BÁO CÁO KỸ THUẬT LẬP TRÌNH

Sinh viên: Nguyễn Việt Thành
Mã số sinh viên: 20202524
Mã lớp: 136089
Giảng viên HD: TS. Hoàng Đức Chính

***Tóm tắt* – Báo cáo bài tập lớn môn Kỹ thuật lập trình về chủ đề "Thiết kế chương trình phần mềm với các hàm và cấu trúc dữ liệu thích hợp để xử lý dữ liệu cảm biến.". Chương trình sử dụng ngôn ngữ C trên hệ điều hành Windows để mô phỏng dữ liệu cảm biến bụi PM2.5 đo nồng độ hạt bụi có trong không khí.**

I. Ý TƯỞNG THIẾT KẾ

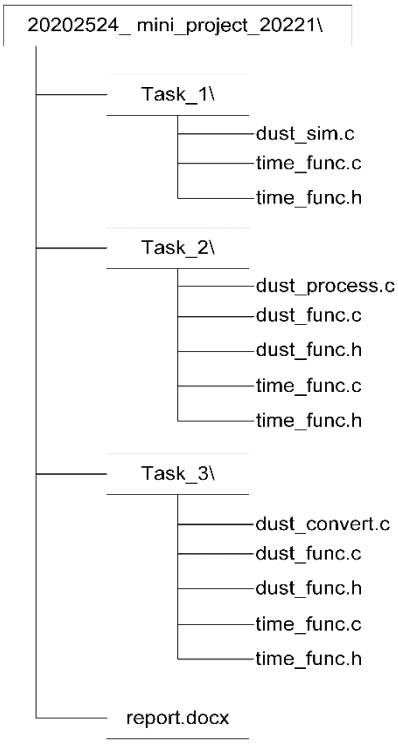
Yêu cầu của bài tập lớn cần thiết kế 3 chương trình chính:

- Chương trình mô phỏng dữ liệu cảm biến bụi
- Chương trình xử lý dữ liệu cảm biến bụi
- Chương trình tạo lập bản tin truyền thông

Do nội dung của các chương trình có mối liên hệ với nhau, cách tiếp cận từ trên xuống được lựa chọn để xây dựng chương trình. Hình 1 mô tả sơ đồ phương pháp tiếp cận từ trên xuống được sử dụng. Trong đó chương trình mô phỏng dữ liệu cảm biến bụi (Task_1) gồm có 3 hàm chính để làm việc việc là time_*start_func*, *sample_quantity*, và *next_sample_time*. Chương trình xử lý dữ liệu cảm biến bụi (Task_2) gồm 2 phần chính: phần xử lý dữ liệu ngoại lệ và phần xử lý dữ liệu hợp lệ, đối với xử lý dữ liệu hợp lệ gồm các phân hoạch thuật toán khác nhau *dust_summary*, *dust_aqi*, và *dust_statistics*. Chương trình tạo lập bản tin truyền thông (Task_3) gồm 4 hàm chính: *time_diff*, *dec2hex*, *float2hex754*, và *checksum*.

Cấu trúc thư mục của toàn bộ chương trình được thể hiện ở Hình 2. Thư mục gồm 3 chương trình chính của từng nhiệm vụ: *dust_sim.c*, *dust_process.c*, và *dust_convert.c*; các hàm quan trọng và các hàm phụ trợ được thiết lập thành 2 thư viện: *time_func.h* và *dust_func.h*. Ngoài ra, sử dụng thêm các thư viện chuẩn của C:

- Task_1: *stdio.h, stdlib.h, string.h, time.h*
- Task_2: *stdio.h, stdlib.h, string.h, math.h*
- Task_3: *stdio.h, stdlib.h, string.h*

