Exercise in #01

Topics covered in this hand in:

- Code standard
- Code structure
- Build tools - compiler
- Makefile
- Functions
- Types, variables

Introduction

In this handin we will write code that looks a bit like code running in one of a vehicle's computers.

The program is a simple control loop[1] reading values from different (faked) peripheral devices. The program does not act on any of the values - it only prints the value to stdout (which is a bit stupid, but hey - this is not a real vehicle).

[1] https://en.wikipedia.org/wiki/Embedded_system#Simple_control_loop

Tasks

Code structure

- Move the code that handles (read) values from the wheel. The code should be put in a separate file as well as a separate directory with appropriate names.

Hint: you 'may' need to add a header file for the wheel functionality

Add engine functionality

- Add a function that initiates the engine code. The function need not do anything.
- Add a function that reads the engine speed (of course this will also be a total fake, see the wheel speed function for inspiration). The value returned should be within the range 1000 to 1500.
- Add code in the main program (in the loop) that reads the engine speed from the engine code, stores it in a variable and prints it out.

Warning feature

- In the loop in the main function you should check if the wheel speed is higher than 1070 and the engine speed is higher than 1300. If it is you should print a warning message on stdout saying something like “You’re pushing things a bit too far”.

Use the makefile

- Make sure that you, or rather the examiners, can use the Makefile to build your source code. Do this by following the instructions in the Makefile itself.

Code standard

- You should use the GNU coding standards[2].

[2] https://www.gnu.org/prep/standards/

Comments (extra, not obligatory)

- Write comments in your code.
- Document the functions using Doxygen and generate a proper API manual.