jb systems
industrial automation

DE HAAGSE HOGESCHOOL
FACULTEIT IT & DESIGN

AFSTUDEERVERSLAG

22-12-2016

Communicatie-architectuur PLC's en Unity

Jordy Scholte
Haagse Hogeschool Delft

Versiebeheer

Versie	Datum	Wijzigingen	Reviewer(s)
0.1	14-09-2016	Eerste versie opgezet	M. Vaandrager
0.2	14-11-2016	Conceptversie	W.F.C. Wieringa
0.3	30-11-2016	Opmerkingen 1e bespreking verwerkt	J. van Peski W.F.C. Wieringa
1.0	22-12-2016	Opmerkingen TTA verwerkt; Laatste Versie	

Referaat

Jordy Scholte, “Communicatie tussen PLC’s en Unity3D simulaties”. Afstudeerverslag opleiding Technische Informatica, Haagse Hogeschool te Delft, 2016.

In dit verslag wordt de afstudeeropdracht beschreven die is uitgevoerd bij het afronden van de opleiding Technische Informatica aan de Haagse Hogeschool te Delft. Deze opdracht is uitgevoerd voor JB Systems te Vlaardingen en besloeg de periode van 29 Augustus tot 22 December 2016.

Naast de gewoonlijke werkzaamheden ontwikkelt JB Systems 3D simulaties van industriële installaties, voor demonstraties, troubleshooten, testen en operatortraining. Voor diverse doeleinden moeten deze simulaties gegevens van fysieke PLC’s kunnen verkrijgen, zoals in- en outputs, datablokken en andere geheugengebieden. De huidige manieren om dit te bewerkstelligen voldoen niet volledig aan de gestelde wensen en eisen. Deze afstudeeropdracht richt zich op het verbeteren van deze communicatiearchitectuur.

Descriptoren

- PLC
- Unity3D
- Scrum
- C#
- Simulatie
- Virtual Reality
- Soft-PLC

Voorwoord

Voor U ligt het procesverslag van de afstudeeropdracht die ik heb uitgevoerd ter afsluiting van mijn studie Technische Informatica aan de Haagse Hogeschool te Delft. Deze opdracht heb ik uitgevoerd bij JB Systems te Vlaardingen, in de periode tussen 29 Augustus en 23 December 2016. Met dit document wil ik aantonen dat ik beschik over HBO werk- en denkniveau, door te laten zien hoe dit afstudeerproces is verlopen en welke keuzes daarin zijn gemaakt.

In dit verslag is er van uit gegaan dat de lezer enige kennis heeft van industriële automatisering en softwareontwikkeling. Hierbij wordt gedacht aan hardware zoals PLC's, enige kennis van programmeertalen, besturingssystemen, netwerken en ontwerptechnieken. Ter ondersteuning zijn er aan het einde van dit document een verklarende woordenlijst en een lijst met afkortingen te vinden.

Tot slot wil ik van de gelegenheid gebruik maken om mijn collega's bij JB Systems te bedanken. Allereerst Maarten Vaandrager en Raoul Molenkamp voor het aanbieden van deze opdracht en de uitstekende begeleiding tijdens het project. Daarnaast wil ik uiteraard de heren Wieringa en van Peski bedanken voor de begeleiding vanuit de opleiding, en de kritiek en adviezen voor het schrijven van dit verslag. Ook gaat mijn dank uit naar de vele collega's en medestudenten bij JB die mijn vragen hebben beantwoord en (delen van) dit verslag hebben nagelopen. Tot slot wil ik de afdeling Hardware bedanken voor de gezellige tijd en het vele lachen.

Jordy Scholte,

Schiedam, 22 december '16

Samenvatting

Voor het afronden van de studie Technische Informatica aan de Haagse Hogeschool te Delft is het noodzakelijk om een afstudeeropdracht uit te voeren. In de loop van zeventien weken is deze opdracht uitgevoerd bij het bedrijf JB Systems te Vlaardingen en Delft door de afstudeerder Jordy Scholte. Dit document beschrijft het verloop van deze opdracht en hoe de resultaten tot stand zijn gekomen.

JB Systems is ontwikkelaar van toepassingen binnen machinebouw, procesautomatisering en offshore. Voor bepaalde projecten wordt tevens een 3D simulatie ontwikkeld met Unity3D voor diverse doeleinden, zoals testen, valideren, operatortraining, mockups of hardware-in-loop (HIL) testen.

In deze afstudeeropdracht is onderzoek gedaan naar betere mogelijkheden om deze simulaties te laten communiceren met fysieke PLC's. De huidige gebruikte manieren zijn niet goed te integreren in Unity3D, te duur of hebben te veel tijd nodig voor configuratie.

Voor aanvang van de afstudeerperiode is deze opdracht omschreven in het afstudeerplan [Bijlage A]. Hierna bleek dat er bij JB Systems nog een derde communicatiesysteem werd gebruikt. Echter was de manier waarop ook deze oplossing was geïmplementeerd niet efficiënt genoeg en gevoelig voor fouten.

Er is gekozen om tijdens dit afstudeerproject gebruik te maken van ontwikkelmethode Scrum. Hierdoor wordt de opdracht verdeeld in sprints. In totaal zijn er 9 sprints geweest. In de eerste sprint van de opdracht - sprint 0 - is de oriëntatiefase uitgevoerd. Hierin is de opdracht verhelderd, zijn de risico's geanalyseerd en is de planning opgesteld. Dit heeft geresulteerd in een plan van aanpak [Bijlage B] waarin deze zaken staan beschreven.

Sprint 1 omvatte het analyseren van de huidige situatie. In Sprint 2 werden verbeterpunten - en aan de hand daarvan systeemeisen - in kaart gebracht. Deze twee sprints hebben als gezamenlijk product een definitiestudie [Bijlage C] opgeleverd.

Sprint 3, 4 en 5 bevatten respectievelijk het onderzoek naar mogelijkheden, ontwerpen waarin deze mogelijkheden worden gebruikt en de voorbereiding op de daadwerkelijke implementatie. Deze drie sprints zijn gedefinieerd als de ontwerpfase en hebben geleid tot het ontwerprapport [Bijlage D].

De laatste twee sprints omvatten het testtraject dat aan het einde van het project is uitgevoerd. Sprint 6 bevat hiertoe een uitleg van hoe met de tests de eisen kunnen worden bewezen, en hoe deze uitgevoerd zullen gaan worden. Sprint 7 behandelt het daadwerkelijke testen een de resultaten. Hieruit zijn ASComm.NET, Sharp7 en WinMOD-Y200 in naar voren gekomen als beste opties, afhankelijk van de situatie.

Inhoud

1 Inleiding

In dit afstudeerverslag wordt het afstudeertraject beschreven dat is uitgevoerd ter afsluiting van de opleiding Technische Informatica. Dit afstudeerproject is over een periode van zeventien weken uitgevoerd bij JB Systems. In dit verslag is te lezen hoe dit project is verlopen, welke keuzes er zijn gemaakt en wordt er toelichting gegeven op genomen beslissingen.

Bij JB Systems worden er regelmatig 3D-visualisaties en –simulaties ontwikkeld van bijvoorbeeld productielijnen, schepen en machines. Vanwege het industriële karakter staan deze simulaties vaak in verbinding met PLC's. Voor aanvang van de afstudeeropdracht werden hiervoor communicatieoplossingen gebruikt welke niet tot volledige tevredenheid werkten; Deze waren oftewel te duur, te traag of te complex. Meer details hierover zijn terug te vinden vanaf pagina 15.

In de loop van het afstudeerproject is er onderzoek gedaan naar alternatieve oplossingen om de communicatie tussen 3D-simulaties en fysieke PLC's beter te verzorgen. Hiertoe is allereerst een analyse uitgevoerd naar de huidige situatie, zodat duidelijk kon worden vastgesteld welke problemen precies moesten worden opgelost. Aan de hand van de hieruit voortgekomen eisen zijn een aantal oplossingen gevonden en geïmplementeerd, waarmee JB Systems in het vervolg in staat is een betere communicatieoplossing te kiezen. Dit traject wordt beschreven in de sprint-hoofdstukken.

Het laatste gedeelte van dit verslag heeft een terugblikkende functie. Hierin wordt een conclusie getrokken, worden aanbevelingen gedaan en wordt het afstudeerproject geëvalueerd. Deze evaluatie behandelt het verloop van het project, de opgeleverde producten, en de aan het begin van het project gekozen beroepscompetenties.