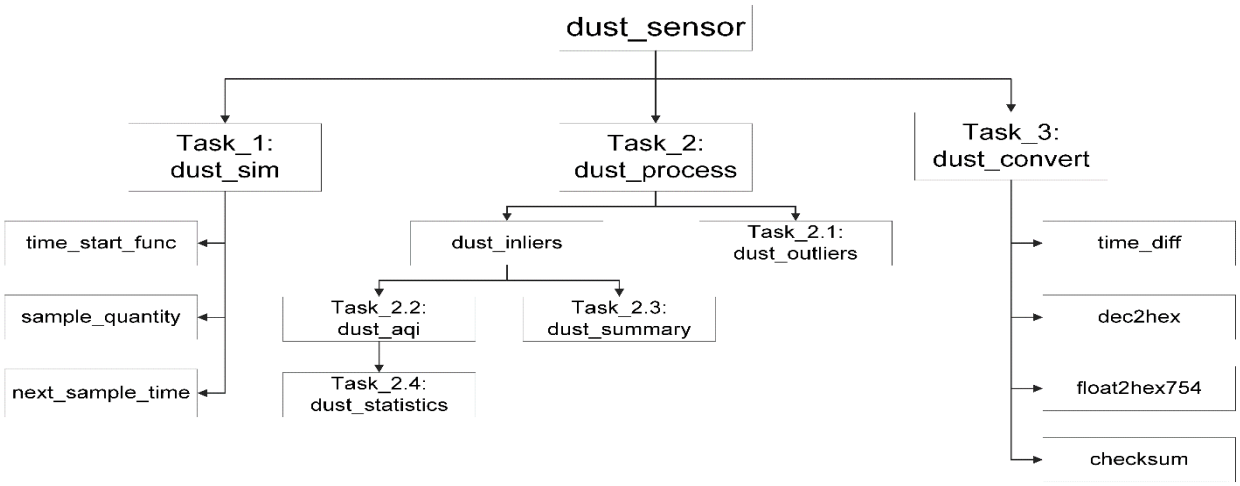


Hình I.2: Cấu trúc thư mục chương trình

II. THIẾT KẾ CHI TIẾT

A. Chương trình mô phỏng dữ liệu cảm biến bụi (Task_1)

Ở Task_1, yêu cầu thiết kế chương trình cho phép người dùng nhập đầu vào là số lượng cảm biến, thời gian trích mẫu và khoảng thời gian đo, đầu ra là một file csv mô phỏng dữ liệu đo của các cảm biến trong trong khoảng thời gian đo định trước, các lần lấy mẫu cách nhau một khoảng thời gian trích mẫu. Với yêu cầu trên, sau khi viết một số hàm/đoạn mã để kiểm tra các lỗi đầu vào, em xây dựng 3 hàm chức năng chính: *time_start_func*, *sample_quantity* và *next_sample_time*.



Hình I.1: Sơ đồ tiếp cận từ trên xuống

1. Time_start_func

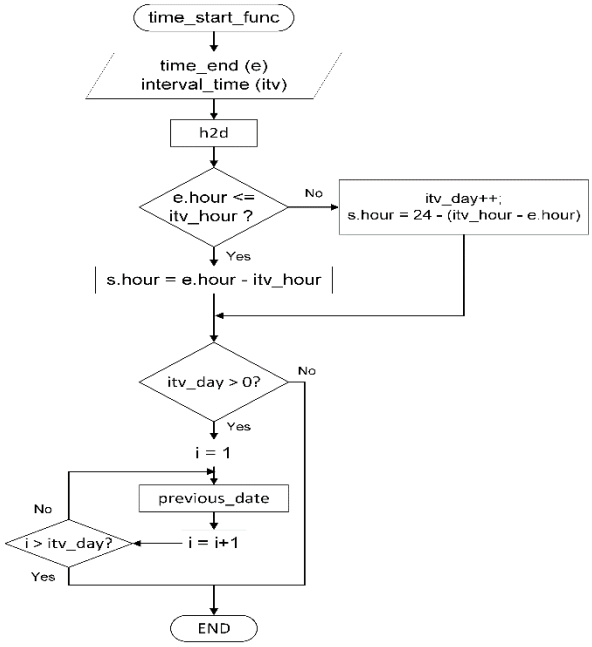
Cú pháp: void time_start_func (struct Time *time_end, int interval_time, struct Time *time_start);

Đầu vào: con trỏ của 1 structure chứa thông tin về thời gian kết thúc việc lấy mẫu và thời gian đo.

