[[1]](#footnote-1)

Report for Speed Sensor System project(2022)

Thien V. Dang, *Fellow*, and Hoa Tran, Dr., *Member*

***Abstract*—This document provides an overview and details of this project.**

# I. Introduction

T

HIS project design and write two programs in C++ with proper functions and data structure to simulate a speed

sensor mounted on a DC motor with the following specification:

* Measurement range: 0 ÷ 3000 rpm (revolution per minutes);
* Resolution: 0.2 rpm.

The design for this project is a simple and easy to grasp which allows the users to satisfy their inquisitiveness.

Our programing will run via command line interface, user may input parameters to customize the result.

Project structure:

* Task 1: Includes speed\_sim.cpp and speed\_sim.exe (compiled from speed\_sim.cpp file) files.
* Task 2: Includes some test files, speed\_process.cpp and speed\_process.exe (compiled from speed\_process.cpp file) files.

# II. Design

Each task, we will show the design of the whole program and each function. The flowcharts of the main program and important function and brief description of this flowchart.

## Task 1

### Design

* void output\_error(int error\_code): print error to the screen and write to file at the same time with *error\_code*; this function is not return value.
* bool assign\_value(bool &has\_set, int &variable, string value, int error\_code): set an int *variable* with *value* as string; if this *variable* *has\_set* or *value* is not a number or *value* exceed integer range then *ouput\_error* and returns false, otherwise returns true.
* bool process\_command\_line(int argc, char \*argv[]): this function iterate through argv array, that is parameter from user command line pass by main, then check each pair of element to *assign\_value*. During the process, if invalid parameters are encountered, a log file corresponding to that error will be written (using the *output\_error* function) and return false. This function returns a *bool*.
* string conver\_to\_string(time\_t time): converts a time to HH:mm:ss format – the result is a string.
* int main(int argc, char \*argv[]): main function of the program (entry point), used to call the function *process\_command\_line* which argc and argv is parameter from user command line. If *process\_command\_line* is true, this function writes generated values to csv file.

