Eindverslag Software Development

Groep: Groep 2

Studenten:

Janek van Wolfswinkel

Wiebe de Boer

Docent: Jos Foppele

Onderwijsinstelling: NHLStenden

# Versiebeheer

|  |  |  |
| --- | --- | --- |
| **Datum** | **Versie** | **Omschrijving** |
| 18-9-2020 | 0.1 | Eerste opzet plan van aanpak |
| 23-9-2020 | 0.2 | Aanvullingen, o.a. planning |
| 23-9-2020 | 0.2.1 | UML Analyse |
| 25-9-2020 | 0.2.2 | Test Plan |
| 30-9-2020 | 0.3 | Correcties plan van aanpak |
| 11-11-2020 | 0.4 | Eerste opzet eindverslag met inleiding, protocol, uml analyse en eerste testresultaten |
| 12-11-2020 | 0.4.1 | Libraries |
| 15-11-2020 | 0.5 | Verdere test resultaten 13-11 en uitbreiding test plan |
| 18-11-2020 | 0.5.1 | Tekst en diagram correcties en test case tabellen |
| 20-11-2020 | 0.5.2 | Test resultaten 20-11 en planning |
| 27-11-2020 | 0.5.3 | Test resultaten 27-11 |
| 1-12-2020 | 0.5.4 | Extra Testresultaten |
| 4-12-2020 | 0.6 | Test resultaten eindtest |
| 12-2020 | 1.0 | Laatste Correcties en resultaten |

# Inhoudsopgave

[Inleiding 4](#_Toc56767675)

[Planning 5](#_Toc56767678)

[UML Analyse 6](#_Toc56767679)

[Use Case Diagram 6](#_Toc56767680)

[Klasse Diagram Simulator 7](#_Toc56767681)

[Klasse Diagram Controller 7](#_Toc56767682)

[Component Diagram 8](#_Toc56767683)

[Deployment Diagram 8](#_Toc56767684)

[Protocol 9](#_Toc56767685)

[Over Protocol 9](#_Toc56767686)

[Over Verkeerslichten 9](#_Toc56767687)

[Header 9](#_Toc56767688)

[Socket Voorbeeld 10](#_Toc56767689)

[Testplan 11](#_Toc56767690)

[Controller 11](#_Toc56767691)

[Simulator 11](#_Toc56767692)

[Test project 11](#_Toc56767693)

[Uitwisseling 11](#_Toc56767694)

[Testresultaten 13](#_Toc56767695)

[Testresultaten Eerst Optionele Sessie 23-10 13](#_Toc56767696)

[Testresultaten Eerst Sessie 30-10 14](#_Toc56767697)

[Resultaten 14](#_Toc56767698)

[Eigen Test Sessie Groep 2 14](#_Toc56767699)

[Testresultaten Tweede Optionele Sessie 6-11 15](#_Toc56767700)

[Testresultaten Tweede Sessie 13-11 16](#_Toc56767701)

[Resultaten 17](#_Toc56767702)

[Eigen Collisie Test 17](#_Toc56767703)

[Testresultaten Extra Sessie 20-11 18](#_Toc56767704)

[Resultaten 19](#_Toc56767705)

[Testresultaten 27-11 20](#_Toc56767706)

[Resultaten 21](#_Toc56767707)

[Testresultaten Eindsessie 22](#_Toc56767708)

[Resultaten 23](#_Toc56767709)

[Bronverwijzingen 24](#_Toc56767710)

# Inleiding

In dit eindverslag van software development van 2020 staan de volgende elementen:

* Planning
* UML Analyse
* Protocol
* Testplan
* Testresultaten

Het plan van aanpak is als extra apart document opgenomen.

## Aanleiding en Eisen

De eisen van het systeem zijn correctheid, overdraagbaarheid, robuustheid, efficiëntie, fouttolerantie en herstartmogelijkheid na een calamiteit. De controller en simulator moeten in verschillende talen geschreven zijn.

Het systeem bestaat uit: de controller stuurt de verkeerslichten aanstuurt en een simulatie die verkeer simuleert (inclusief verkeerslichten). Optioneel is een regressietester die de communicatie controleert.

## Doel

Doel van het project was een correct, robuust, efficiënt en fouttolerant concurrent systeem te bouwen.

# Planning

Groep 2 heeft onderling afgesproken om woensdag bijeen te komen en ook vrijdag na het college van software development.

De lange termijn planning is als volgt:

