[[1]](#footnote-1)

Mini project Report

Prepared by: Vo Khoi Thanh Son, Do Quang Thanh, Dinh Quang Anh

Tutor: Hoang Duc Chinh

Course: EE3490E Fundamentals of Embedded Programming

Date: Semester 2, Academic Year 2021/2022

# I. INTRODUCTION

T

HIS report serves as an assessment of knowledge introduced throughout the course, as well as programming skills in C/C++ language. Students are required to design and write program(s) in C or C++ with proper functions and data structure to simulate a speed sensor mounted on a DC motor with specific technical specification.

The main idea of the project is to generate csv files with a view to recording and keeping track of the data on the sensors. In our program, a struct is used to store all the required data on sensors including id, timestamp and random speed. Then we manage to access these members of the struct to process the given tasks.

# II. DESIGN

TASK 1:

+ Function name: createFile

+ Input variables: 3

* Meaning: number of sensors, sampling time and simulation interval
* Name: num\_s, sim\_st, sim\_i
* Type: sufficient integers (int)
* Passing arguments: via variables

+ Output: 1

* Meaning: the output file stream which will write data to speed\_sim\_Timestamp.csv
* Name: myFile
* Type: ofstream

+ Pre-conditions: nothing special!

+ Post-conditions:

* All information about id, timestamp and speed should be written to file

TASK 2:

* **Task 2.1**

+ Function name: outlier\_data

+ Input variables: 1

* Meaning: name of the file to read
* Name: fileName
* Type: string
* Passing argument: via variable

+ Output: 1

* Meaning: the output file stream which will write data to outlier\_data.csv
* Name: outlier\_data
* Type: ofstream

+ Pre-conditions: nothing special!

+ Post-conditions:

* Member ran\_speed of struct variable outlier must be < 900 or > 1600
* **Task 2.2**

+ Function name: data\_summary

+ Input variables: 1

* Meaning: name of the file to read
* Name: fileName
* Type: string
* Passing argument: via variable

+ Output: 1

* Meaning: the output file stream which will write data to data\_summary.csv
* Name: data\_summary
* Type: ofstream

+ Pre-conditions: nothing special!

+ Post-conditions: nothing special!

* **Task 2.3**

+ Function name: data\_statistics

+ Input variables: 1

* Meaning: name of the file to read
* Name: fileName
* Type: string
* Passing argument: via variable

+ Output: 1

* Meaning: the output file stream which will write data to data\_statistics.csv
* Name: data\_statistics
* Type: ofstream

+ Pre-conditions: nothing special!

+ Post-conditions: nothing special!

* **Task 2.4**

+ Function name: sorted\_algorithm

+ Input variables: 1

* Meaning: name of the file to read
* Name: fileName
* Type: string
* Passing argument: via variable

+ Output: 0

+ Pre-conditions: nothing special!

+ Post-conditions:

* The elements in array member sorted\_speed of struct variable sorted\_id has been sorted in ascending order

# III. RESULTS

We successfully create the files in task1 and task 2 as well as the log files containing the errors, including:

● “speed\_data\_TIMESTAMP.csv”, in which TIMESTAMP is replaced by the starting time of the simulation in Unix timestamp in second; containing id, time and speed of a specific number of sensors entered from the user.

● “outlier\_data.csv”, which counts the number of outliers (sensors with speed < 900 or > 1600) in addition to showing the id, time and speed of those outliers.

● “data\_summary.csv”, which stores the largest, smallest, and average speed for each sensor id (apply to the valid data points only)

● “data\_statistics.csv”, counting the number of times that speed increment and decrement between two consecutive valid data points exceed 100 rpm per second for each speed sensor.

● “sorted\_data.csv”, which sort the valid data ascendingly by id and values of speed, then display on the screen the time taken to run the sorting algorithm.

● “task1.log”, including all task 1 errors if any happens.

● “task2.log”, including all task 2 errors if any happens.

Additional part:

In task 2 one additional error is added: “error 2.4: invalid command line argument”, which will show up when users enter more than 3 inputs or invalid command for sorting data (different from “-s”)

# IV. CONCLUSION

Almost all of the tasks given in this project have been completed, including the required and optional part as well as possible errors, with the features strictly following the template.

What we are still unable to accomplish is that the file is incapable of processing large data inputs from the user (for example the simulation interval is too large), thus failing to generate csv files in task 2, while task 1 still functions well.