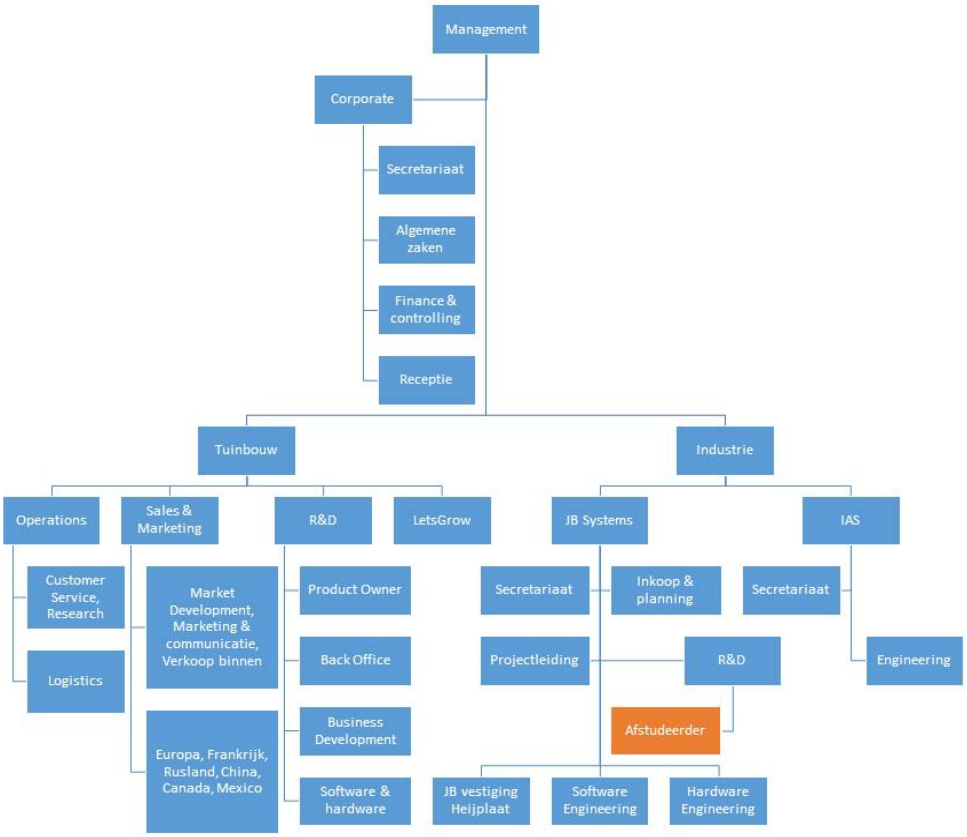
Tot slot zijn er in het separaat bijgeleverde bijlagenboek de documenten te vinden die tijdens dit project zijn geschreven. Hiernaar wordt vanuit dit verslag verwezen.

2 JB Systems

JB Systems verzorgt industriële automatisering voor klanten in de machinebouw, offshore en procesindustrie. Het is onderdeel van Batenburg techniek en bestaat op dit moment uit ongeveer 70 engineers. De afstudeeropdracht is uitgevoerd onder de afdeling R&D van de vestiging Vlaardingen. Daarnaast is er een aantal dagen gewerkt op het nieuwe kantoor in Delft.

Binnen JB Systems is een grote variëteit aan vakkennis aanwezig. Hieronder vallen softwareontwikkeling, electrical engineering, paneelbouw, installatie en alle combinaties hiervan. Hierdoor kunnen ze een groot aantal verschillende projecten aannemen binnen diverse vakgebieden, van machinebouw tot het opleveren van complete SCADA systemen. Het merendeel van de projecten behelst het opleveren van turn-key producten en lijnen aan OEM's en productie-industrie. Dit houdt in dat zowel de bouw van hardware als alle softwarecomponenten worden gerealiseerd door JB Systems, en de opdrachtgever een kant-en-klaar product opgeleverd krijgt.

Figuur 1 toont de structuur van de Hoogendoorn Groep, waar JB Systems onderdeel van is. De plek van de afstudeerder is aangegeven in oranje. JB Systems opereert, net als de andere business units, als zelfstandig bedrijf. Deze units delen een aantal bedrijfsondersteunende afdelingen, zoals finance, met elkaar om kosten en overhead te beperken.



Figuur 1: Organogram Hoogendoorn Groep

Niet getoond in de afbeelding is Batenburg N.V. Dit is het moederbedrijf van Hoogendoorn en omvat nog veel meer business units, welke op eenzelfde manier opereren als de bedrijven binnen Hoogendoorn.

3 Opdrachtomschrijving

In dit hoofdstuk wordt beschreven hoe de afstudeeropdracht tot stand is gekomen. Hiervoor wordt gekeken naar de aanleiding, probleemstelling en doelstelling. Hiermee wordt duidelijk hoe de huidige situatie er uit ziet en wat er wordt verwacht van de situatie na het voltooien van de afstudeeropdracht.

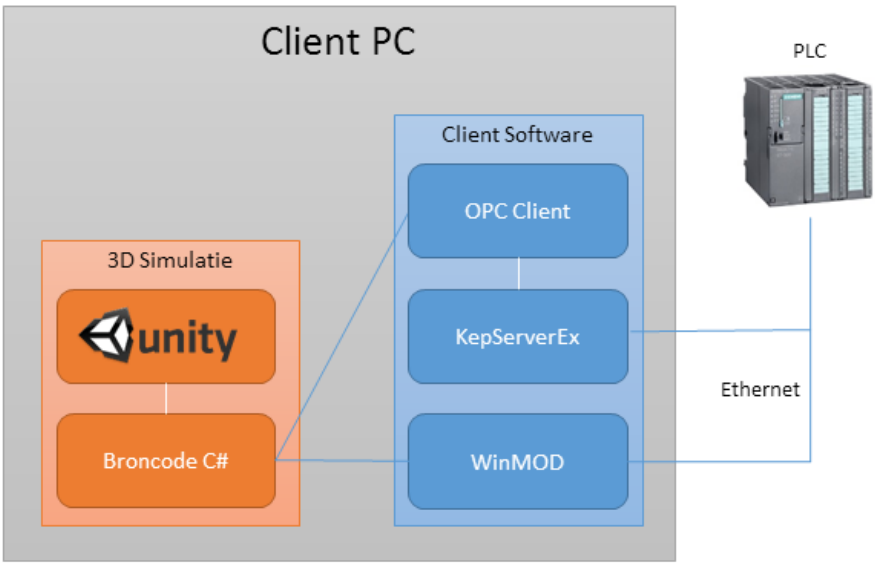
3.1 Aanleiding en huidige situatie

JB Systems maakt voor sommige producten een 3D-simulatie voor de klant. Deze kunnen worden gebruikt voor demonstraties, troubleshooten, testen en operatortraining. Deze simulaties worden ook gekoppeld aan fysieke PLC's. Op dit moment wordt daarbij gebruik gemaakt van OPC en WinMOD. Voor OPC is het gebruik van een OPC-server noodzakelijk. Hiervoor is de keuze gevallen op KepServerEx. Bij aanvang van de opdracht bleek dat er nog een derde manier in gebruik is: Een C# library genaamd S7.NET+. Deze lijkt volgens de developers goed te werken, maar is alleen te gebruiken met Siemens S7 PLC's.

Deze softwarepakketten draaien samen met de simulatie op een PC. Zij vormen als het ware een communicatie-laag tussen de simulatiesoftware en de fysieke PLC's. De simulatiesoftware zelf is geschreven in Microsoft Visual Studio en gebruikt de Unity-engine om de simulatie te renderen. De hierbij gebruikte programmeertaal is C#, welke de standaard is binnen het gebruikte .NET framework.

Er wordt voor het grootste deel gebruik gemaakt van Siemens PLC's. Echter komt het ook voor dat er andere merken hardware worden gebruikt.

Een schematische weergave van alle beschreven componenten is te zien in Figuur 2. Hierin is een onderverdeling gemaakt tussen de simulatie ("3D Simulatie") en de losse softwarepakketten ("Client Software").



Figuur 2: Schematische weergave huidige situatie

3.2 Probleemstelling

De huidige opzet blijkt foutgevoelig en instabiel; De communicatie tussen de software en de PLC valt soms uit, of verstuurde waarden komen niet aan bij het ontvangende component. Daarnaast is de snelheid van communicatie ook niet altijd stabiel. Het vermoeden is dat dit wordt veroorzaakt door de OPC- en WinMOD-pakketten, omdat deze een extra laag creëren en nooit zijn ontworpen voor dit doel.

3.3 Doelstelling

Het doel van de afstudeeropdracht is een architectuur waarmee de simulaties kunnen communiceren met de PLC's, waarin de problemen met de huidige situatie niet voorkomen. In een ideale situatie zou deze architectuur werken met alle modellen PLC's en geïntegreerd zijn in de broncode van de simulatie, met minimaal tot geen gebruik van externe softwarepakketten zoals nu wel het geval is.

3.4 Projectgrenzen

Tijdens het project mag er van een aantal zaken uit worden gegaan. Ten eerste bevinden de simulatie-PC, de PLC en eventuele overige hardware zich in dezelfde fysieke ruimte. De apparatuur wordt met elkaar verbonden door middel van ethernet of Industrial Ethernet, en de gekozen communicatiemethode zal dus via deze link moeten kunnen werken.

De Unity simulaties hoeven niet door de afstudeerder te worden ontwikkeld. Er is echter wel kennis benodigd over de werking hiervan om de communicatiearchitectuur correct te kunnen implementeren.

3.5 Onderzoeksvraag

De huidige manieren om de simulatie met PLC's te laten communiceren is te duur, instabiel of omslachtig te implementeren. Er is dus de vraag naar een oplossing welke geen last heeft van deze problemen. De onderzoeksvraag die hieruit voorkomt is dus als volgt geformuleerd:

"Welke snelle en stabiele manieren van communicatie zijn mogelijk tussen Unity3D en een PLC?"

Om deze vraag te kunnen beantwoorden moeten er eerst een aantal deelvragen worden beantwoord:

1. Welke alternatieven zijn er beschikbaar?
2. Wanneer is een oplossing snel en stabiel?

Deze vragen zullen in de loop van het onderzoek worden beantwoord.

3.6 Opdrachtomschrijving

Bovenstaande in acht nemend kan de opdracht als volgt worden omschreven: Er moet worden uitgezocht welke manieren beschikbaar zijn om een Unity simulatie te laten communiceren met een fysieke PLC. Dit moet indien mogelijk kunnen gebeuren met elk gangbaar merk en type PLC. Daarbij moet de snelheid en stabiliteit van deze manieren kwantificeerbaar worden vastgelegd, zodat een vergelijking kan worden gemaakt met de huidige oplossingen. Hiervoor is het tevens nodig om te bepalen wanneer een oplossing beter is dan de huidige. Hieruit zal een vergelijkende lijst moeten komen waaraan de opdrachtgever kan refereren bij het ontwikkelen van nieuwe simulaties.