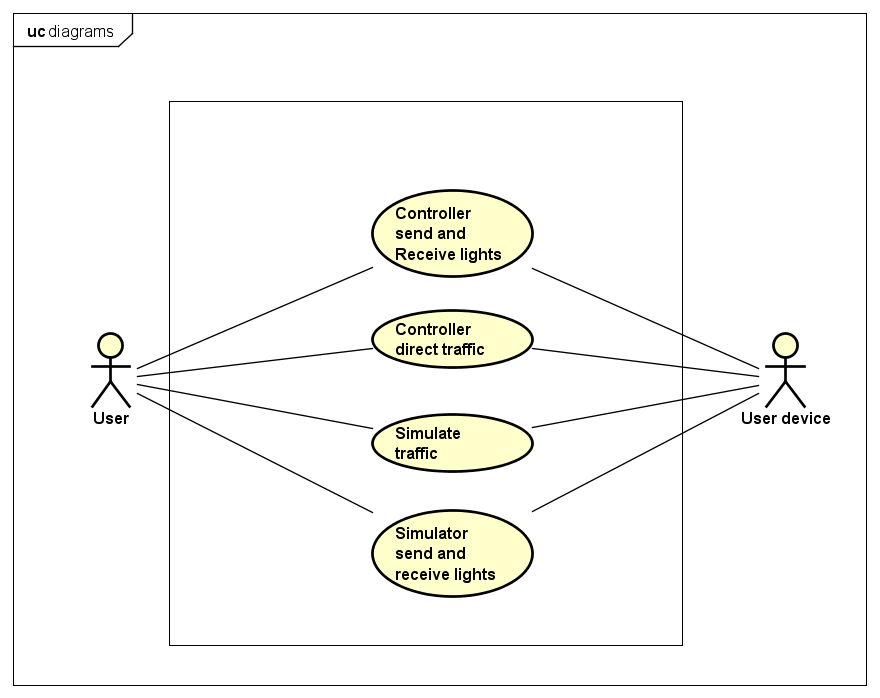
|  |  |  |  |
| --- | --- | --- | --- |
| Algemene Bijeenkomsten | Deadlines en overig | Testsessies | Onderzoek |
| 11-9 |  |  |  |
| 18-9 |  |  |  |
| 25-9 |  |  |  |
| 2-10 | Controle en feedback kaart Roland |  | Header Sidney |
| 9-10 | Socket verzenden en ontvangen |  | Conventies header |
| 23-10 | Simulator en controller socket 1 baan | Optionele test 1 baan | Json socket testen |
| 30-10 |  | Test 1 baan |  |
| 6-11 | Simulator en controller alle banen | Optionele test |  |
| 13-11 |  | Test alle banen |  |
| 20-11 |  | Extra Test |  |
| 27-11 |  | Extra test |  |
| 4-12 | Inleveren blackboard | Eind Test |  |

Er is voor de planning van het programmeren en overige taken zoals de analyse en verslagen ook gebruik gemaakt van trello met scrum sprints, met een todo, doing, testing/controle en done lijsten.

# UML Analyse

In de UML analyse zijn die diagrammen te zien van het project, met de controller en verkeerssimulator.