# V. TASK ALLOCATIONS

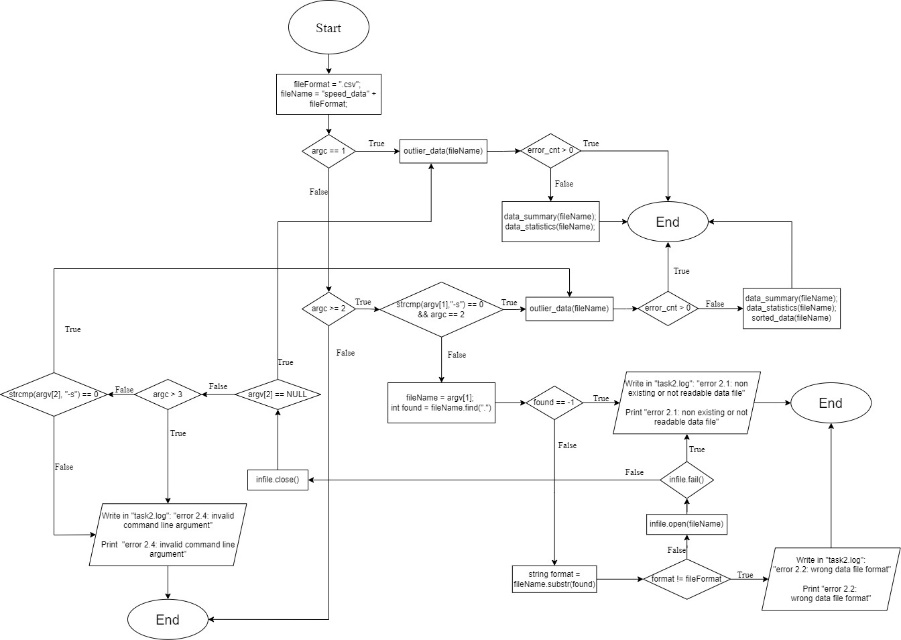
Vo Khoi Thanh Son (45%): Designing the function for all the tasks and in charge of writing the code for each task, writing the report.

Do Quang Thanh (35%): Writing and checking errors for each task and additional requirement.

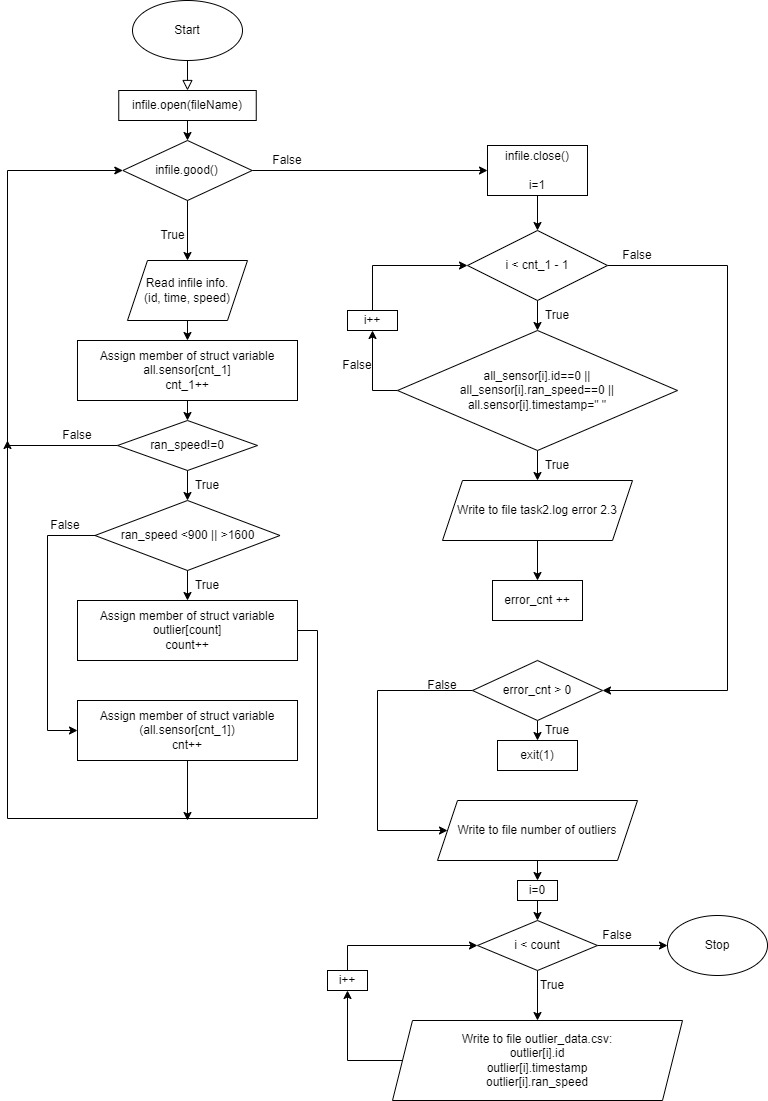
Dinh Quang Anh (20%): Drawing the flowchart for important functions and revise the code for good functionality.

