Opdrachten week 2 les 2 – Unittesten

In week 1 heb je testsoftware geschreven met behulp van een eenvoudige macro. In deze les leer je om gebruik te maken van een eenvoudig test-framework. Er zijn vele commerciële en gratis test-frameworks beschikbaar. Wij maken in deze cursus gebruik van Catch2.

Je leert deze les 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.

Lees nu de Catch2 tutorial globaal door:
https://github.com/catchorg/Catch2/blob/devel/docs/tutorial.md

2.2.1 In les 2 van week 1 heb je een UDT buffer geïmplementeerd en daar eenvoudige testcode voor geschreven. Schrijf nu testcode voor dit UDT met behulp van het test-framework Catch2. Zorg ervoor dat alle code in `buffer.c` goed getest wordt. Je kunt Catch2 in WSL installeren m.b.v. het commando `sudo pacman -Sy catch2`.

In les 1 van week 2 heb je geleerd om te werken volgens de git-flow werkwijze. In deze workflow worden features in aparte branches los van de development branch ontwikkeld. Als er echter een groot aantal features los van elkaar ontwikkeld worden, dan kan het integreren van deze features voor veel problemen zorgen omdat de code op de development branch na het afsplitsen van de feature branch inmiddels al vele wijzigingen heeft ondergaan. De programmeur van een bepaalde feature moet na het voltooien hiervan eerst alle wijzigingen vanuit de development branch doorvoeren in zijn code en testen in combinatie met de nieuwe feature, voordat de feature in de development branch kan worden opgenomen. Dit kan leiden tot wat in de praktijk soms 'integration hell' wordt genoemd.

Het toepassen van continue integratie of in het Engels: 'Continuous Integration'[1] (CI), kan dit probleem voorkomen. Bij het toepassen van deze manier van werken wordt de code op elke feature branch zo vaak mogelijk geïntegreerd met de code op de development branch. Het is daarbij natuurlijk wel van groot belang dat de delopment branch alleen maar werkende,

[1] Zie: https://en.wikipedia.org/wiki/Continuous_integration.

goed geteste code bevat. Er moet dus niet vergeten worden om alle beschikbare testcode na elke commit op de development branch uit te voeren. Door het toepassen van een Bitbucket pipeline[2] kun je ervoor zorgen dat dit automatisch gebeurt zodat het niet vergeten kan worden en zodat een commit alleen maar kan slagen als alle tests succesvol zijn uitgevoerd.

2.2.2 Zorg ervoor dat de tests die je hebt geschreven bij opdracht 2.2.1 automatisch uitgevoerd worden als je een commit doet op de *development* branch.

2.2.3 Breid de UDT buffer uit met een functie om het aantal elementen in het buffer op te vragen:

```
unsigned int number_of_elements_in_buffer(buffer b);
```

Maak hiervoor een feature branch aan (conform gitflow[3]) en schrijf *eerst* de testcode en dan pas de implementatie (conform TDD). Merge de feature branch vervolgens met de development branch en tot slot met de master branch.

CI is lastig toe te passen bij het ontwikkelen van embedded software omdat de software alleen op de embedded hardware echt getest kan worden. Om CI toch toe te kunnen passen bij het ontwikkelen van embedded systemen kunnen zogenoemde test-stubs[4] gebruikt worden. Bijvoorbeeld in de temperatuurlogger die je in les 1 van week 2 hebt ontwikkeld voor de CC3220S LaunchPad, kan de functie om de temperatuur in te lezen vervangen worden door een test-stub functie die steeds 200 teruggeeft (wat overeenkomt met 20 °C). Als het belangrijk is dat de code van de temperatuurlogger met verschillende temperaturen getest wordt, dan kan de stub functie bij elke aanroep een andere (van tevoren vastgelegde) temperatuur teruggeven.

Er zijn diverse tools beschikbaar die je kunnen helpen bij het gebruik van test-stubs, bijvoorbeeld CMock[5] en cmocka[6]. De code van de temperatuurlogger bevat zoveel platformspecifieke code dat het niet zinvol is om deze code op een pc te testen. De in de temperatuurlogger toegepaste UDT buffer heb je natuurlijk al wel op de pc getest.

---

[2] Zie: https://www.atlassian.coteststubm/continuous-delivery/tutorials/continuous-integration-tutorial.

[3] Installeer, als je dat nog niet gedaan hebt, de volgende plugin in Visual Studio Code: https://marketplace.visualstudio.com/items?itemName=vector-of-bool.gitflow.

[4] Zie: https://en.wikipedia.org/wiki/Test_stub.

[5] Zie: https://www.throwtheswitch.org/cmock.

[6] Zie: https://cmocka.org/