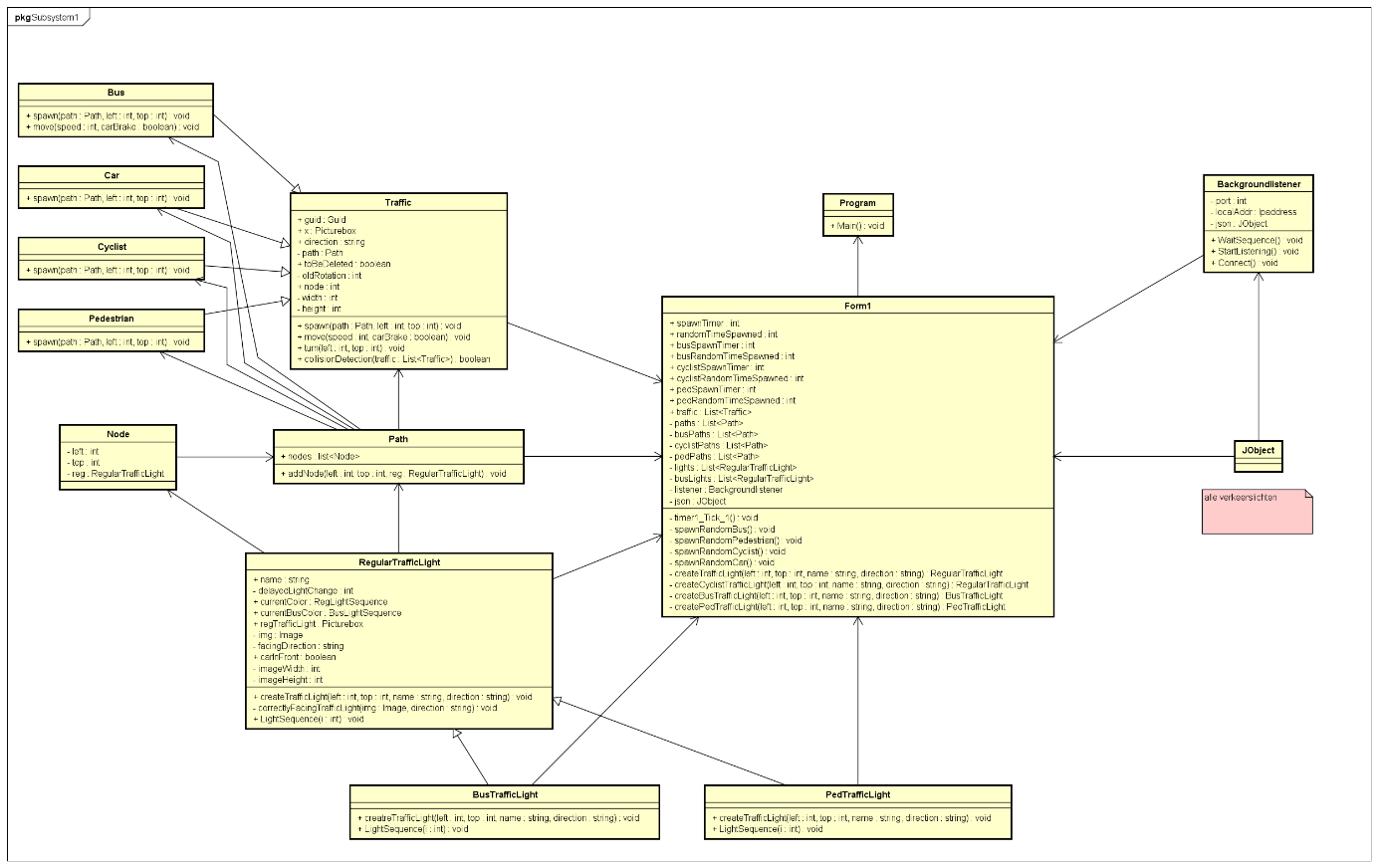
## Use Case Diagram



Figuur 1 Use Case Diagram

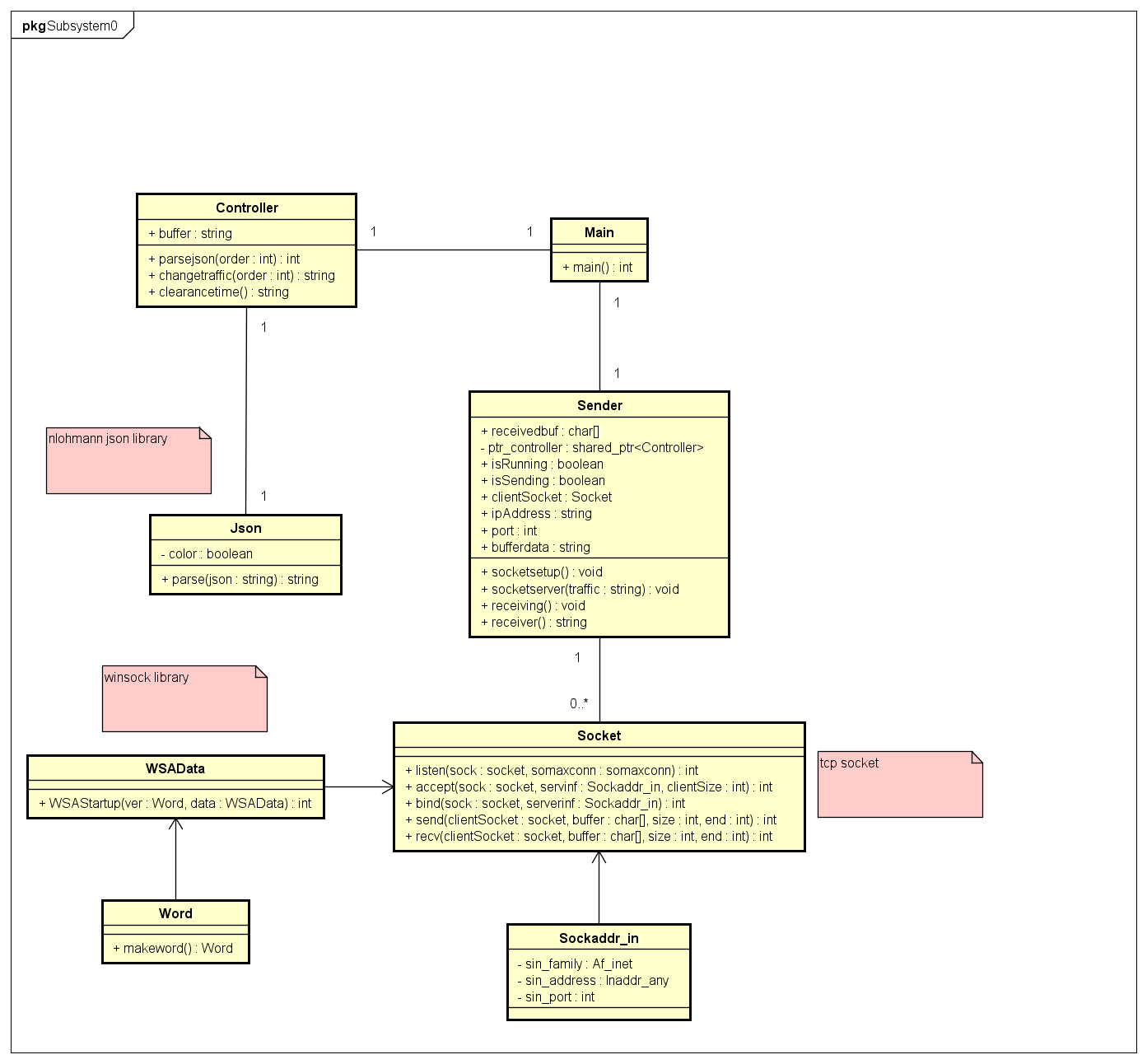
## Klasse Diagram Simulator

De simulator maakt gebruik van windows forms en voor het parsen van de json van newtonsoft.