Het is mogelijk dat er meerdere opties als beste voor een bepaalde situatie worden aangemerkt. Ook is het mogelijk, doch onwaarschijnlijk, dat er geen betere opties beschikbaar zijn.

3.7 Op te leveren producten

In de loop en aan het einde van het afstudeerproject worden er diverse producten en rapporten opgeleverd aan de opdrachtgever en de examinatoren. Deze producten zijn vastgesteld voor aanvang van het afstudeerproject in overleg met de opdrachtgever en de afstudeerbegeleider.

1. Plan van aanpak
2. Een definitiestudie/eisendocument
3. Een ontwerprapport

4. Eén of meerdere werkende proof-of-concepts
5. Testresultaten en –rapportage
6. Een implementatiedocument/-handleiding
7. Een procesverslag van de volledige afstudeeropdracht

3.8 Rollen en organisatie

Bij dit afstudeerproject zijn diverse mensen betrokken. De opdracht wordt uitgevoerd door afstudeerder Jordy Scholte. Vanuit de opleiding TI zijn hiervoor twee examinatoren aangewezen: De heer W.F.C. Wieringa is de eerste begeleider en –examinator, en de heer J. van Peski is de tweede examinator. De contactmomenten met de opleiding zijn terug te vinden in planning in hoofdstuk 5.3.

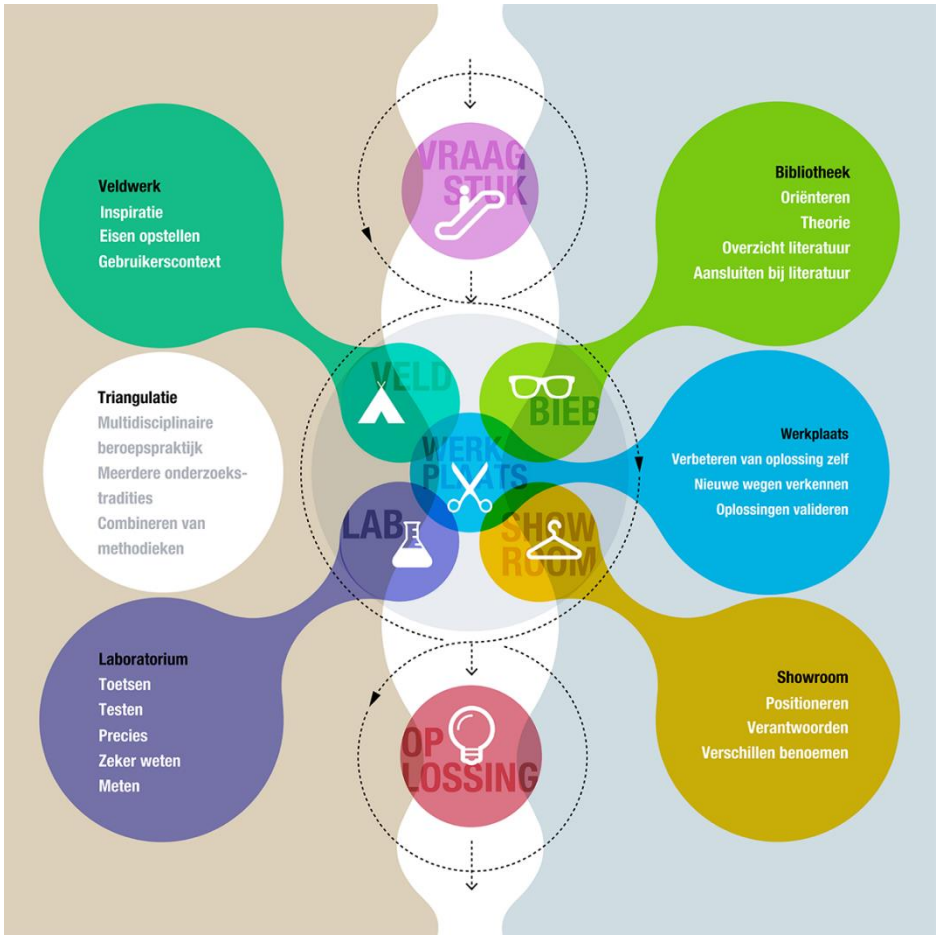
Bij JB Systems is de heer Maarten Vaandrager de opdrachtgever en begeleider van de afstudeerder.

4 Achtergrond

In dit hoofdstuk wordt de onderzoeksstrategie van het project behandeld. Aangezien dit het gehele project overspant wordt deze niet onder een sprint geplaatst. Tevens wordt alle tijdens dit afstudeerproject gebruikte software uiteengezet.

4.1 Onderzoeksmethode

De onderzoeksaanpak is ingedeeld volgens de 'Methodenkaart in de beroepspraktijk van ICT en Media' [1]. Hierbij zijn de verschillende onderdelen van onderzoek opgedeeld in een aantal categorieën (Figuur 3).



Figuur 3: Methodenkaart onderzoek [1]

De categorie 'Veldwerk' is als eerste verricht: De uitkomst hiervan is het plan van aanpak [Bijlage B] en deels de definitiestudie [Bijlage C]. Het uitgevoerde literatuuronderzoek is in te delen onder de categorie 'Bibliotheek'. Ook dit is onderdeel van de definitiestudie. 'Werkplaats' en 'Laboratorium' zijn beide terug te vinden in de ontwerp- en realisatiefase, in respectievelijk het ontwerprapport [Bijlage D] en het testrapport [Bijlage E]. Een stukje 'Showroom' is in de meeste fasen wel in meer of mindere mate terug te zien, maar de aanwezigheid is het sterkst in dit procesverslag.

4.2 Gebruikte software

Onderstaande software is gebruikt bij het uitvoeren van dit project. Wanneer er verderop in dit document versienummers ontbreken betreft dit de hier genoemde software.

Astah Professional 7 [2]
Voor een aantal UML diagrammen in de documentatie is gebruik gemaakt van Astah Professional 7

Microsoft Office 2013 [3]
De tijdens dit project opgeleverde documenten en grafieken zijn gemaakt met de applicaties uit de Microsoft Office-suite.

Microsoft Windows 10 Professional en Windows 7 Professional [4]
De simulaties zijn ontwikkeld en gecompileerd op Windows 10 Professional. Resultaten kunnen afwijken wanneer nieuwere of oudere versies van Windows worden gebruikt.

Modbus Slave 6.1.3 [5]
Dit programma simuleert een Modbus slave apparaat. Het is gebruikt om Modbus drivers te testen. Het programma kan ook een PLC emuleren.

NetToPLCsim V0.9 [6]
Software om een virtuele PLC in Siemens PLCSIM een virtuele netwerkadapter toe te wijzen zodat deze bereikt kan worden door andere software of machines.

Siemens PLCSIM V13 SP1 [7]
Software van Siemens om het gedrag van software welke is gemaakt met het TIA Portal te controleren zonder het programma in te laden in een fysieke PLC

Siemens TIA Portal V13 SP1 [8]
De Siemens PLC's zijn geprogrammeerd en gemonitord met de Totally Integrated Automation (TIA) Portal.

Unity 5.4 [9]
De 3D-omgevingen die zijn gebruikt tijdens het project zijn ontwikkeld met Unity. Ook de communicatie met de PLC's wordt hier vanuit opgezet.

Visual Studio Community 2015 [10]
Deze versie van Microsoft Visual Studio wordt standaard met Unity meegeleverd om scripts mee te bewerken. Deze scripts worden ontwikkeld met C#, de 'standaard' programmeertaal van Visual Studio.

VMware Workstation 12 Pro [11]
Een hypervisor voor virtuele machines. Hiermee kunnen virtuele PC's met een eigen besturingssysteem worden aangemaakt en gedraaid worden binnen een ander OS.

WinDBG [12]
Bij het ontwikkelen van simulaties met PC-PLC communicatie is er enkele keren gebruik gemaakt van WinDBG om uit te zoeken of een probleem lag bij scripts, libraries, of Unity zelf

5 Sprint 0: Oriëntatie

Ondanks dat er in dit stadium nog geen softwareontwikkeling is verricht, wordt deze periode toch als Scrum-sprint aangemerkt omdat het toegestaan is om Scrum in te zetten voor andere zaken als softwareontwikkeling [13]. Tevens kan er een product worden aangewezen als resultaat van deze periode, namelijk het plan van aanpak [Bijlage B].

Tijdens de eerste sprint in de eerste week van het project is, naast het kennismaken met JB Systems, de tijd gebruikt om het project op te starten. In overleg met de opdrachtgever zijn de planning en spintbacklog opgesteld. Ook zijn de risico's geanalyseerd.

5.1 Ontwikkelmethode

De juiste ontwikkelmethode is belangrijk voor het succesvol afronden van het project. Hierbij is er de keuze uit een groot aantal verschillende methodes. Bij het kiezen van de ontwikkelmethode moet rekening worden gehouden met een aantal kenmerken.

1. De methode is geschikt voor projectteams van één persoon
2. Tijdens het project moeten verschillende onderzoeken worden gedaan
3. Een groot deel van het project bestaat uit softwareontwikkeling
4. De afstudeerder heeft weinig kennis van het probleemdomein
5. Het einddoel is vrij los geformuleerd
6. De gebruikelijke methode bij JB Systems is een aangepaste SCRUM

Aan de hand van deze projectkenmerken zijn een aantal eisen opgesteld om een ontwikkelmethode te kiezen. In Figuur 4 is een lijst opgesteld met een aantal populaire en/of bij de afstudeerder bekende ontwikkelmethodieken. De tabel toont in hoeverre deze methodieken voldoen aan de gestelde eisen. Hiermee is het maken van de keuze eenvoudiger en overzichtelijker.