Đầu ra: thông tin về thời gian khi bắt đầu lấy mẫu.

Trong việc xây dựng của hàm time_start_func, em sử dụng thêm 2 hàm phụ trợ:

- void h2d (int interval_time, int* day_result, int* hour_result): quy đổi (giờ) ra (ngày) + (giờ).
- void previous_date (struct Time *now_ptr, struct Time *pre_ptr): tìm ngày ngay trước của một ngày.



Hình II.1: Lưu đồ time_start_func

2. Sample_quantity

Cú pháp: int sample_quantity (int itv, int spl)

Đầu vào: thời gian trích mẫu, thời gian đo.

Đầu ra: số lần lấy mẫu cần thực hiện.

3. Next_sample_time

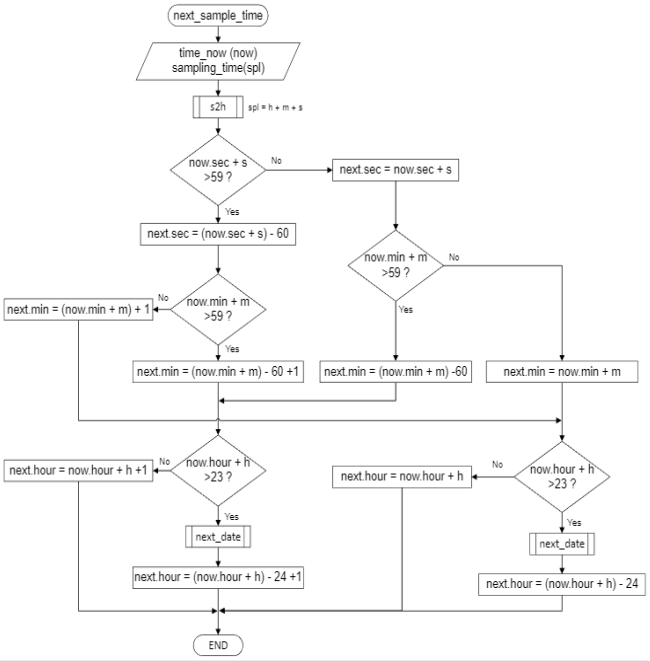
Cú pháp: void next_sample_time (struct Time *now, int sampling_time, struct Time *next)

Đầu vào: con trỏ của 1 structure chứa thông tin thời gian lấy mẫu đang thực hiện và thời gian trích mẫu.

Đầu ra: thông tin thời gian của lần lấy mẫu kế tiếp.

Trong việc xây dựng của hàm time_start_func, em sử dụng thêm 2 hàm phụ trợ:

- void s2h (int spl_time, int* hour_ptr, int* min_ptr, int* sec_ptr): quy đổi (giây) ra (giờ) + (phút) + (giây).
- void next_date (struct Time *now_ptr, struct Time *next_ptr): tìm ngày tiếp theo.



Hình II.2: Lưu đồ next_sample_time

Trong quá trình người dùng sử dụng, không tránh khỏi các thao tác sai hoặc các lỗi hệ thống. Các lỗi ghi nhận được sẽ được lưu vào một log file (task1.log) với các lỗi có thể xảy ra được định nghĩa dưới đây:

- Error 01: Invalid command
- Error 02: Invalid argument
- Error 03: Denied access dust_sensor.csv

B. Chương trình xử lí dữ liệu cảm biến bụi (Task_2)

Với một file csv có định dạng như như ở Task_1, sau khi qua một số đoạn mã để kiểm tra, viết chương trình xử lí và xuất dữ liệu ra các file csv khác nhau.

1. Task_2.1: dust_outliers.csv

Với file csv đầu ra, em thực hiện việc lọc dữ liệu thành các phần riêng biệt: dữ liệu hợp lệ, dữ liệu dị biệt và các dòng dữ liệu bị sai cấu trúc. Dữ liệu dị biệt được tổng hợp và viết thành file dust_outliers.csv, dữ liệu hợp lệ được tổng hợp vào 1 file tạm dust_inliers.csv để phục vụ các nhiệm vụ sau, và các dòng dữ liệu bị lỗi sẽ được thông báo ra file báo lỗi.