Figuur 2 Klasse Diagram Simulator

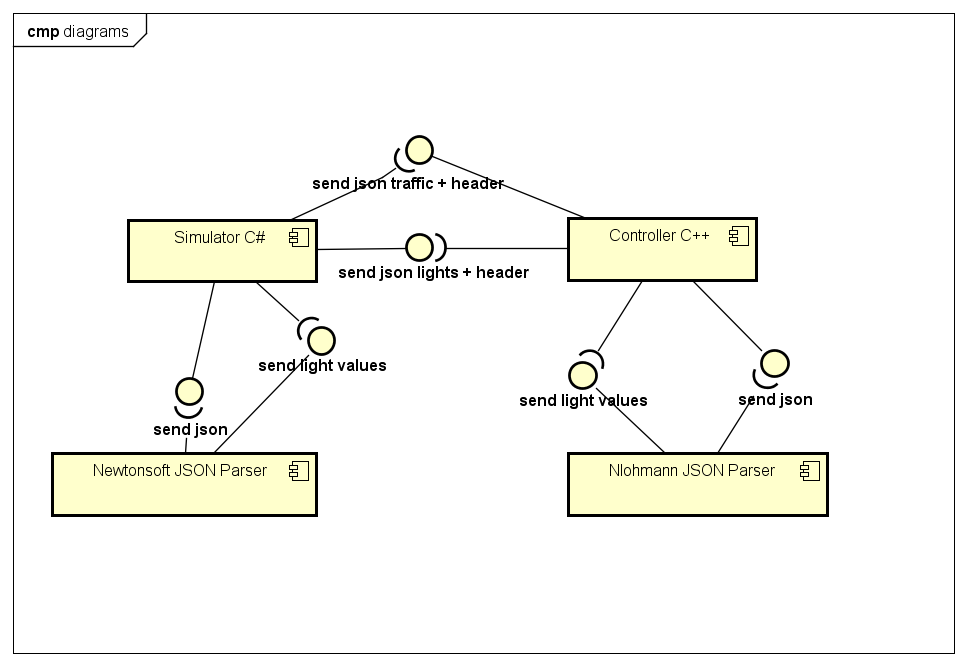
Klasse Diagram Controller

De controller maakt gebruik van winsock voor het aanmaken van de tcp socket en voor het parsen van de json wordt gebruik gemaakt van nlohmann json.

Figuur 3 Klasse Diagram Controller

## Component Diagram

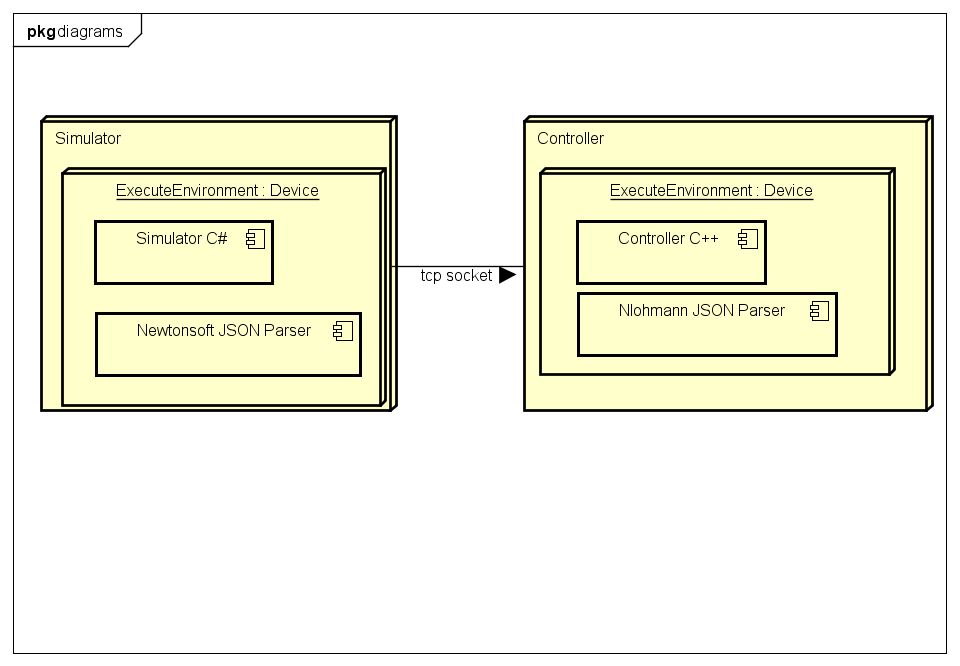
Json string met een header ervoor wordt verzonden over en weer met een tcp socket. In de json zijn de waarden 0 voor rood verkeerslicht en 1 voor groen verkeerslicht.



Figuur 4 Component Diagram

## Deployment Diagram

Het binden van de socket gebeurd in de controller. Deze fungeert in feite als een soort van tcp socket. De fasering en aansturen van verkeerslichten gebeurd volgens het protocol in de controller.



Figuur 5 Deployment Diagram

# Protocol

Het protocol is gedeeld op een drive met de overige groepen. Het is opgesteld in overleg met de andere groepen bij de bijeenkomsten op vrijdag.

De link naar dat protocol is te vinden op: <https://onedrive.live.com/?authkey=%21AJe%5FlPMJO4YgtWo&id=6DC6103027A7EFC3%211707&cid=6DC6103027A7EFC3>

## Over Protocol

De bytes encoding is in utf8. Dit om problemen met de socket te voorkomen.

## Over Verkeerslichten

De waarden in de json string staan voor het volgende:

Rood licht = 0, groen licht = 1

## Header

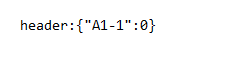
Om te voorkomen dat een bericht deels uitgelezen wordt bij de tcp socket wordt een header ervoor verzonden.

Voor het goed communiceren met de verschillende clients en de controller hebben we een standaard header format gekozen. Aan de hand van deze header is het voor de controller en de client duidelijk hoe lang de package is. Op deze manier ontstaat er tijdens het communiceren geen fouten.



Figuur 6 header en data

De header is een kleine toevoeging aan de normale package en wordt voor de daadwerkelijke data gezet. De header bestaat uit de lengte van de data en gebruikt een divider om aan te geven wanneer de data begint. Als divider is er gekozen voor de char ‘:’.



Figuur 7 Voorbeeld header en data

In figuur is een voorbeeld te zien van een header (van 1 baan). Als standaard wordt gezorgd dat er altijd alle whitespaces verwijderd wordt in de data van de package, dit wordt gedaan zodat er zo min mogelijk data wordt gestuurd.

De header is in dit voorbeeld geval dus 25:



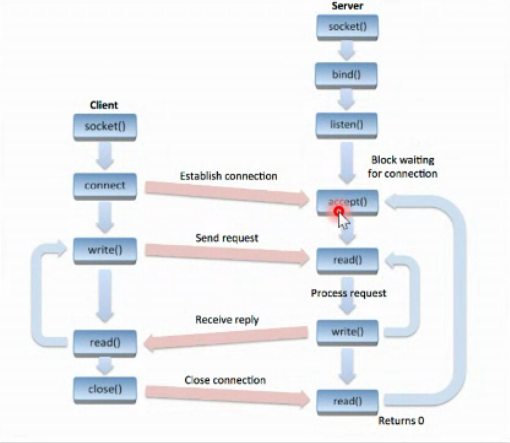
Figuur 8 Voorbeeld plakken header aan het package

Type Verzonden Data

De data verzonden over de socket is JSON.

