Leerdoelen per week, week 1 t/m 3 – EMS30

In dit document worden alle leerdoelen die aan bod zijn gekomen in de lessen van week 1 tot en met 3 overzichtelijk weergegeven. Mocht je ergens tegenaan lopen tijdens het maken van de eindopdrachten, dan kun je dit document gebruiken om de nodige informatie te vinden.

Inhoudsopgave

Week 1

Les 1 – Modules in C

Je hebt deze les geleerd hoe je:

- de *interface* van een module in een `.h` bestand kunt *declareren*;
- de *implementatie* van een module in een `.c` bestand kunt *definiëren*;
- deze module kunt gebruiken en testen op een pc;
- deze module kunt gebruiken en testen op een CC3220S LaunchPad.

Les 2 – Dynamisch geheugenallocatie

Je hebt deze les geleerd hoe je:

- dynamisch geheugen kunt alloceren en weer vrij kunt geven;
- een FIFO-buffer kunt implementeren die dynamisch groeit en krimpt met behulp van een *singly linked list*;

- een FIFO-buffer als een *user defined type* kunt definiëren waardoor je meerdere buffers in een programma kunt gebruiken.

Week 2

Les 1 – Git

Je hebt deze les geleerd hoe je:

- softwareontwikkeling kan doen volgens de zogenoemde git-flow;
- gelijktijdig meerdere nieuwe features voor een embedded applicatie kunt ontwikkelen.

Les 2 – Unittesten

Je hebt deze les geleerd hoe je:

- C-code systematisch kan testen met behulp van het eenvoudige test-framework Catch2;
- tests automatisch na elke commit kan laten uitvoeren, door gebruik te maken van Bitbucket pipelines;
- test-stubs kunt gebruiken om software die bedoeld is voor een embedded systeem, toch zo veel mogelijk op een pc te kunnen testen.

Week 3

Les 1 – Memory tests en test coverage

Je hebt deze les geleerd hoe je:

- fouten bij het gebruik van dynamische geheugenallocatie zoals geheugenlekken kunt opsporen met speciale tools zoals valgrind;
- weet dat je genoeg testcode hebt geschreven door gebruik te maken van test coverage metingen.

Les 2 – Coding standards en static code analysis

Je hebt deze les geleerd hoe je:

- gebruik kunt maken van verschillende *C coding standards*;

- gebruik kunt maken van verschillende *static code analysis* tools, waaronder Cppcheck, om de kwaliteit van je C-code te verbeteren.