM2_5um;
        uint16_t countPM5_0um;
        uint16_t countPM10_0um;
        uint8_t  version;
        uint8_t  error;
        uint16_t checksum;
    } packetdata;
};
```