## Socket Voorbeeld

In het protocol zat ook een voorbeeld van sockets. Het was aan te raden dat voorbeeld te volgen voor de mensen die onbekend waren met sockets.



Figuur 9 Voorbeeld Flowchart Socket Server (Controller) en Client (Simulator)

# Testplan

## Controller

Voor de controller worden unit tests gemaakt. Er is een unit test voor elke belangrijke functie (dus geen getters en setters) in de controller klasse, maar de functies omtrent de controller betreffende verzenden en ontvangst en het parsen en niet de socket. Bij aanpassingen in de code, doordat bijvoorbeeld er veranderingen zijn het protocol of socket wordt deze unit test van de functie gebruikt als regressie test om te zien of de functie nog goed werkt en is zoals verwacht.

Voor de socket wordt ook een unit test gemaakt en een json string verstuurd en gecontroleerd of deze goed wordt opgevangen met een assert.

Er wordt voor de unit test een mock gebruikt zodat deze verwachte waarde assert kan worden.

## Simulator

Voor de simulatie worden eveneens unit tests gemaakt. Minstens één unit test voor iedere (belangrijke, dus geen getters/setters) functie behalve de socket en de private functies. Ook hierbij worden regressietests toegepast zodat de werking van een functie kan worden getest aan de eisen die hier eerder aan gesteld zijn, dus of de werking nog zoals verwacht is.

Eveneens wordt de socket getest of berichten(JSON) vanuit de controller op de juiste manier wordt afgevangen. Ook het schrijven van data(JSON) vanuit de socket wordt getest.

Hierbij wordt ook gebruik gemaakt van een mock string, zodat het verwachte resultaat op een veilige manier gecontroleerd kan worden die niet invloed heeft op het daadwerkelijke programma.

## Test project

Voor zowel de simulator als controller is er een eigen test project in visual studio.

## Uitwisseling

Door middel van downloaden en uitwisseling wordt ook getest of de controller werkt met andere deelproducten van andere groepen.

Hierbij wordt getest dat er:

* Json verzonden wordt
* Json ontvangen wordt
* Of het de juiste json is
* De programma’s starten
* Of er geen collisie is

Dit zodat de controller en simulator acceptabel werkt met de package, met andere woorden json en header.

Tevens wordt gekeken of de deelproducten werken op het windows 10 systeem van de andere groepen.

De te beantwoorden vragen bij het test plan zijn bij de fase met 1 baan en met alle banen zijn:

* Start de simulator? (simulator met systeem)
* Start de controller? (controller met systeem)
* Wordt JSON verzonden? (tcp socket)
* Wordt JSON ontvangen? (tcp socket)
* Is de JSON correct?

Optioneel fase:

* Is dit ontvangen te zien in bijvoorbeeld de console?
* Wordt er terug verzonden en ontvangen (dus van simulator naar controller)?

Simulator collisie fase:

* Gaan de verkeerslichten afwisselend op groen, oranje, rood?
* Kruist verkeer niet meer in de simulator?

Dit wordt per groep bekeken en de pass of geen pass bijgehouden in een excel werkblad volgens het stramien van de testcase zoals te zien is bij de resultaten van de laatste optionele test en eind test sessies.

# Testresultaten

Bij de testresultaten worden de resultaten weergegeven van de afzonderlijke test sessies. Er zijn test case tabellen gemaakt voor het testen. De resultaten worden ook beschreven per groep, waar dat nodig is. Test stappen waarbij er sprake is van uitwisseling met groepen worden met een \* weergegeven. In het rood is in de test case tabel weergegeven of de stap faalt, oranje als het faalt van een andere groep en groen als de stap slaagt.

## Testresultaten Eerst Optionele Sessie 23-10

Het terugontvangen was nog niet klaar, er is voor één baan getest in plaats van al het verkeer.

Het resultaat bij het testen met groep 4 werkte niet. De simulator wilde niet starten. Dit kwam doordat het programma in mappen te diep was opgeslagen.

De controller van groep 7 werkte met de simulator van groep 2. De simulator returnde een 1 in de console. Daaruit maakten we op dat het één baan ontving.

Groep 6 had problemen met de controller opstarten. Wel was de header vlug aangepast. Er is voor de volgende test sessie afgesproken dat er geen vlugge aanpassingen meer zouden komen, omdat dit teveel tijd zou nemen.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testcase: | Eerste Optionele Socket Test | Testcase beschrijving: | Eerste optionele sockettest |  |  |
| Stappen | Beschrijving | Soort | Verwachte Uitkomst | Werkelijke Uitkomst | Testresultaat |
| 1 | Controller Start | handmatig | Controller start na een tijdje zonder foutmeldingen | Start niet bij 1 groep | 2x pass, 1x fail |
| 2 | Simulator Start | handmatig | Simulator start zonder foutmeldingen | start | pass |
| 3\* | Controller zend json en Header | handmatig | Controller zend over de tcp socket json met een header | Werkt met 2 groepen | 2x pass, 1x fail |
| 4\* | Simulator ontvangt json en header | handmatig | Simulator ontvangt over de tcp socket json met een header | Werkt met 2 groepen | 2x pass, 1x fail |