### Flowchart

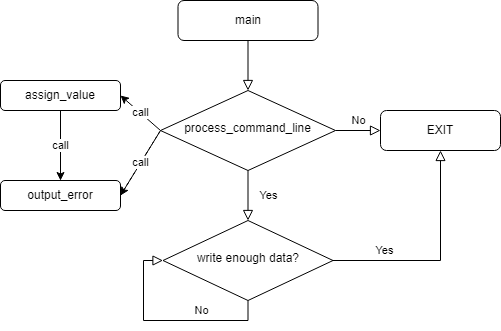


Figure .. Relationship of functions

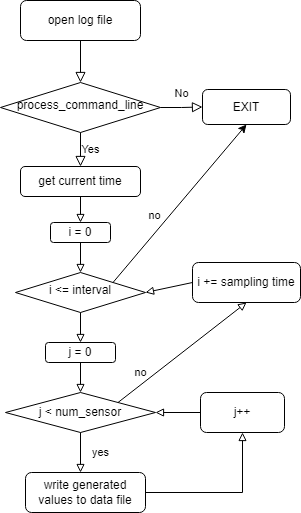


Figure .. Flowchart of the main program

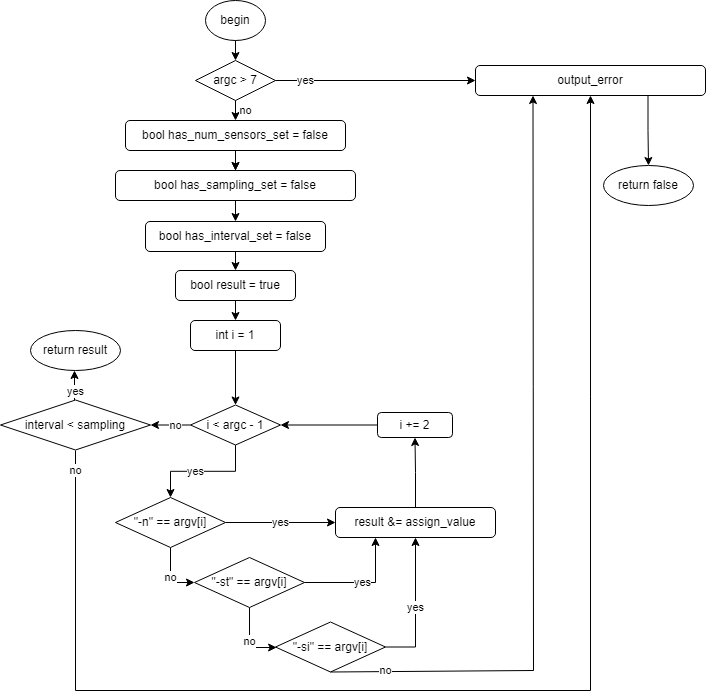


Figure .. Flowchart of the function process\_command\_line

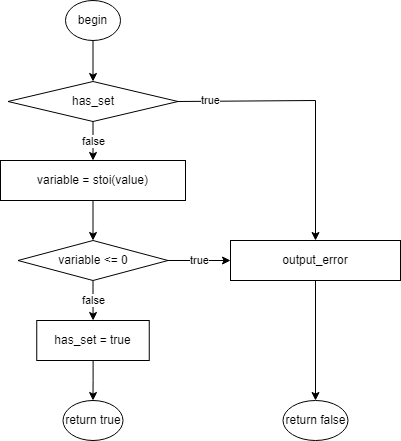


Figure .. Flowchart of the function assign\_value

## Task 2

### Design

* struct record: include id of sensor, hour, minute, second this record was recorded and speed at this time.
* void output\_error(int error\_code): print error to the screen and write to file at the same time with *error\_code*; this function is not return value.
* bool process\_command\_line(int argc, char \*argv[]): this function check if user typed file name or not and user need to sort data by pass “-s” parameter. This function returns a *bool*: if third parameter is “-s” (if exists) and arvc < 4 then return true; otherwise return false.
* int parse\_record(string line, record &result): reads line, check its format, splits it to elements, cast them to *result* parts and check if they are valid in range. Return *error\_code* if any error occurs; otherwise return 0.
* bool load\_data\_and\_filter\_out\_outlier(): read data file line by line, parse them to records. If parse process returns error, then *output* it to screen. Check whether record is outlier, if it is, save it to file; if not save to *data\_bins*. *Data\_bins* is a vector of vectors that separate record by its id, same id is the same vector, all vector are in *data\_bins*.
* int time\_duration(record start, record end): calculate the duration in second of two record.
* void set\_simulation\_interval(record &result): this function receive a record and set *simulation\_interval* to it as hour, minute and second. Simulation interval is calculated by get *time\_duration* of latest record and earliest record.
* vector<record> do\_summary(vector<record> bin): bin is a vector of record have same id. This function calculate max, min and mean of this bin, then returns as a vector orderly.
* void summary\_data(): iterate through *data\_bins*, let each bin *do\_summary* then save it to file.
* vector<int> do\_statistics(vector<record> bin): this function iterate through a bin and count the times gradient of speeds exceed 100 per second. Returns as a vector, first element count increment and second one count decrement.
* void statistics\_data(): iterate through *data\_bins*, let each bin *do\_statistics* then save it to file.
* bool compare\_speed(record r1, record r2): determine if speed of r1 is less than speed of r2.
* void sort\_data(): iterate through *data\_bins*, that is sorted by *id*, then sorts elements of each bin by *sort* function of C++, calculate sorting time, add them and save it to file at the end. Also save *data\_bins*, that sorted by *id*, and by *speed*.
* int main(int argc, char \*argv[]): main function of the program (entry point), used to call the function *process\_command\_line* which argc and argv is parameter from user command line. If *process\_command\_line* is true, this function *load\_data\_and\_filter\_out\_outlier*, if don’t have any error (except missing data), program will *summary\_data* and *statistics\_data*. If user have typed “-s” option, program *sort\_data* before exit.