Eis	Nr.	Waterval	RUP	OpenUP	Scrum	XP	RAD
Geschikt voor één persoon	O1	+	~	~	~	-	+
Biedt mogelijkheid tot onderzoek	O2	+	+	+	+	~	-
Is geschikt voor softwareontwikkeling	O3	+	+	+	+	+	+
Is uit te voeren in iteraties	O4	-	+	+	+	+	+
Kan overweg met een einddoel dat niet 100% vaststaat	O5	-	~	~	+	+	+
Ervaring afstudeerder met methodiek	O6	+	~	-	+	-	+
Voorkeur van de opdrachtgever	O7	-	-	-	+	-	-

\+ Methode voldoet aan deze eis
~ Methode voldoet deels aan deze eis
\- Methode voldoet niet of nauwelijks aan deze eis

Figuur 4: Selectiematrix methodieken

Met behulp van deze matrix is de keuze gevallen op SCRUM. Deze methode heeft veruit de meeste plusjes. Helaas past het niet perfect en zullen er wat concessies moeten worden gedaan.

5.1.1 Scrum

Scrum is een Agile ontwikkelmethode die werkt met sprints. Deze sprints duren 1 tot 3 weken en met elke sprint worden onderdelen van een sprint backlog afgewerkt, welke daaraan was toegevoegd tijdens de sprint planning. Deze onderdelen zijn tot stand gekomen door middel van user stories, welke

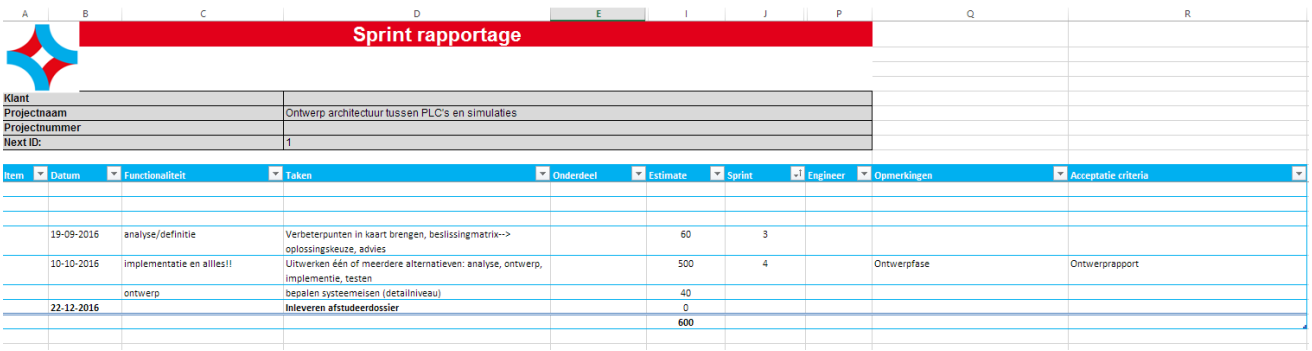
door de product-owner (opdrachtgever of klant) worden beheerd, maar in principe door iedereen kunnen worden toegevoegd.

Scrum is in principe bedoeld om gebruikt te worden door een team van meerdere personen. Met een aantal kleine aanpassingen is dit echter aangepast aan de huidige situatie waar één persoon de ontwikkeling op zich neemt. Hieronder volgen de aanpassingen welke zijn gedaan:

1. De daily scrum, waarbij een korte vergadering van maximaal 15 minuten wordt gehouden door het projectteam, wordt uitgevoerd met andere stagiairs en afstudeerders, en is niet dagelijks verplicht. Daarnaast wordt er minimaal wekelijks met de stagebegeleider een meeting gehouden
2. De wekelijkse meeting met de stagebegeleider betekent het afsluiten van een sprint of dient ter evaluatie van de voortgang van de huidige sprint. De afstudeerder bepaalt samen met de opdrachtgever de backlog en de sprint plannings. Omdat zowel de afstudeerder als de opdrachtgever kennis hebben van de materie en de enige deelnemers zijn binnen het project, kunnen deze backlog items worden beschouwd als user stories.
3. De verschillende binnen Scrum gedefinieerde rollen worden door de afstudeerder uitgevoerd. De opdrachtgever kan in dit geval worden gezien als product-owner.
4. Als product wordt er in een aantal gevallen geen software opgeleverd, maar documentatie, onderzoek of zelfs eisen. Het is echter niet ongewoon om scrum buiten puur softwareontwikkeling in te zetten. [13]
5. Scrum werkt niet met fasen, terwijl deze wel in het afstudeerplan [Bijlage A] zijn vastgesteld. Door per sprint te benoemen wat het overwegende 'thema' is (oriëntatie, definitie, ontwerp of realisatie) wordt deze originele fasering toch aangehouden.

Met deze aanpassingen hangt de methode ergens tussen Scrum en RUP, maar ten bate van de consistentie wordt er in de rest van het document gewoon gesproken over Scrum.

Om het Scrum-proces te beheren is er in het begin van het project gebruik gemaakt van de software Icescrum. Dit is een web-based applicatie waarmee de voortgang van het project, de backlogs en de sprints kunnen worden bijgehouden. Na verloop van tijd is dit echter opgegeven vanwege een aantal eigenaardigheden, vastlopers van de software en het moeilijk inpassen van éénpersoons-Scrum. Er is wel verder gegaan met het bijhouden van de backlog en voortgang in het Excel-werkblad dat bij JB Systems intern wordt gebruikt (Figuur 5).



Sprint rapportage

Klant	
Projectnaam	Ontwerp architectuur tussen PLC's en simulaties
Projectnummer	
Next ID:	1

Item	Datum	Functionaliteit	Taken	Onderdeel	Estimate	Sprint	Engineer	Opmerkingen	Acceptatie criteria
	19-09-2016	analyse/definitie	Verbeterpunten in kaart brengen, beslissingmatrix--> oplossingskeuze, advies		60	3			
	10-10-2016	implementatie en allles!!	Uitwerken één of meerdere alternatieven: analyse, ontwerp, implementie, testen		500	4		Ontwerpfase	Ontwerprapport
		ontwerp	bepalen systeemeisen (detailniveau)		40				
	22-12-2016		**Inleveren afstudeerdossier**		0				
					600				

Figuur 5: Impressie van de JB Systems backlog

5.2 Risicoanalyse

In het plan van aanpak [Bijlage B] is een risicoanalyse opgesteld. Door deze risico's van te voren te beschrijven kunnen er passende maatregelen worden getroffen om de kans van slagen van het project te doen vergroten. Deze risicoanalyse is tot stand gekomen door te kijken naar de verschillende kenmerken van het project, zoals beschreven in het hoofdstuk Ontwikkelmethode.

In de risicoanalyse zijn zaken als ziekte of verlies van werk bewust weggelaten, omdat dit zeer algemene risico's zijn die doorgaans ongewenst zijn voor de afstudeerbegeleider en –beoordelaars.

Hieronder staat één van de risico's die zijn behandeld als voorbeeld:

Risico	**Voor compatibiliteit afhankelijk van leveranciers van software**	
Gevolg	Nieuwe software werkt niet met al aanwezige, bijvoorbeeld vanwege architectuurverschillen (32- vs. 64-bit) of OS	
Kans		Middel
Kans vermindering	Van te voren testen of libraries e.d. kunnen worden meegecompileerd in kleine testprogramma's	
Impact		Groot
Impactvermindering	Alternatieve software voor handen hebben	
Plan B	Kijken of broncode kan worden bemachtigd, anders betreffende oplossing overslaan	

Zoals te zien is er een weging meegegeven aan de kans en de impact. Deze zijn ingedeeld in klein, middel en groot. De ernst van deze weging is als volgt gedefinieerd:

Klein Lichte ergernis, zeer weinig invloed op haalbaarheid van het project, eenvoudig op te lossen.

Middel Levert extra werk op wanneer niet goed afgevangen, maar er hoeft nog geen sprake te zijn van projectvertraging.

Groot Ernstige vertraging of voortijdige beëindiging van project. Al het mogelijke moet worden gedaan om deze scenario's te voorkomen