Tabel 1 Test case en resultaten eerste socket test

## Testresultaten Eerst Sessie 30-10

Het terug ontvangen was nog niet klaar, er is voor één baan getest in plaats van al het verkeer.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testcase: | Eerste Socket Test | Testcase beschrijving: | Eerste sockettest |  |  |
| Stappen | Beschrijving | Soort | Verwachte Uitkomst | Werkelijke Uitkomst | Testresultaat |
| 1 | Controller Start | handmatig | Controller start na een tijdje zonder foutmeldingen | Start niet bij 2 groepen | 3x pass, 2x fail |
| 2 | Simulator Start | handmatig | Simulator start zonder foutmeldingen | start | pass |
| 3\* | Controller zend json en Header | handmatig | Controller zend over de tcp socket json met een header | Werkt met 2 groepen | 2x pass, 3x fail |
| 4\* | Simulator ontvangt json en header | handmatig | Simulator ontvangt over de tcp socket json met een header | Werkt met 2 groepen | 2x pass, 3x fail |
| 5 | Json is correct | handmatig | De json is volledig en kan correct geparsed worden | Fasen wisselen niet af | fail |

Tabel 2 Test case en resultaten eerste socket test

### Resultaten

De volgende groepen deden mee met de test sessie: 2, 3, 6, 7, 8, 13.

De controller en simulator werkte met twee groepen. Het ontving json en returnde een 1 van één baan in de console. Wel merkte Roland van groep 3 op dat de fasen hetzelfde bleven. Het ontving dus wel json, maar niet op elk moment de juiste json.

## Eigen Test Sessie Groep 2

De json fasen zijn voor de optionele testsessie zijn ook nog getest door groep 2 zelf op de woensdag 4 november voor de optionele test sessie op 6 november. Daaruit bleek dat de fasen nu wel afwisselden, maar dat was met de eigen controller en simulator.

In de console van de simulator was te zien dat het afwisselend 0 of 1 weergaf van de ene geteste baan. Daaruit kon opgemaakt worden dat de fasen afwisselden in de controller. Er was nog collisie van verkeer.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testcase: | Eigen test | Testcase beschrijving: | Optionele eigen test |  |  |
| Stappen | Beschrijving | Soort | Verwachte Uitkomst | Werkelijke Uitkomst | Testresultaat |
| 1 | Controller Start | handmatig | Controller start na een tijdje zonder foutmeldingen | Start | pass |
| 2 | Simulator Start | handmatig | Simulator start zonder foutmeldingen | Start | pass |
| 3 | Json is correct | handmatig | De json is volledig en kan correct geparsed worden | Correct | pass |
| 4 | Simulator lichten wisselen | handmatig | De verkeerslichten wisselen in de simulator naar verloop van tijd | Gezien dat het wisselt | pass |
| 5 | Simulator geen collisie | handmatig | Er kruist geen verkeer over elkaar heen | Wel collisie gezien | fail |

Tabel 3 Testcase en resultaten eigen testsessie

## Testresultaten Tweede Optionele Sessie 6-11

Het terug ontvangen was nog niet klaar, er is voor één baan getest in plaats van al het verkeer. Slechts twee groepen hadden zich ingeschreven te weten groep 2 en groep 4.

Het resultaat bij het testen met groep 4 slaagde niet. De simulator van groep 4 werd door windows defender verhinderd te starten.

De readme van groep 4 was ook nog niet aangepast in de github repo.

## Testresultaten Tweede Sessie 13-11

Het terugontvangen vanaf de simulator was klaar, er is voor alle banen getest van al het verkeer zoals in de lange termijn planning stond beschreven.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testcase: | Tweede Socket Test | Testcase beschrijving: | Tweede sockettest |  |  |
| Stappen | Beschrijving | Soort | Verwachte Uitkomst | Werkelijke Uitkomst | Testresultaat |
| 1 | Simulator verkeerslicht creatie en json parser | unit | Naam hetzelfde in de lijst, geparse waarde is 1 of 0 | Hetzelfde lijst en integer 1 | pass |
| 2 | Controller parser en traffic | unit | Geparse waarde is 1 en string is overeenkomstig mock string | Integer 1 en overeenkomstige string | pass |
| 3\* | Controller Start | handmatig | Controller start na een tijdje zonder foutmeldingen | Start voor groep 3,4, 7,8 | 4x pass, overig fail |
| 4 | Simulator Start | handmatig | Simulator start zonder foutmeldingen | start | pass |
| 5\* | Controller zend json en Header | handmatig | Controller zend over de tcp socket json met een header | Groep 3, 7 | 2x pass, overig fail |
| 6\* | Simulator ontvangt json en header | handmatig | Simulator ontvangt over de tcp socket json met een header | Groep 3, 7 | 2x pass, overig fail |
| 7\* | Json is correct | handmatig | De json is volledig en kan correct geparsed worden | Groep 3, 7 ontvangt | 2x pass, overig fail |
| 8 | Simulator zend terug | handmatig | Simulator zend json en header terug naar de controller | Stuurt waarden terug | pass |
| 9 | Controller ontvangt | handmatig | Controller ontvangt json en header van de simulator terug | controller hangt bij terugzenden simulator groep 3 | fail |

Tabel 4 Test case en resultaten tweede socket test

### Resultaten

De resultaten van de test met de andere groepen waren als volgt:

De controller van groep 1 start niet.

De controller van groep 3 start wel, maar geeft assertion failed melding bij het terug ontvangen van de simulator. Dit komt doordat de volgorde van de json namen van de verkeerslichten anders is.

De controller van groep 4 scherm resolutie van de laptop was te laag waardoor de button niet zichtbaar was.

De repository van groep 5 was leeg. Deze kon dus niet getest worden.

