# Mini project

## I. Project description:

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 the following specification:

- Measurement range: 0 ÷ 3000 rpm (revolution per minutes)
- Resolution: 0.2 rpm

## II. Technical requirements:

The required tasks as below:

### 1. Task 1:

Write a program which allows user to provide the number of sensors (corresponding to number of DC motors which need to be monitored), the sampling time and measurement duration (simulation interval) and the starting time. These inputs should be provided by using command-line as below:

```
C:\\speed_sim -n [num_sensors] -st [sampling] -si [interval]
```

Where:

- `speed_sim`: is the execution file name without extension.
- `-n [num_sensors]` are two arguments to set the number of sensors and must be provided together, `[num_sensors]` should be replaced by a specific number. An error message should be shown if only one of these two are presented. If both of these two are missing, the program will use the default value for the number of sensors which is 1.
- `-st [sampling]` are two arguments to set the sampling time and must be provided together, `[sampling]` should be replaced by a specific number of seconds. An error message should be shown if only one of these two are presented. If both of these two are missing, the program will use the default value for the sampling time which is 10 seconds.
- `-si [interval]` are two arguments to set the simulation interval and must be provided together, `[interval]` should be replaced by a specific number of seconds. An error message should be shown if only one of these two are presented. If both of these two are missing, the program will use the default value for the simulation interval which is 1 hour. Please note that the simulation duration must be greater than the sampling time.

The program will generate a data set consisting of sensor id, the timestamp, and random values of speed with the **starting time of the simulation** is the current time which is based on system time.

- The sensor ids are the number from 1 to the number of sensors, i.e. if the number of sensors provided by the user is 10, we have 10 sensors with the ids are 1, 2, 3, …, 10.
- The timestamp of each speed value is in format of HH:MM:SS (hour:minutes:second), e.g.:15:04:02
- The data set is saved to a file name "speed_data_TIMESTAMP.csv" in which TIMESTAMP is replaced by the **starting time of the simulation** in Unix timestamp in second. This csv file should follow the format of a comma-separated values (CSV) file, please refer here for more information: https://www.ietf.org/rfc/rfc4180.txt

For example: `C:\\speed_sim -n 3 -st 1 -si 30`

The above command will generate "speed_data_1653985800.csv" with the following content:

```
id,time,values
1,08:30:00,1500.6
3,08:30:00,1200.2
2,08:30:00,200.2
1,08:30:03,100.2
3,08:30:03,1200.4
2,08:30:03,1600.2
…
```

The program should be able to generate at least 1000 data points.

2. **Task 2:**

Write another program to process the data in "speed_data_TIMESTAMP.csv" with the content followed the one generated in task 1. The program should be run by using a command-line as below:

```
C:\\speed_process [data_filename.csv]
```

Where:

- `speed_process`: is the execution file name without extension.
- `[data_filename.csv]` is the csv file which consists of the speed data set. If the user does not provide the filename, the program will use the default name which is "speed_data.csv".

For example: `C:\\speed_sim speed_data_1653985800.csv`

The program should be able to handle at least 1000 data points which may be included in the data file.

This program should implement the following sub-tasks:

a. **Tasks 2.1:**

Assuming that the speed of the motor in normal operation are varied within 900 ÷ 1600 rpm (lower bound and upper bound are included). The program needs to validate the speed data in the data file, data points with the speed values which are not in the valid range have to be filtered as outliers. The outlier data points are stored in a file named "outlier_data.csv" which should follow the below format:

```
number of outliers: 4
id, time, values
1, 08:30:00,1700.6
3, 08:30:00,200.6
3, 08:30:03,1900.4
2, 08:30:03,100.2
```

The first line is "number of outliers: X" where X is the number of invalid data points, e.g. 4 in the above example.

