Lab 4 : Pattern implementation

In this lab you will implement your own version of the GPIO class to replace libGPIO and you will follow the hardware proxy pattern to do so. The example.cpp from Lab 3 will teach you how to handle file access from code, but you will need to make a lot of choices regarding when and how to do what. The are a multitude of solutions, but you are expected to find the best one using the single responsibility principle, encapsulation and maximum code reuse.

Design

Following the hardware proxy pattern you first have to design the GPIO class. You must decide on what attributes it should have and define the operations needed. It should of course be compatible with the code you wrote in lab 3, so the public interface should not change. Use this opportunity to start using StarUML for the class diagram. The diagram used in Lab 3 can be found on BlackBoard and you should use that as a starting point. Think about what private functions you will need in order to maximise code reuse.

Implementation

Implement the necessary functions in your GPIO class and test that they work. In this step it is ok to take the easy way and copy-paste from your own code or from the example in order to quickly test if the code works as expected, but keep an eye open for opportunities to factor out code that is common to multiple operations.

Refactoring

Make sure that you have as little repeated code in your functions. Use private functions to factor out the repetitions. Make sure that your code is readable, has the comments it should and in general follows the code standard. When the code is perfect, send it to can@mmmi.sdu.dk for evaluation.