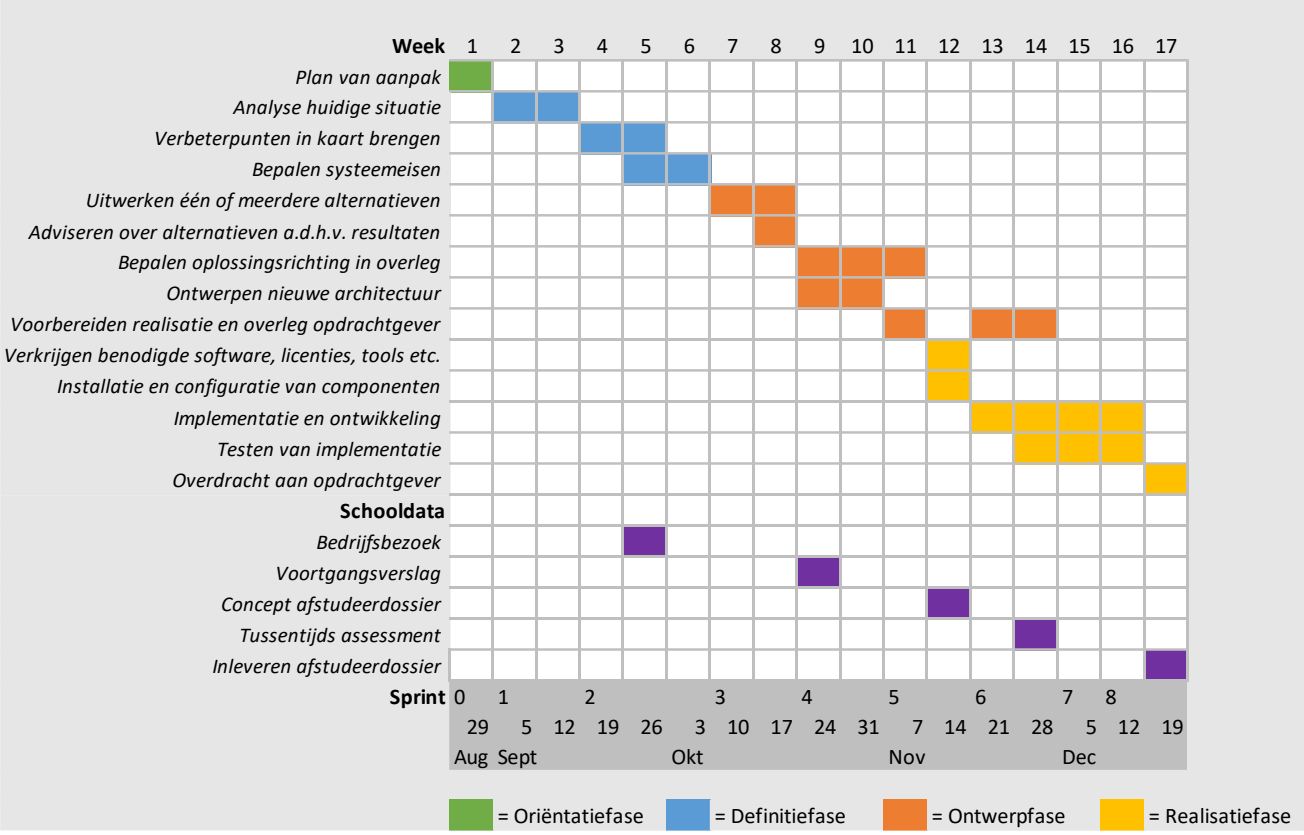
5.3 Planning

Het project is in het afstudeerplan [Bijlage A] onderverdeeld in vier fasen: De oriëntatiefase, definitiefase, ontwerpfase en realisatiefase. Omdat de op te leveren documenten ook in deze fasen zijn te verdelen worden deze fasen aangehouden in de planning, ook al wordt er gebruik gemaakt van Scrum. Er is per sprint ook aan te wijzen of deze meer nadruk legt op oriëntatie, definitie, ontwerp of realisatie. Deze fasen zijn daarom met kleurcoderingen herkenbaar gemaakt in de timeline die te zien is in Figuur 6.

De fasen zijn op hun beurt onderverdeeld in sprints. In de onderste rijen van de timeline is te zien wanneer deze sprints van start gaan. Deze duren één tot drie weken. De planning dient als globale richtlijn met wat er naar verwachting op een bepaald moment wordt behandeld. Door het iteratieve karakter van Scrum kan indien nodig na een bepaalde sprint nog tijd worden besteed aan een nieuwe iteratie van de vorige sprint.

Per week zijn één of meerdere taken ingedeeld, welke links van de timeline staan beschreven. Op sommige momenten kunnen de taken elkaar overlappen. Dit is mogelijk omdat SCRUM de ruimte biedt om aanpassingen te doen aan in vorige sprints opgeleverde producten.

Tot slot zijn belangrijke schooldata aangegeven in het paars. Deze momenten vinden plaats op één dag, maar de exacte data zijn nog niet bekend. Daarom zijn de weken gemarkeerd waarin deze data naar verwachting zullen vallen.



Figuur 6: Planning in een timeline

5.4 Teststrategie

Voor het opleveren van een stuk software moeten er tests uitgevoerd worden. Deze tests zijn reeds uitgevoerd wanneer een user story of backlog item als gereed wordt bestempeld. Omdat het gaat om een klein project worden er maar een paar verschillende testsoorten gebruikt. Hierbij kan voornamelijk worden gedacht aan functionele tests die richten op de correcte werking van een script of klasse. Daarnaast kan het meten van snelheid en stabiliteit worden gezien als niet-functionele tests op het vlak van zowel effectiviteit, betrouwbaarheid als efficiëntie.

6 Sprint 1: Analyse Huidige Situatie

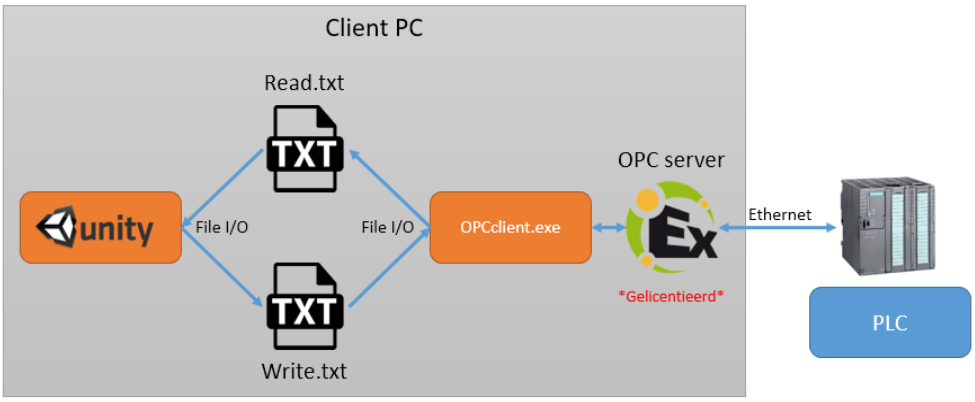
Voordat er kan worden bepaald wat de eisen voor nieuwe communicatiemogelijkheden in moeten houden moet er een definitieanalyse worden uitgevoerd. Hieronder valt een analyse van de huidige situatie en de problemen die daarbij komen kijken. Aan de hand van de hieruit voortkomende resultaten kan er gekeken worden naar wat er moet worden verbeterd en wat er wordt verwacht en geëist van nieuwe communicatiemogelijkheden.

Voorafgaand aan de afstudeeropdracht werd er gebruik gemaakt van OPC en WinMOD voor de communicatie met de PLC. Bij aanvang van de opdracht bleek ook dat vlak daarvoor, bij wijze van experiment, een keer gebruik gemaakt is van de S7.NET library. Dit hoofdstuk beschrijft de architectuur van deze oplossingen en analyseert aan de hand van technische gegevens en gesprekken met de betrokken medewerkers tegen welke problemen en limitaties aan wordt gelopen.

6.1 OPC DA

OPC is een industriële communicatiestandaard [14] waarmee onder andere PLC's kunnen worden uitgelezen en aangestuurd. Er is altijd een server nodig als centraal punt, welke uiteraard ook op de lokale client kan draaien. Alle andere apparatuur en software zijn clients. Er zijn meerdere varianten van OPC [15], maar bij JB Systems wordt er gebruik gemaakt van de 'oude' OPC DA specificatie.

Bij JB Systems wordt er gebruik gemaakt van een gratis C# Library van de OPC Foundation waarmee een client kan worden opgezet [16]. Deze library komt in de vorm van een DLL bestand. Het is ze echter niet gelukt deze direct in Unity te integreren. Dit is nogmaals geprobeerd door de afstudeerder, maar dit lijkt niet mogelijk te zijn omdat de library afhankelijk is van het .NET 4.0 framework dat niet campatibel is met Unity [17, 18]. Om toch de communicatie op te kunnen zetten is er een losstaande applicatie ontwikkeld welke door middel van txt-bestanden gegevens uitwisselt met de Unity simulatie (Figuur 7). Deze losse applicatie is vervolgens de OPC-client die gebruik maakt van de bovengenoemde DLL, en communiceert met de OPC-server, in dit geval KepServerEX 5 [19].



Figuur 7: Schematische weergave OPC-communicatie

De stappen om de OPC-oplossing te integreren zijn grofweg als volgt:

1. Kepserver installeren
2. Twee licenties regelen voor kepserver: Eén voor de server, en één voor de specifieke PLC drivers [20]
3. Toevoegen OPC Client in PLC en tags 'openbaar maken'
4. OPC server aanmaken in KepServer en tags toevoegen
5. Ontwikkelen van losstaande OPC client die leest en schrijft van tekstbestanden

6. Unity laten lezen en schrijven van tekstbestanden

Er is bij JB één werkende simulatie aanwezig welke gebruik maakt van deze configuratie met OPC. Deze is verder onder de loep genomen om te kijken wat de performance hiervan is. Deze applicatie leest 10 Reals en schrijft 4 Reals van en naar een Siemens S7-1200 PLC.

Na een analyse van de code van zowel de Unity simulatie als de losstaande Client-applicatie bleek dat de communicatiefrequentie in de code is gelimiteerd op 10Hz. Navraag hierover bij de ontwikkelaar van de applicatie leverde als antwoord op dat dit met opzet was gedaan omdat bij hogere snelheden vaak schrijfacties worden gemist door de OPC-Server.

Om dit te voorkomen wordt in de Client applicatie na elke schrijfactie gecontroleerd of deze wijziging inderdaad is doorgevoerd in de server. Door na elke schrijfactie dezelfde waarde continu terug te lezen wordt bevestigd wanneer de server de wijziging heeft ontvangen. Pas daarna wordt verder gegaan met de uitvoer van de applicatie.

Dit is getest door de afstudeerder door de verversingsfrequentie te verdubbelen naar 20Hz. Hierbij is geconstateerd dat pas na meerdere leesacties de wijziging van een waarde kan worden bevestigd. Effectief varieerde de snelheid hierdoor tussen de 12 en 14Hz. Nog hogere frequentie-instellingen leken zelfs contraproductief te werken en vanwege de vele time-outs vaak langzamer dan 10Hz te werken.