b. **Tasks 2.2:**

Identify the largest, smallest, and average speed for each sensor id (apply to the valid data points only), and store the results in a file named "data_summary.csv". The example is as the following:

```
id, parameters, time, values
1, max, 08:30:00,1500.6
1, min, 09:31:03,950.8
1, mean,02:00:00, 1200.5
2, max, 08:35:00,1590.6
2, min, 09:32:03,900.8
2, mean, 02:00:00, 1100.5
3, max, 09:05:02,1200.6
3, min, 09:21:03,920.8
3, mean, 02:00:00, 1000.5
…
```

The time of the max and min values in the file above are the earliest timestamps those values appear. Meanwhile the time of the mean is the simulation interval.

c. **Task 2.3:**

We assume that the motor can change (increase or decrease) the speed by maximum 100 rpm per second. Please count the number of times that speed increment and decrement between two consecutive data points exceed 100 rpm per second for each speed sensor (motor). Hint: the acceleration/deceleration between two consecutive data points should be calculated first.

The results should be stored in a csv file named “data_statistics.csv” with the following format:

```
id, direction, frequency
1, increment, 4
1, decrement, 0
2, increment, 10
2, decrement, 3
3, increment, 0
3, decrement, 5
…
```

**d. Task 2.4 (optional):**

Choose a sorting algorithm and sort the valid data in the data_file.csv if the argument –s is added as below.

```
C:\\speed_process [data_filename.csv] -s
```

The sorted data should be stored in a file name “sorted_data.csv”. The data should be sorted ascendingly by id and values, i.e. the smallest sensor id should appear first, and all the values of the same sensor id should be sorted from the smallest to the largest. The time taken to run the algorithm should be measured and display on the screen as “sorting duration: X seconds” where X is a specific number of the time measured. An example of sorted data file is given as below:

```
sorting duration: 0.02 seconds
id,time,values
1,08:30:03,100.2
1,08:30:00,1500.6
2,08:30:00,200.2
2,08:30:03,1600.2
3,08:30:00,1200.2
3,08:30:03,1200.4
```

## 3. Additional requirements (optional):

Any error occurs should be recorded in an individual line in a log file (normal text file format). The log files for task 1 and task 2 are task1.log and task2.log respectively. The error recorded in the log file must follow the below format:

```
error CODE: description
```

`CODE` should be replaced by an error code in the format as illustrated in section 3a and 3b

Students can figure out more error cases rather than ones listed in 3a and 3b, those cases should be described in the report as well.

**a. Task 1 error log file:**

Some of the errors may happen in task 1 can be recorded in the log file task1.log as below:

```
error 1.1: invalid command line argument
error 1.2: invalid number of sensors
error 1.3: invalid number of sampling time
error 1.4: invalid duration
error 1.5: unable to save data
```

**b. Task 2 error log file:**

Some of the errors may happen in task 1 can be recorded in the log file task1.log as below:

```
error 2.1: non existing or not readable data file
error 2.2: wrong data file format
error 2.3: missing data in line 2
```

The error 2.1 means that the input file may not be presented or corrupted.
The error 2.2 means that the input file is readable, but the content of the file is not in proper format.
The error 2.3 means that there missing data in a particular line such as there is no id, no timestamp or no value or nothing (just an empty line). The first is line 0 which is the header line, the data starts from line 1 or second line in the data file. For example: in line 2 of the following file, the id is missing, thus the error message written in the log file can be "error 2.3: missing data in line 2". There may be multiple lines with missing data, all error lines must be recorded in the log file, e.g.: there is missing data in line 4 in the below sample csv file as well.

| line | speed_data_1653985800.csv |
|---|---|
| 0 | id,time,values |
| 1 | 1,08:30:00,1500.6 |
| 2 | ,08:30:00,1200.2 |
| 3 | 2,08:30:00,200.2 |
| 4 | 1, ,100.2 |
| 5 | 3,08:30:03,1200.4 |
| 6 | 2,08:30:03,1600.2 |
| … | … |

## III. Design:

Students can use top-down approach to design the program, i.e.: it is required to provide a diagram to illustrate the functions developed and their relationship. If the program is written in multiple files, a diagram needs to be provided to illustrate the relationship of the files as well. What the structure is and how the files are designed and organized should also be introduced.