# VI. APPENDIX

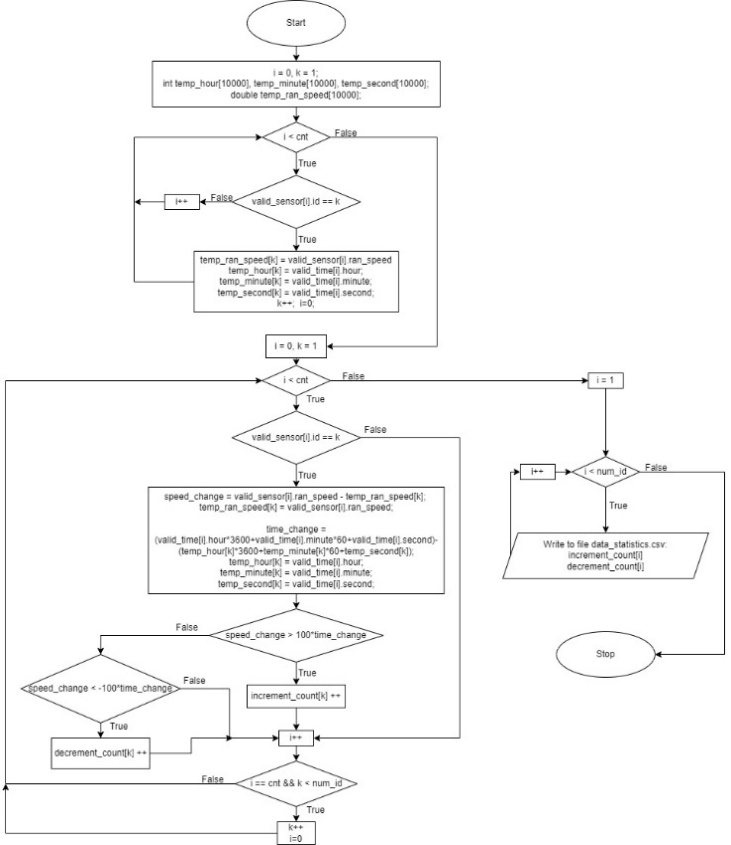
***main() task 2***



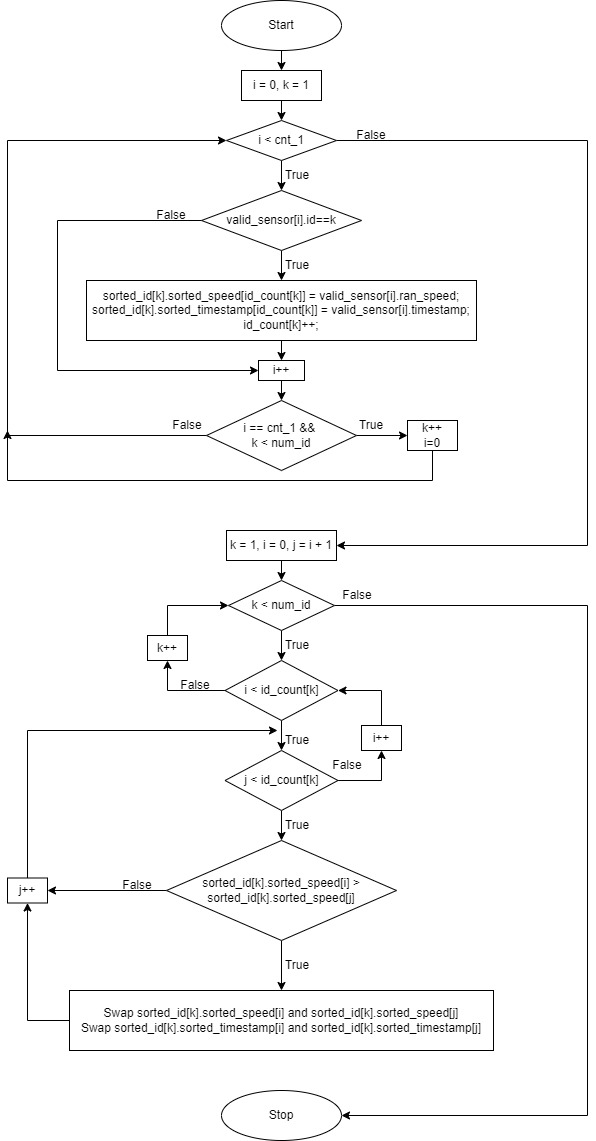
***outlier\_data(fileName)***



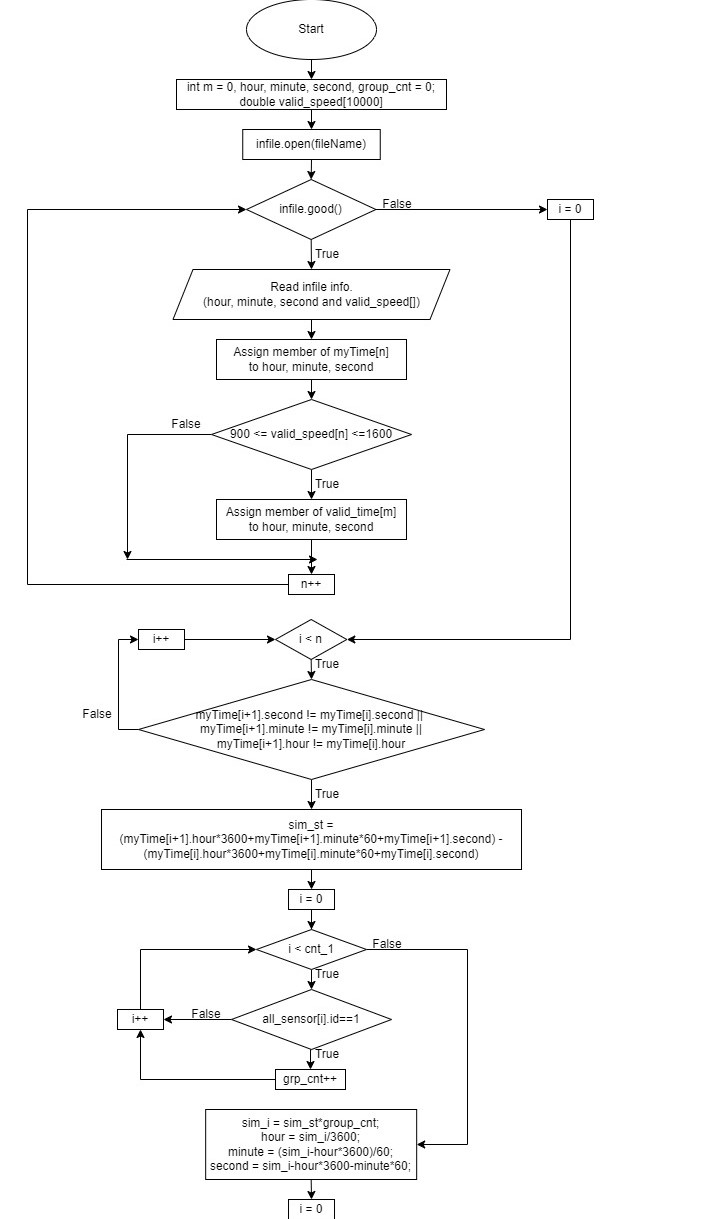