De json van groep 6 was niet compleet, maar hun controller startte wel.

De controller van groep 7 werkt en er wordt json en header correct ontvangen.

De controller van groep 8 had een onjuist port nummer, dit was 50131 in plaats van 54000. Er werd dus geen socket verbinding gemaakt.

De controller van groep 13 geeft een error bij het opstarten.

Groep 16 was niet aanwezig bij de test sessie.

Er is later op dezelfde dag nog een keer getest met groep 3. Door de controller van groep 2 concurrent te maken met een thread bij het terug ontvangen van de simulator van groep 3 werkte dit wel. Het startte, het verbond, er werd json en header over en weer verzonden.

Verder waren er in de simulator nog collission tussen voetverkeer en autoverkeer waarbij voetgangers nog konden kruisen met auto’s wanneer het auto lichten op groen gingen en de voetganger nog halverwege op de weg was.

### Eigen Collisie Test

De collisie is nog bekeken daarna en aangepast. Er was na de aanpassingen geen collisie meer te zien tussen autoverkeer en fietsers en voetgangers.

## Testresultaten Extra Sessie 20-11

Om het handmatig testen met de eind test sessies gestructureerd te houden is een test case gemaakt en later ingevuld wanneer de daadwerkelijke testsessie plaatsvond.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testcase: | Eindtest | Testcase beschrijving: | Optionele eindgebruikerstest |  |  |
| Stappen | Beschrijving | Soort | Verwachte Uitkomst | Werkelijke Uitkomst | Testresultaat |
| 1 | Simulator verkeerslicht creatie en json parser | unit | Naam hetzelfde in de lijst, geparse waarde is 1 of 0 | Hetzelfde lijst en integer 1 | pass |
| 2 | Controller parser en traffic | unit | Geparse waarde is 1 en string is overeenkomstig mock string | Integer 1 en overeenkomstige string | pass |
| 3\* | Controller Start | handmatig | Controller start na een tijdje zonder foutmeldingen | Groep 3, 4, 5, 6, 7, 8 start, groep 13 niet door scanner | 6x pass, 1x fail |
| 4 | Simulator Start | handmatig | Simulator start zonder foutmeldingen | Start | pass |
| 5\* | Controller zend json en Header | handmatig | Controller zend over de tcp socket json met een header | Groep 3, 5, 6, 7, 8 wel, groep 4 niet | 5x pass, 1x fail |
| 6\* | Simulator ontvangt json en header | handmatig | Simulator ontvangt over de tcp socket json met een header | Groep 3, 5, 6, 7, 8 wel, groep 4 niet | 5x pass, 1x fail |
| 7\* | Json is correct | handmatig | De json is volledig en kan correct geparsed worden | Groep 3 json correct, niet correct voor 5, 6, 7, 8 | 1x pass, 4x fail |
| 8 | Simulator zend terug | handmatig | Simulator zend json en header terug naar de controller | Zend terug | pass |
| 9\* | Controller ontvangt | handmatig | Controller ontvangt json en header van de simulator terug | Niet iedere groep stuurt terug, maar controller kan ontvangen mits json correct | 1x pass |
| 10 | Simulator lichten wisselen | handmatig | De verkeerslichten wisselen in de simulator naar verloop van tijd | Gezien dat het wisselt | pass |
| 11 | Simulator geen collisie | handmatig | Er kruist geen verkeer over elkaar heen | Geen collisie gezien | pass |

Tabel 5 Testcase en resultaten optionele testsessie

### Resultaten

De resultaten van de test met de andere groepen waren als volgt:

Groep 1 was niet aanwezig.

Groep 3 ging goed.

Groep 4 was later niet meer aanwezig en socket ging niet goed, omdat de solution onveranderd was en de button nog steeds bruikbaar met de lage schermresolutie.

Groep 5 fasen wisselden niet.

Groep 6 json terminator character.

Groep 7 fasen wisselden niet.

Groep 8 json niet correct, had spaties erin.

Groep 13 virusscanner zorgde voor problemen met starten.

Groep 16 was niet aanwezig.

Over het terugsturen van de simulator is afgesproken dat dit moet, maar dat optioneel is of de controller dit ontvangt en er iets mee doet.

## Testresultaten 27-11

Voor de eindtesten zijn dezelfde testcase gebruikt. De resultaten en uitkomsten zijn te vinden in de volgende tabel.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testcase: | Eindtest | Testcase beschrijving: | Optionele eindgebruikerstest |  |  |
| Stappen | Beschrijving | Soort | Verwachte Uitkomst | Werkelijke Uitkomst | Testresultaat |
| 1 | Simulator verkeerslicht creatie en json parser | unit | Naam hetzelfde in de lijst, geparse waarde is 1 of 0 | Hetzelfde lijst en 1 integer | pass |
| 2 | Controller parser en traffic | unit | Geparse waarde is 1 en string is overeenkomstig mock string | Integer 1 en overeenkomstige string | pass |
| 3\* | Controller Start | handmatig | Controller start na een tijdje zonder foutmeldingen | Groep 13 en 1 start niet, overig wel | 6x pass, 2x fail |
| 4 | Simulator Start | handmatig | Simulator start zonder foutmeldingen | Start | pass |
| 5\* | Controller zend json en Header | handmatig | Controller zend over de tcp socket json met een header | Groep 5 zend geen json, overig wel die wilt starten | 5x pass, 3x fail |
| 6\* | Simulator ontvangt json en header | handmatig | Simulator ontvangt over de tcp socket json met een header | Groep 4 heeft incorrecte json, filtert spaties eruit | 5x pass, 3x fail |
| 7\* | Json is correct | handmatig | De json is volledig en kan correct geparsed worden | Groep 4 heeft incorrecte json | 4x pass, 4x fail |
| 8 | Simulator zend terug | handmatig | Simulator zend json en header terug naar de controller | Zend terug | pass |
| 9\* | Controller ontvangt | handmatig | Controller ontvangt json en header van de simulator terug | Groep 4 en 5 was niet correcte json | 4x pass, 4x fail |
| 10 | Simulator lichten wisselen | handmatig | De verkeerslichten wisselen in de simulator naar verloop van tijd | Gezien dat het wisselt | pass |
| 11 | Simulator geen collisie | handmatig | Er kruist geen verkeer over elkaar heen | Geen collisie gezien | pass |