Flowchart diagrams should be provided for important and main functions. Description of the features, inputs and outputs, pre-conditions, post-conditions of these functions should also be presented.

## IV. Coding style:

The coding style should follow GNU as described in this url:
https://www.gnu.org/prep/standards/html_node/Writing-C.html

In summary, the coding style should include the following points:

- The code writing should be formatted well by using proper curly bracket, indentation, line break to differentiate code block.etc.
- Comments should be given to explain the code clearly for the readers.
- Variables and functions should be named properly in English, they should be concise and self-described.
- Hard-coding should be avoided.

Besides, students should not use any non-standard libraries.

## V. Tools:

Editor: Visual studio code (https://code.visualstudio.com/download)
Compiler: gcc or g++ in MinGW-w64 (https://sourceforge.net/projects/mingw-w64/)
Student should specify the operating system (Windows, Linux, MacOS) in the report as well.

## VI. Report and Submission guidelines:

- Students should do the project in a team of maximum 3 members and minimum 2 members (unless the lecturer assigns differently).
- The data file, results, source codes and report files should be in the same folders. Unrelated files should be removed before submission. The structure of the project folder should be as below:

- The parent folder should be name as “groupdid_mini_project.zip” in which “groupid” is replaced by the group id. There are two subfolders “task_1” and “task_2” which consist of the related files (source codes, data file, results) for each task. An example of the project folder is given below

```
221234_222345_223456_mini_project\
  │
  ├── task_1\
  │     │
  │     ├── files of task 1
  │     │
  ├── task_2\
  │     │
  │     ├── files of task 2
  │     │
  │___ report.docx
  │
```

  The whole project folder should be compressed in a **zip** file named as “groupdid_mini_project.zip”, e.g.: “221234_222345_223456_mini_project.zip” then submitted via Team assignment. Please do not use any other compressed file format like *.rar. Students can use 7zip to compress the file, it is free.
- The report.docx is the group report which should be in English, created with Microsoft Word. The main content of the report must include:
  - **Introduction:** The main idea and project file structure (if there are multiple source code files in each task).
  - **Design:** The design of the whole program and each function (if there are too many functions then select a few most important ones). The flowcharts of the main program and important functions (at least 1 flowchart) and brief description of this flow chart.
  - **Results:** The results and discussion on the results.
  - **Conclusion:** The conclusion which should mentioned what have been done and have not been done in your work.
  - **Task allocations:** The task allocation amongst the group and self-evaluate percentages of each member contribution in the project (the total of all members is 100%).
  - **References:** The references if any information in the report is referred in external sources.

  The report should not exceed 4 pages (A4 size) and should strictly follow the attached template.
  Do NOT put the source code in the report.
- Each group must do the project yourself. Do NOT copy from any other groups. Each group should keep the work confidential. If any two or more submissions are similar in source code or report, all the groups will be considered that they do not submit the project, it does not matter who copies from whom.
- No late submission is allowed.
- No submission results in 2 points will be deducted from the mid-term exams (total of the exams is 10 points).
- The submission will be evaluated as below:
  - Complete all the tasks with all the features (each optional task will be given 1 additional point). (4 points)
  - Good design and highly reusable. (2 points)
  - Good coding style with proper naming convention for variables and functions, clean code, meaningful comments. (2 points)
  - The report content must be consistent with the source code, otherwise it will not be marked. The report strictly follows the template. It should be well-organized, comprehensive and easy to understand with minimum grammatical and spelling mistakes. (2 points)
  - Time of submission. Amongst two groups who have the same score, the one who submits earlier takes advantage.

---

**End of the mini project!**