6.1.1 Voordelen

Het grootste voordeel van OPC is de wijdverbreide ondersteuning binnen de industrie, waardoor het samenwerkt met nagenoeg elk gangbaar merk en type PLC [21, 20, 22]. In de hierboven beschreven toepassing werkt het stabiel en snel genoeg.

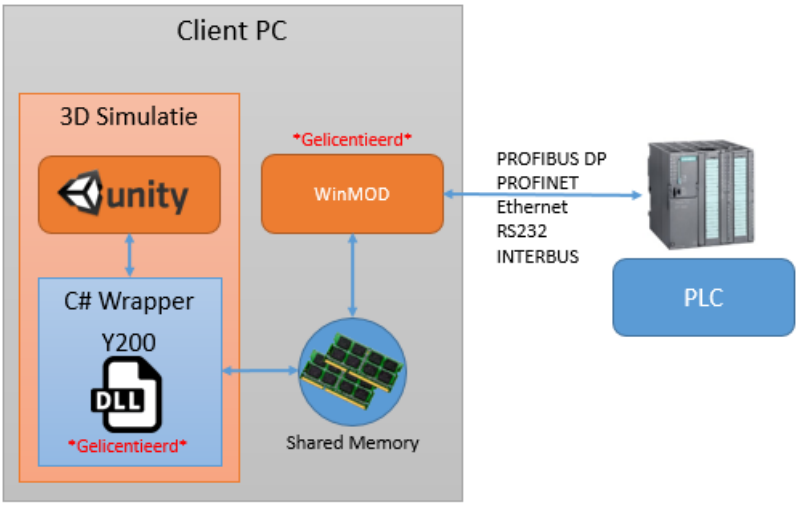
6.1.2 Nadelen

Het tien maal per seconde lezen en schrijven van 14 Reals is niet bijzonder hoog naar hedendaagse standaarden. Dit zijn, exclusief overhead, slechts 560 bytes per seconde, terwijl de fysieke verbinding tussen de hardware bestaat uit gigabit-ethernet. Dit is ook de reden dat OPC DA alleen bij bovenstaand project wordt gebruikt: Het is gewoonweg niet snel genoeg voor andere projecten.

Daarnaast moeten er voor de server voor elk afzonderlijk merk PLC licenties aangeschaft worden [20]. Deze licenties zijn runtime-licenties, en dienen per opgeleverd product te worden aangeschaft. Dit komt voor één project neer op ongeveer €2.000,- wat gezien de magere prestaties erg veel geld is.

6.2 WinMOD/Y200

WinMOD is een softwarepakket waarmee real-time simulatie, visualisatie en communicatie kan worden verzorgd van en met industriële systemen. Het kan communiceren met nagenoeg elk type PLC en met elke gangbare veldbus [23, 24]. Voor WinMOD is een losstaande C++ DLL beschikbaar waarmee externe applicaties kunnen communiceren met WinMOD door middel van een shared memory driver [25]. Dit systeem valt onder de naam Y200, en is een uitbreiding op WinMOD. Deze DLL kan met behulp van een C# wrapper worden benaderd door Unity.Tevens kan de Y200 library gebruikt worden in een aantal andere pakketten, waaronder Simulink. Hierdoor kunnen meerdere applicaties gebruik maken van hetzelfde stukje gedeeld geheugen.



Figuur 8: Schematische weergave WinMOD/Y200-communicatie

Voor het opzetten van dit systeem zijn grofweg de volgende stappen benodigd [26]:

1. Installeren WinMOD
2. WinMOD project aanmaken en juiste PLC driver laden
3. I/O mappen - Afhankelijk van het type PLC kan dit automatisch
4. Y200 instellen: Driver laden, ranges aanmaken en koppelen aan WinMOD project door operands toe te kennen
5. Licentie van WinMOD, PLC driver en Y200 in orde maken: Licentie updaten op dongle en licentiefile toevoegen aan WinMod
6. Y200 DLL toevoegen aan Unity en code schrijven om gedeeld geheugen aan te spreken

De Y200-WinMOD koppeling wordt op het moment gebruikt in een project waarbij 96 Reals van de PLC worden uitgelezen door Unity. Deze I/O list wordt 60 keer per seconde opgehaald. Dit is tot op heden het project met de grootste omvang. De snelheid en bandbreedte zijn ruim voldoende. Deze cijfers worden daarom als maatstaf voor andere oplossingen aangehouden.

6.2.1 Voordelen

Over de werking van WinMOD is men erg tevreden; De software werkt erg stabiel en snel, en biedt mogelijkheid om op een event-driven basis te communiceren, hoewel dit nog niet wordt gebruikt. Daarnaast werkt het met elk type PLC op bijna elke industriële veldbus [24, 27].

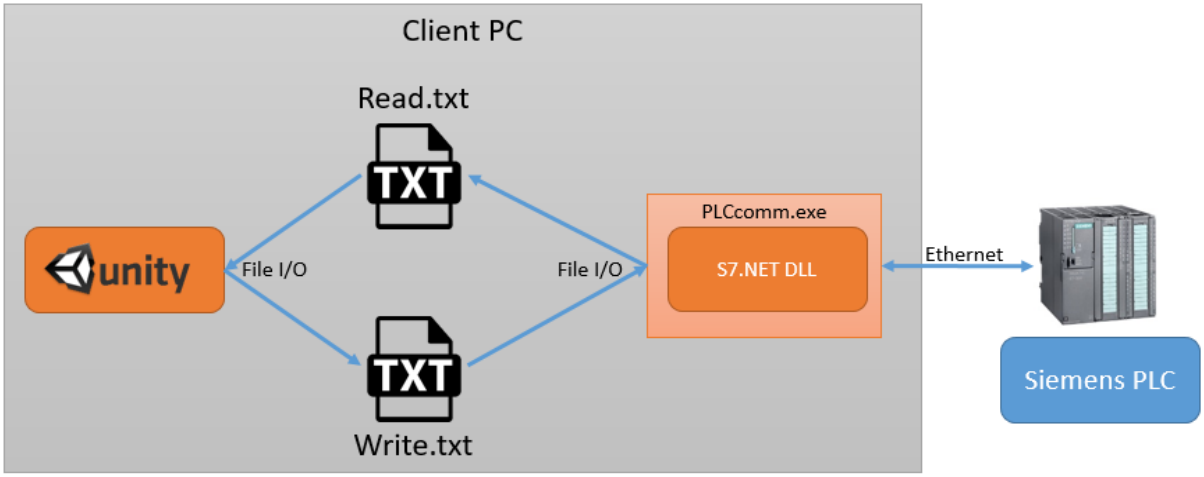
6.2.2 Nadelen

De prijs van WinMOD is hekel punt: Een licentie loopt al gauw richting de €14.000,- wat voor kleinere projecten vaak niet is te verantwoorden.

6.3 S7.NET

S7.NET is een gratis verkrijgbare opensource library [28] waarmee het Siemens S7 protocol [29, 30] kan worden geïntegreerd in C# applicaties. Dit is het protocol wat de S7-PLC's van Siemens gebruiken voor onderlinge communicatie, connectie met HMI's en SCADA systemen, en de verbinding met software als het TIA Portal. Hierdoor is er met het S7 protocol volledige toegang te verkrijgen tot de geheugengebieden van de CPU. Voor de ontwikkeling van deze en andere libraries is het Siemens S7 protocol ge-reverse-engineered. Siemens of andere commerciële partijen bieden dus geen ondersteuning aan voor het gebruik hiervan.

Net als bij OPC is het bij JB Systems niet gelukt om S7.NET direct te integreren in Unity. Dit komt omdat S7.NET volgens de betreffende developer bij JB Systems gebruik maakt van .NET 4.0, wat niet wordt ondersteund door Unity [17, 18]. Dat is althans het vermoeden, want ondanks dat de DLL succesvol kan worden geïmporteerd in een Unity project, crasht de omgeving zodra deze wordt afgespeeld. Deze ondersteuning staat echter wel op de roadmap [31], maar is op dit moment nog niet bruikbaar. Hier is daarom tevens gebruik gemaakt van tussenliggende txt-bestanden om Unity met een stand-alone applicatie te laten communiceren. Deze laatste applicatie bevat de S7-protocol drivers om met de PLC te kunnen communiceren (Figuur 9) .



Figuur 9: Schematische weergave S7.NET-communicatie

Voor het opzetten van deze constructie zijn de volgende stappen benodigd:

1. Ontwikkelen van losstaande S7.NET applicatie die leest en schrijft van tekstbestanden
2. Juiste PLC variabelen/adressen toevoegen in S7.NET-applicatie
3. Unity laten lezen en schrijven van tekstbestanden

6.3.1 Voordelen

Het gebruik van het S7-protocol biedt directe toegang tot alle mogelijke geheugengebieden en datablokken van een S7-PLC. Bij directe integratie in een eigen applicatie is het de communicatiemogelijkheid met het minste aantal stappen in vergelijking met OPC en WinMOD, wat de complexiteit van de software en latency in de communicatie ten goede komt.

6.3.2 Nadelen

Het grootste nadeel aan het gebruik van S7.NET is de incompatibiliteit van Unity met het .NET 4.0 framework. Hierdoor worden er door het gebruik van een tussenliggende applicatie met tekstbestanden extra lagen toegevoegd welke de complexiteit van het geheel verhogen. Dit is uiteraard niet de meest efficiënte oplossing. Daarnaast lijkt de ontwikkeling van de library gestaakt en is deze voor het laatst geüpdatet in 2009 [28]. De kans is daarom zeer groot dat deze library niet met toekomstige modellen zal werken. Uiteraard ontbreekt elke vorm van professionele support, omdat de S7.NET library een gratis, open-source project is.