***data\_statistics(fileName)***

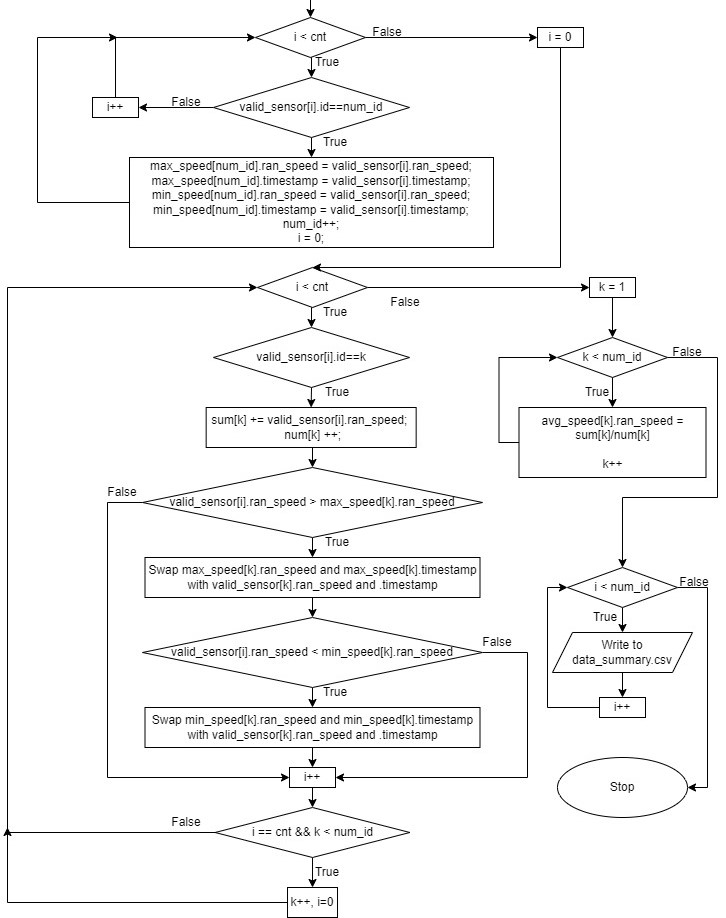


***sorted\_algorithm(fileName)***



***data\_summary(fileName)***





1. VV

   Vo Khoi Thanh Son, Do Quang Thanh, Dinh Quang Anh | 16/7/2022 [↑](#footnote-ref-1)