### Flowchart

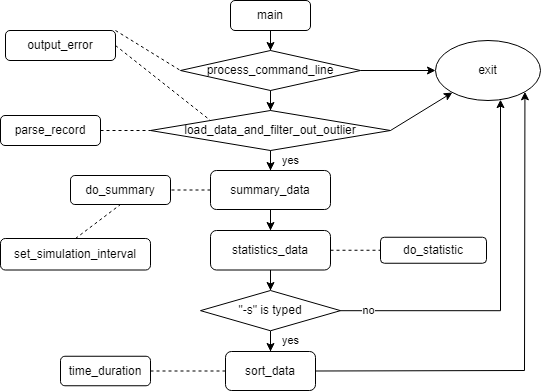


Figure .. Relationship of functions

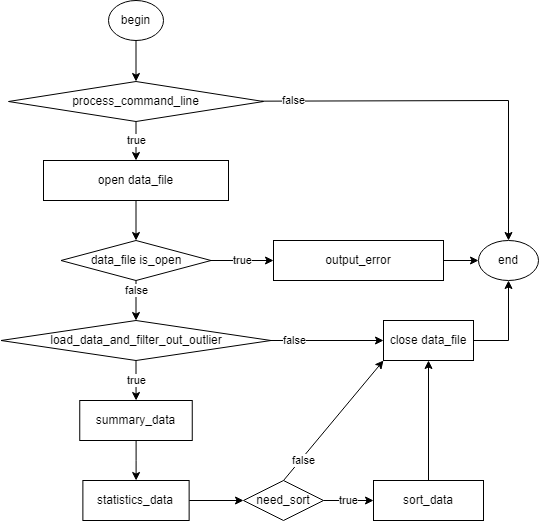


Figure .. Flowchart of the main program

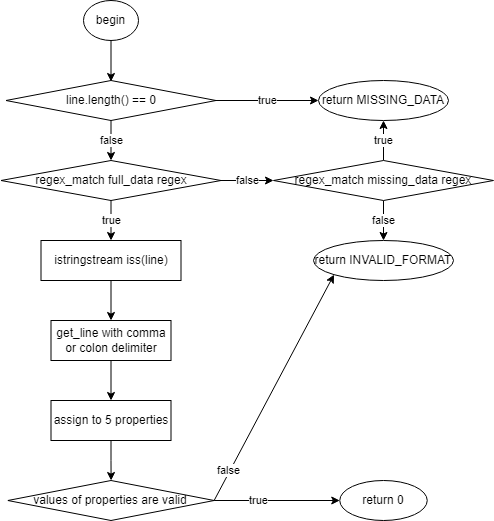


Figure .. Flowchart of the function parse\_record

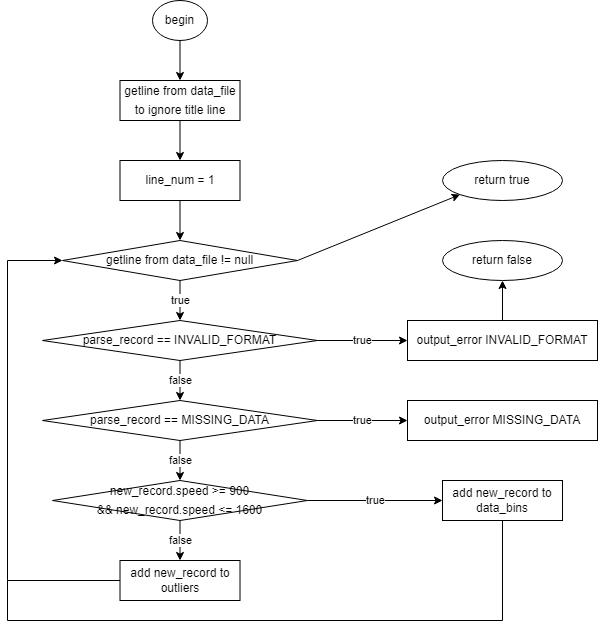


Figure .. Flowchart of the function load\_data\_and\_filter\_out\_outlier

# III. Results

The program runs correctly according to the requirements of the given topic, has checked the happy cases such as the user miss some parameters and bad cases such as the user enters the parameter in the wrong format or number out of range (-n -st,

-n -6).

Almost error about file accessing is handled.

Optional task 2.4 and Additional requirements 3 also implemented.

Although the project was successful, there are areas that could be improved on. Unfortunately, there was no time to complete the extra parts of the project.

# IV. Conclusion

Overall, this is a small project conducted individually so flaws are inevitable in designs, through some evaluations and examinations it is concluded that it still need to be improved and needed to be more comprehensive. With regard to this, there would need enhancement in the coding method and add some features to the project in the future.

Some of a few ideas that we have come up with can be listed below:

* Adding more commands.
* Looking for more bugs to fix so that the program can run smoothly as users will.
* Defining more errors if there are any.

# V. Task allocations

## Member 1

### Task 1

### Task 2

## Member 2

### Task 1

### Task 2

# VI. References

*Basic format for electronic documents (when available online):*

Issuing Organization. (year, month day). *Title*. [Type of medium]. Available: site/path/file

*Example:*

1. U.S. House. 102nd Congress, 1st Session. (1991, Jan. 11). *H. Con. Res. 1, Sense of the Congress on Approval of Military Action*. [Online]. Available: LEXIS Library: GENFED File: BILLS
2. **~~First A. Author~~** ~~(Fellow, IEEE) and all authors may include biographies if the publication allows. Biographies are often not included in conference-related papers. Please check the Information for Authors to confirm. Author photos should be current, professional images of the head and shoulders. The first paragraph may contain a place and/or date of birth (list place, then date). Next, the author’s educational background is listed. The degrees should be listed with the type of degree in what field, which institution, city, state, and country, and year the degree was earned. The author’s major field of study should be lowercase. ~~

~~The second paragraph uses the preferred third person pronoun (he, she, they, etc.) and not the author’s last name. It lists military and work experience, including summer and fellowship jobs. Job titles are capitalized. The current job must have a location; previous positions may be listed without one. Information concerning previous publications may be included. The format for listing publishers of a book within the biography is:~~ *~~Title of Book~~* ~~(publisher name, year) similar to a reference. Current and previous research interests end the paragraph.~~

~~The third paragraph begins with the author’s preferred title and last name (e.g., Dr. Smith, Prof. Jones, Mr. Kajor, Ms. Hunter, Mx. Riley). List any memberships in professional societies other than the IEEE. Finally, list any awards and work for IEEE committees and publications.~~

**~~Second B. Author~~**~~, photograph and biography not available at the time of publication.~~

**~~Third C. Author, Jr.~~** ~~(Member, IEEE), photograph and biography not available at the time of publication.~~

1. 07/11/2022

   Thien V. Dang was with the University of Transport and Communication, Ha Noi, Viet Nam (e-mail: author@ boulder.nist.gov).

   Hoa Tran is with the University of Transport and Communication, Ha Noi, Viet Nam (e-mail: author@ boulder.nist.gov). [↑](#footnote-ref-1)