Tot slot werkt het S7-protocol uitsluitend met Siemens S7 PLC's, omdat dit hun eigen proprietary protocol is. Omdat er bij JB Systems af en toe ook gebruik gemaakt wordt van andere merken, zou de ideale oplossing hier ook compatibel mee moeten zijn.

6.4 Conclusie Analyse

De huidige gebruikte communicatiemogelijkheden hebben allemaal gemeen dat ze uit meerdere tussenstappen bestaan in plaats van direct in Unity te worden geïntegreerd. Dit brengt extra complexiteit met zich mee, wat op zijn beurt de kans op bugs, fouten en problemen vergroot [32, 33, 34, 35]. Dit is ook de ervaring van de engineers die zich bezighouden met de implementatie in Unity; Vooral de OPC methode is vrij instabiel en wordt het liefst afgeschaft. Ook kost het veel tijd om te implementeren en het naderhand doorvoeren van een wijziging neemt vaak veel tijd in beslag.

De grootste winst zal te behalen zijn in het elimineren van de tekstbestanden en tussenliggende applicaties, en een manier te vinden om de communicatielibraries direct te integreren in de Unity simulatie. Bij het gebruik van een nieuwe communicatiemogelijkheid is het verstandig om van te voren te kijken of hiermee de directe integratie mogelijk is, hetzij met gebruik van een wrapper.

7 Sprint 2: Eisen

Aan de hand van de resultaten uit sprint 1 is een lijst met eisen opgesteld in overleg met de opdrachtgever. De communicatietechnieken die in de volgende sprints worden behandeld moeten hieraan voldoen. Om een weging mee te geven aan de eisen zijn deze tevens voorafgaand aan het onderzoek ingedeeld volgens de MoSCoW-methode [36]:

Must have	Het product moet aan deze eisen voldoen.
Should have	Deze eisen zijn zeer gewenst, maar het product blijft bruikbaar indien hier niet aan wordt voldaan.
Could have	Deze eisen worden niet aan voldaan wanneer er onvoldoende tijd beschikbaar is, of deze op een andere manier niet haalbaar blijken.
Won't have	Deze eisen komen in dit project niet aan bod, maar kunnen worden meegenomen in toekomstige projecten.

7.1 Opstellen van de eisen

Voor het opstellen van de eisen zijn er een vijftal vragen gesteld:

1. Met welke PLC's moet de oplossing werken?
2. Wanneer is een oplossing snel?
3. Wanneer is een oplossing stabiel?
4. Hoeveel moeite mag de integratie en configuratie kosten?
5. Heeft de opdrachtgever nog speciale wensen?

Deze vragen zijn deels bedacht aan de hand van de onderzoeksvraag uit de Opdrachtomschrijving., en deels aan de hand van de resultaten uit hoofdstuk 6. Door deze vragen te beantwoorden zullen zoveel mogelijk eisen al kunnen worden opgesteld. Ik zag twee mogelijkheden om deze vragen te beantwoorden:

- Interview met de opdrachtgever en engineers die betrokken zijn bij deze kwestie
- Een baseline vaststellen aan de hand van de huidige situatie
- Kijken naar de eisen van lopende projecten en projecten die op de planning staan.

Omdat de tweede optie reeds is uitgevoerd in Sprint 1, bleven alleen de eerste en de derde optie over.

7.1.1 Met welke PLC's moet de oplossing werken?

Het antwoord op deze vraag is volledig afhankelijk van de wensen van JB Systems. Daarom is er gesproken met de opdrachtgever en de twee engineers die bij R&D de simulaties ontwikkelen: Wilbert Blom en Mark Hopman. Ook is er gesproken met twee PLC-engineers: Tim Hartvelt en Jaap de Jong. Daarbij is de vraag gesteld: Welke merken PLC komen het meest voor tijdens projecten. Hierbij kwam Siemens duidelijk naar voren als meest genoemd merk. Daarnaast werden de merken Allen-Bradley, Beckhoff en Schneider Electric veel genoemd.

Hieruit is de eis af te leiden dat de oplossing in ieder geval met Siemens hardware zal moeten werken. De huidige lijn van siemens is de S7-serie. Het is dus een must-have eis dat de gekozen oplossing werkt met Siemens S7 PLC's.

De andere drie merken welke een aantal keer werden vermeld kunnen worden gezien als gangbaar binnen JB Systems. Het is dus wenselijk dat er oplossingen zijn die communicatie voorzien met deze merken. Wanneer er bijvoorbeeld een goede oplossing wordt gevonden welke alleen met Allen-

Bradley werkt, zal deze ook worden meegenomen in de tests en vergelijking. De eis voor gangbare merken is opgesteld als should-have eis, omdat deze merken wat minder vaak voorkwamen.

Tot slot is er de mogelijkheid opengehouden dat een oplossing met elk denkbaar merk en type PLC werkt. Omdat het praktisch onmogelijk is om te controleren of een oplossing hieraan voldoet is deze eis afgezwakt tot could-have. Als er volgens de documentatie van een communicatieoplossing met meer merken kan worden gewerkt dan de vier voorheen genoemde, wordt er van uitgegaan dat er aan deze eis wordt voldaan.

7.1.2 Wanneer is een oplossing snel?

Om te bepalen hoeveel waardes op welke snelheid moeten worden gelezen of geschreven is er gekeken naar drie projecten welke op dit moment lopen. Hiervan zijn de communicatie-eisen en I/O-lists bestudeerd en vergeleken. Het project met veruit de langste I/O list is het Bravenes project. Dit project bestaat uit de simulatie van een valpijpschip genaamd Bravenes [37]. Deze I/O-list [Bijlage F] bevat 96 Reals die door Unity moeten worden uitgelezen. Om een klein beetje marge te houden wordt dit getal in dit project afgerond op 100.

Verder is in dit project de eis gesteld dat voor een realistische simulatie deze waardes elke 10 milliseconden moeten worden uitgelezen. Verder zijn alle waardes in de I/O list bestempeld als read-only. Daarom wordt in het afstudeerproject de eis opgeschreven als “100 Reals op 100Hz in één richting”, dus lezen óf schrijven. Met 100Hz wordt dus niet bedoeld 100 keer per seconde ‘read-modify-write’.

Dit levert gelijk een maximale communicatielatency van 10ms op. Dit betreft de tijd dat het duurt voordat de data van het ene apparaat het andere heeft bereikt. Dit is ook opgesteld als must-have eis.

Tijdens een gesprek met Wilbert Blom is naar voren gekomen dat achteraf gezien toch niet alle waardes op de I/O-list gebruikt gaan worden, en zoals beschreven in 6.2 is de reële snelheid op het moment 60Hz. Hierom is de eis afgezwakt naar should-have. Desondanks blijft Bravenes toch het project met de langste I/O-list, waardoor bovenstaande analyse nog steeds correct is.

7.1.3 Wanneer is een oplossing stabiel?

De meeste simulaties zullen niet langer achtereenvolgens hoeven te draaien dan de duur van een demonstratie of operatortraining. Dit bedraagt sowieso minder dan één werkdag van acht uur. Voor simulaties als de Kraansimulator van het Maritiem Museum is het ook niet erg om deze na sluitingstijd of voor openingstijd even te herstarten. Er kan dus gesteld worden dat een simulatie zelden langer dan 24 uur aaneengesloten hoeft te draaien. Om er zeker van te zijn dat de communicatie oplossingen dit halen is deze tijd verdubbeld en als must-have eis opgesteld.

Hierbij moet wel worden opgemerkt dat de stabiliteit van de communicatie zal moeten worden gemeten, en niet die van de simulatie. Een slecht geprogrammeerde simulatie kan namelijk problemen zoals vastlopers veroorzaken zonder dat dit iets met de communicatie te maken heeft. Hier zal tijdens het testtraject dus rekening mee moeten worden gehouden.

7.1.4 Hoeveel moeite mag de integratie en configuratie kosten?

Zoals beschreven in Sprint 1: Analyse Huidige Situatie wordt er in een aantal gevallen gebruik gemaakt van een tussenapplicatie met tekstbestanden om een oplossing te implementeren welke niet compatibel is met Unity. Dit is niet wenselijk omdat er tijd moet worden gestoken in de ontwikkeling van een separate applicatie, welke ook bij elke wijziging aan de communicatie in de simulatie moet worden aangepast. Daarnaast introduceren de tekstbestanden een afhankelijkheid op de disk-I/O, wat

per definitie langzamer is dan (cache-)geheugen. Het is daarom zeer gewenst dat een alternatieve oplossing uit minder componenten bestaat, wat is opgesteld als should-have eis.

Het liefst wordt een oplossing direct in Unity geïntegreerd. Dit leverde de vraag op wanneer een library wel of niet compatibel is. Hierbij maakte Wilbert Blom de opmerking dat dat iets te maken had met .NET frameworks.

Hierop is de afstudeerder op kort onderzoek op internet uitgegaan naar wat de eisen zijn aan externe libraries wanneer deze geïntegreerd moeten worden in Unity. Hieruit is gebleken dat Unity werkt met een oude versie van het Mono framework, wat op zijn beurt een opensource implementatie van het .NET framework is. Helaas loopt deze Mono-versie zwaar achter, en kan een simulatie niet gecompileerd worden wanneer er afhankelijkheden worden gebruikt op het .NET framework hoger dan versie 3.5 [17, 18]. Daarnaast zijn niet alle .NET mogelijkheden in Mono geïmplementeerd, waardoor niet kan worden gegarandeerd dat alle code werkt [38].