Ngoài ra dòng đầu tiên của dust_outliers.csv thông báo số lượng dòng dữ liệu dị biệt.

Trước khi bắt đầu tính toán, em viết một đoạn mã lệnh làm việc với dust_inliers.csv xác định số id lớn nhất từ đó suy ra số lượng id, và quét hai dòng đầu-cuối để lấy thông tin thời gian bắt đầu-kết thúc quá trình đo. Lần đọc dust_inliers.csv tiếp theo sẽ xử lý dữ liệu cho cả Task_2.2, 2.3, và 2.4.

2. Task_2.2: dust_aqi.csv

Trong khi xử lý dữ liệu cho Task_2.2, em có sử dụng một số hàm nhỏ cho thuận tiện:

- int element_check (struct Time* now, struct Time* first, struct Time* last): kiểm tra xem mốc thời gian (now) có giữa (first) và (last) không.
- int aqi_num (float value): nội suy tuyến tính giá trị đo ra giá trị aqi.

Task_2.2 chỉ xử lý đối với từng dòng dữ liệu để kiểm tra xem có nằm trong khoảng thời gian đang xét, truyền giá trị vào các mảng để xử lý, khi chuyển sang khoảng thời gian tiếp theo sẽ xóa hết dữ liệu để lấy không gian làm việc.

3. Task_2.3: dust_summary.csv

Cấp phát động số lượng phù hợp các biến để tính tổng, đếm số lần lấy mẫu, tìm giá trị lớn nhất-nhỏ nhất. Sử dụng lại hàm next_sample_time với thời gian trích mẫu 1 giây để tính thời gian đo.

4. Task_2.4: dust_statistics.csv

Đối với mỗi cảm biến cần 7 biến để tính số giờ của mỗi cấp ô nhiễm nên cấp phát động (7*số lượng cảm biến) biến để làm việc.

Xử lý đối với các nhiệm vụ ở Task_2 chủ yếu tính toán số học, ít phải xây dựng các hàm khác. Các lỗi ghi nhận được sẽ được lưu vào một log file (task1.log) với các lỗi có thể xảy ra được định nghĩa dưới đây:

- Error 01: Invalid command.
- Error 02: File not found or cannot be accessed.
- Error 03: Invalid csv file.
- Error 04: Data is mising at line XY.

C. Chương trình tạo lập bản tin truyền thông (Task_3)

Làm việc với một file csv định dạng như ở Task_2.2, cần tạo lập được bản tin truyền thông. Các nhiệm vụ cần thực hiện:

- ID: chuyển từ số thập phân sang hệ 16 (1 byte)
- Time: chuyển từ số thập phân sang hệ 16 (4 bytes)
- Giá trị đo: chuyển từ số thực sang hexadecimal chuẩn IEEE 754 (4 bytes)
- AQI: chuyển từ số thập phân sang hệ 16 (2 byte)
- Checksum: tính toán với các byte hệ 16 (1 byte)

Với các nhiệm vụ cần thực hiện, xây dựng các hàm chính trong Task_3:

1. dec2hex

Cú pháp: void dec2hex (int decimalNumber, int bytes_num, char *hex_bytes_display);

Đầu vào: số nguyên và độ dài byte cần biểu diễn

Đầu ra: chuỗi biểu diễn bằng số hexa với độ dài yêu cầu

2. float2hex754

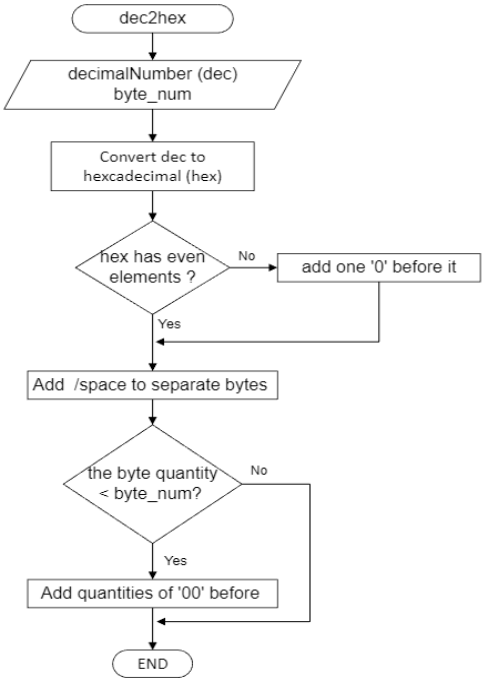
Cú pháp: void float2hex754(double realNumber, int bytes_num, char* hex_bytes_display)