Tabel 6 Testcase en resultaten eerste eindgebruikerstest

### Resultaten

De resultaten van de test met de andere groepen waren als volgt:

Groep 1 startte de controller niet.

Groep 3 ging goed.

Groep 4 jar werkt wel bij Janek, omdat hij de juiste Java versie heeft, maar ze hebben wel spaties in de json.

Groep 5 zend niet door incorrecte json ontvangst.

Groep 6 ging goed.

Groep 7 ging goed.

Groep 8 ging goed.

Groep 13 Controller start niet.

Groep 16 was niet aanwezig.

Testresultaten 1-12

Voor de eindtesten zijn dezelfde testcase gebruikt. De resultaten en uitkomsten zijn te vinden in de volgende tabel.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testcase: | Eindtest | Testcase beschrijving: | Optionele eindgebruikerstest |  |  |
| Stappen | Beschrijving | Soort | Verwachte Uitkomst | Werkelijke Uitkomst | Testresultaat |
| 1 | Simulator verkeerslicht creatie en json parser | unit | Naam hetzelfde in de lijst, geparse waarde is 1 of 0 | Hetzelfde lijst en 1 integer | pass |
| 2 | Controller parser en traffic | unit | Geparse waarde is 1 en string is overeenkomstig mock string | Integer 1 en overeenkomstige string | pass |
| 3\* | Controller Start | handmatig | Controller start na een tijdje zonder foutmeldingen |  |  |
| 4 | Simulator Start | handmatig | Simulator start zonder foutmeldingen | Start | pass |
| 5\* | Controller zend json en Header | handmatig | Controller zend over de tcp socket json met een header |  |  |
| 6\* | Simulator ontvangt json en header | handmatig | Simulator ontvangt over de tcp socket json met een header |  |  |
| 7\* | Json is correct | handmatig | De json is volledig en kan correct geparsed worden |  |  |
| 8 | Simulator zend terug | handmatig | Simulator zend json en header terug naar de controller | Zend terug | pass |
| 9\* | Controller ontvangt | handmatig | Controller ontvangt json en header van de simulator terug |  |  |
| 10 | Simulator lichten wisselen | handmatig | De verkeerslichten wisselen in de simulator naar verloop van tijd | Gezien dat het wisselt | pass |
| 11 | Simulator geen collisie | handmatig | Er kruist geen verkeer over elkaar heen | Geen collisie gezien | pass |

Tabel 7 Testcase en resultaten eerste eindgebruikerstest

### Resultaten

De resultaten van de test met de andere groepen waren als volgt:

Groep 1

Groep 3 ging goed.

Groep 4

Groep 5

Groep 6

Groep 7

Groep 8

Groep 13

Groep 16

## Testresultaten Eindsessie

Voor de eindtesten zijn dezelfde testcase gebruikt. De resultaten en uitkomsten zijn te vinden in de volgende tabel.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testcase: | Eindtest | Testcase beschrijving: | Optionele eindgebruikerstest |  |  |
| Stappen | Beschrijving | Soort | Verwachte Uitkomst | Werkelijke Uitkomst | Testresultaat |
| 1 | Simulator verkeerslicht creatie en json parser | unit | Naam hetzelfde in de lijst, geparse waarde is 1 of 0 | Hetzelfde en 1 integer | pass |
| 2 | Controller parser en traffic | unit | Geparse waarde is 1 en string is overeenkomstig mock string | Interger 1 en overeenkomstige string | pass |
| 3\* | Controller Start | handmatig | Controller start na een tijdje zonder foutmeldingen |  |  |
| 4 | Simulator Start | handmatig | Simulator start zonder foutmeldingen | start | pass |
| 5\* | Controller zend json en Header | handmatig | Controller zend over de tcp socket json met een header |  |  |
| 6\* | Simulator ontvangt json en header | handmatig | Simulator ontvangt over de tcp socket json met een header |  |  |
| 7\* | Json is correct | handmatig | De json is volledig en kan correct geparsed worden |  |  |
| 8 | Simulator zend terug | handmatig | Simulator zend json en header terug naar de controller | Zend terug | pass |
| 9\* | Controller ontvangt | handmatig | Controller ontvangt json en header van de simulator terug |  |  |
| 10 | Simulator lichten wisselen | handmatig | De verkeerslichten wisselen in de simulator naar verloop van tijd | Gezien dat het wisselt | pass |
| 11 | Simulator geen collisie | handmatig | Er kruist geen verkeer over elkaar heen | Geen collisie gezien | pass |

Tabel 8 Testcase en resultaten eerste eindgebruikerstest

### Resultaten

De resultaten van de test met de andere groepen waren als volgt:

Groep 1

Groep 3 ging goed.

Groep 4

Groep 5

Groep 6

Groep 7

Groep 8

Groep 13

Groep 16

# Bronverwijzingen

Test case. Wikipedia 1 oktober 2020. Geraadpleegd op 23-10-2020 via: <https://en.wikipedia.org/wiki/Test_case>