Đầu vào: số thực dương và độ dài byte cần biểu diễn.

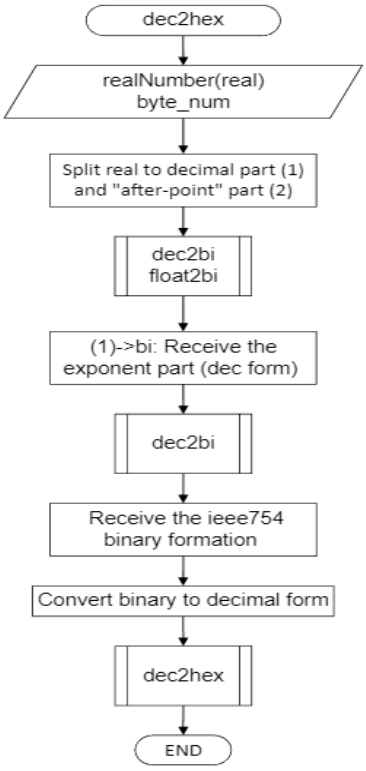
Đầu ra: chuỗi byte hexa biểu diễn số thực trên theo chuẩn IEEE754

Để xây dựng được hàm float2hex754, em sử dụng thêm các hàm phụ:

- void dec2bi (int decimalNumber, char* binary): chuyển thập phân sang nhị phân.
- void float2bi (double realNumber, char* binary): chuyển số thực dương nhỏ hơn 1 sang nhị phân.



Hình II.3: Lưu đồ dec2hex



Hình II.4: Lưu đồ float2hex754

3. time_diff:

Cú pháp: int time_diff (struct Time *time_f_ptr, struct Time *time_l_ptr)

Đầu vào: thông tin 2 mốc thời gian

Đầu ra: khoảng cách giữa 2 mốc thời gian đó (giây)

Sử dụng hàm next_sample_time để xây dựng hàm.

4. checksum:

Cú pháp: void checksum (char hex_12[100], char* result)

Đầu vào: chuỗi ký tự chứa 12 byte hexa cách nhau bởi dấu cách

Đầu ra: kết quả checksum (1 byte)

Sử dụng các hàm:

- hex2dec
- dec2bi
- dec2hex
- bi2dec

Các lỗi ghi nhận được sẽ được lưu vào một log file (task3.log) với các lỗi có thể xảy ra được định nghĩa sau:

- Error 01: Invalid command.
- Error 02: File not found or cannot be accessed.
- Error 03: Invalid csv file.
- Error 04: Cannot overwrite hex_filename.dat
- Error 05: Data is mising at line XY.

Sau khi hoàn thành việc thiết kế các hàm, các Task, lưu đồ thuật toán tổng quát cho toàn bộ chương trình được thể hiện ở Hình II.5.

III. KẾT LUẬN

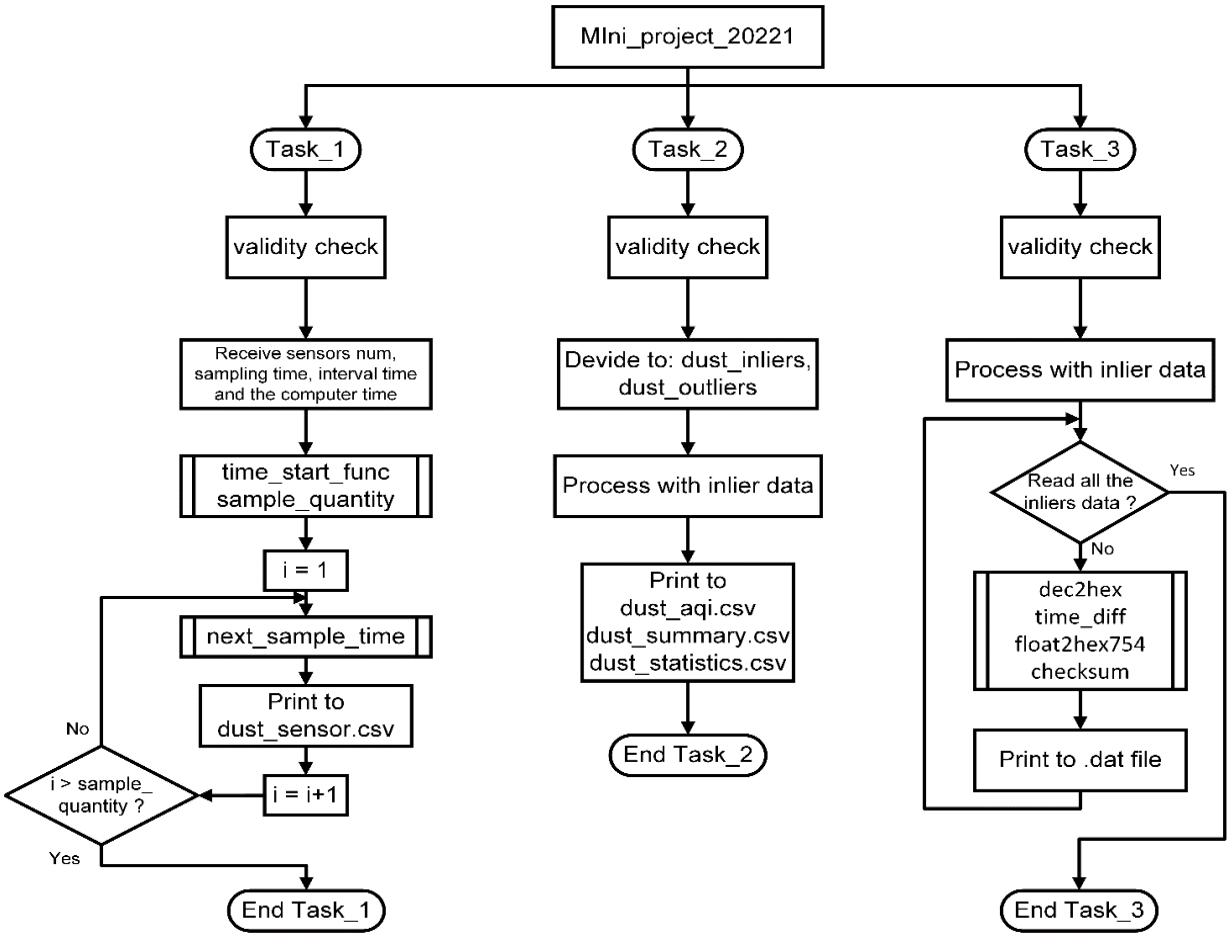
Tổng quan về bài tập lớn, em đã thực hiện được cơ bản những yêu cầu đặt ra:

- Task_1: Tạo được file mô phỏng với chất lượng yêu cầu, nhận diện được một số lỗi thao tác cũng như lỗi khi chạy chương trình.
- Task_2: Tạo được các file yêu cầu, xử lí được lượng lớn dữ liệu đầu vào cho kết quả chính xác, nhận diện được một số lỗi.
- Task_3: Tạo được file truyền tin yêu cầu, nhận diện được một số lỗi.

Tuy vậy, kết quả thu được vẫn còn nhiều hạn chế:

- Tốc độ thực hiện ở task 3 khá chậm.
- Chưa thể bao quát hết các lỗi khi thực hiện chương trình.
- Chương trình chưa được ngắn gọn, tối ưu.

Em xin cảm ơn thầy Hoàng Đức Chính đã ra bài tập rất chất lượng, giúp em được vận dụng các kiến thức đã học và tìm hiểu nhiều kiến thức mới.



Hình II.5: Lưu đồ tổng quát chương trình