Om deze reden is de eis opgesteld dat een oplossing compatibel moet zijn met het .NET framework 3.5 en geen afhankelijkheden heeft op hogere versies.

Daarnaast is vaak de PLC software al ontworpen, of zelfs ontwikkeld wanneer er nog aan de simulatie wordt gewerkt. Zelfs als dit niet het geval is, is het vanuit een veiligheids- en continuïteitoogpunt een verstandige zaak om niet teveel wijzigingen aan de PLC code door te hoeven voeren. Daarnaast is het voor de simulatie realistischer als de code op de PLC 100% identiek is aan wat er op locatie bij de fysieke hardware wordt gebruikt. Iets als een configuratie-instelling op de PLC valt hier niet onder, aangezien dit soms ook moet gebeuren bij industrie-standaarden als OPC. Als er ergens dus in de PLC een vinkje moet worden gezet om een oplossing werkend te krijgen hoeft dit geen bezwaar te zijn. Hierdoor is deze eis opgesteld als should-have eis.

7.1.5 Heeft de opdrachtgever nog speciale wensen?

Met de bovenstaande eisen is naar de opdrachtgever gestapt met de vraag of hij hier nog iets aan toe te voegen had. Het eerste wat hierbij naar voren kwam was de prijs. Gezien de prijs van een WinMOD licentie is het de eis dat een alternatieve oplossing sowieso goedkoper is. Daarnaast vertelde hij dat de latency-eis van 10ms prima is, maar dat het niet erg is wanneer een enkele transmissie er eens wat langer over doet. Zolang de gemiddelde latency maar kleiner of gelijk aan 10ms blijft. Hieruit is een eis over jitter opgesteld. Deze mag niet meer dan 50ms bedragen.

7.2 Eisenlijst

Nr.	Beschrijving	Prioriteit
1	Snelheid van communicatie = 100Hz, bij 100 Reals per datastroom, één richting	Should
2	Oplossing mag niet duurder zijn dan huidige duurste mogelijkheid (WinMOD ~14€)	Must
3	De gekozen oplossing moet werken met Siemens S7 PLC's	Must
4	De gekozen oplossing moet werken met gangbare merken en types PLC's	Should
5	De gekozen oplossing moet universeel inzetbaar zijn voor alle types PLC	Could
6	De jitter mag niet meer dan 50ms bedragen	Must
7	De latency van één datatransmissie mag niet meer dan 10ms bedragen	Must
8	De software moet langer (meer dan achtenveertig uur) zonder blijven draaien dan de huidige oplossingen zonder vast te lopen	Must
9	Nieuwe oplossingen moeten uit minder componenten (software, connecties, licenties etc.) bestaan dan de huidige, ter verlaging van de complexiteit van het geheel en de vermindering van kans op problemen.	Should

10	De software moet vanuit Unity benaderbaar zijn (te integreren in C#/.NET applicatie)	Must
11	Het is zeer gewenst om voor de het opzetten van de communicatie geen aanpassingen te hoeven doen aan de code op de PLC	Should
12	De gekozen oplossing mag geen merkbare invloed uitoefenen op de vernieuwingssnelheid (frames per second) van de simulatie	Must

Tabel 1: Eisen

Deze eisen zijn gebaseerd op de ervaringen van de opdrachtgever met de huidige communicatiepaden. In de zoektocht naar mogelijke oplossingen zullen deze eisen als leidraad worden aangehouden bij de initiële selectie.

7.3 Afbakening

Naast de eisen zijn er zowel in het afstudeerplan[Bijlage A], het plan van aanpak [Bijlage B] en de definitiestudie [Bijlage C] een aantal randvoorwaarden opgesteld.

De PLC en simulatie-PC bevinden zich in dezelfde ruimte en zijn verbonden door middel van ethernet of Industrial Ethernet. Dat houdt in dat gebruikte protocollen hierover getransporteerd moeten kunnen worden. Indien een communicatieprotocol ook over een andere interface getransporteerd kan worden is dat een pré.

Verder moeten mogelijke methoden direct, zonder omwegen te implementeren zijn in C#. Het werken met tussenstappen als tekstbestanden wordt het liefst vermeden.

Indien blijkt dat er geen communicatietechniek is welke beter aan de eisen voldoet dan de huidig gebruikte manieren, is dat voor de opdrachtgever een acceptabele conclusie. Eventueel kan in dat geval worden bekeken of er aanpassingen kunnen worden gedaan aan de huidige technieken om deze te verbeteren.

Tot slot wordt voor de simulaties de Unity3D-engine gebruikt. Het creëren van een volledig werkende of realistische simulatie is echter niet de opdracht van de afstudeerder; De resultaten van deze afstudeeropdracht moeten wel compatibel zijn met bestaande en toekomstige simulaties die worden ontwikkeld door JB Systems.

8 Sprint 3: Literatuuronderzoek

8.1 Literatuur

Het begin van deze sprint bestond uit de zoektocht naar literatuur over soortgelijke projecten. Hiervoor is allereerst gezocht op Google Scholar naar papers, scripties en verslagen, omdat deze een hoge mate van integriteit en betrouwbaarheid bezitten. Hierbij is er gezocht op combinaties van de volgende sleutelwoorden:

- Unity3D
- PLC
- OPC UA
- (Industrial) Communication
- PC PLC Communication
- S7 Protocol
- Modbus
- Siemens
- Allen Bradley

In deze zoektocht zijn een aantal bestaande onderzoeken gevonden die enige overlap hebben met deze afstudeeropdracht. In veel van deze onderzoeken is het uitlezen van I/O en registers van PLC's slechts een deelonderwerp en wordt er veelal gebruik gemaakt van OPC-varianten [39, 40, 41], Virtuele PLC's of HMI's [39, 42], of is er geen gedetailleerde beschrijving van de gebruikte technieken [43].

Binnen de academische literatuur zijn er twee verwijzingen gevonden naar nieuwe oplossingen:

1. Snap7 [41]
2. Experior [42]

Deze worden toegevoegd aan de lijst met mogelijkheden en zullen verderop in het traject verder worden onderzocht.

Vanwege gebrek aan literatuur was het nodig om verder onderzoek uit te voeren met behulp van niet-academische bronnen zoals Google, fora en blogs. Door te zoeken op dezelfde zoekwoorden als hierboven zijn een aantal nieuwe software libraries naar voren gekomen. Deze libraries zijn voor het merendeel ontwikkeld door personen of communities welke tegen dezelfde problemen zijn aangelopen als JB Systems, bij het opzetten van PLC-PC communicatie.

Het nadeel van deze onofficiële oplossingen is het gebrek aan professionele support; Een fabrikant zal geen hulp verlenen wanneer er problemen ontstaan door het benaderen van een PLC met behulp van deze libraries. Daarentegen is het community-karakter en het feit dat deze pakketten overwegend opensource zijn een pluspunt, omdat bij het vinden van problemen er actief aan de ontwikkeling kan worden meegeholpen, in tegenstelling tot slechts het passief rapporteren van bugs.

8.2 Gehanteerde eisen tijdens onderzoek

Bij de selectie van communicatieoplossingen is niet de gehele eisenlijst uit 7.2 gehanteerd, omdat aan de hand van slechts documentatie onmogelijk kan worden vastgesteld of de gevonden oplossingen aan al deze eisen voldoen. Daarentegen is er gebruik gemaakt van een verkorte lijst zonder weging, te zien in Tabel 2. Dit houdt in dat het mogelijk is dat een oplossing op de longlist terecht komt die niet of gedeeltelijk aan een bepaalde eis voldoet. Zo wordt bijvoorbeeld de mogelijkheid opengehouden voor oplossingen die met bepaalde merken PLC's werken, maar niet met Siemens. Zolang deze

oplossing aan de andere eisen voldoen kan deze alsnog op de longlist komen te staan. Dit is gedaan zodat de opdrachtgever later een betere keus kan maken voor bepaalde projecten.

Nr.	Beschrijving
2	Oplossing mag niet duurder zijn dan huidige duurste mogelijkheid (WinMOD ~14€)
3	De gekozen oplossing moet werken met Siemens S7 PLC's
4	De gekozen oplossing moet werken met gangbare merken en types PLC's
5	De gekozen oplossing moet universeel inzetbaar zijn voor alle types PLC
9	Nieuwe oplossingen moeten uit minder componenten (software, connecties, licenties etc.) bestaan dan de huidige, ter verlaging van de complexiteit van het geheel en de vermindering van kans op problemen.
10	De software moet vanuit Unity benaderbaar zijn (te integreren in C#/.NET applicatie)

Tabel 2: Eisen bij onderzoek

Alle gevonden resultaten zijn geplaatst in de lijst in het volgende hoofdstuk. De drie oplossingen die reeds in gebruik zijn, zijn tijdens dit onderzoek genegeerd wanneer deze zijn tegengekomen. Deze worden uiteraard later wel meegenomen in de uiteindelijke vergelijking.

Mogelijk voldoet geen enkele communicatieoplossing volledig aan alle eisen, maar in dat geval is er voor toekomstige projecten wel een beslissingsmatrix voorhanden waarmee snel een *next-best* keuze kan worden gemaakt.

8.3 Resultaten en gevonden software

Tijdens het literatuuronderzoek zijn een aantal mogelijke softwarepakketten en libraries gevonden waarmee PLC's kunnen worden gemanipuleerd vanaf een applicatie op een PC. Elk van deze softwarepakketten hebben hun voor- en nadelen. Aan de hand van beschikbare literatuur is een indicatie gemaakt van welke prestaties mogen worden verwacht van deze pakketten.

In Tabel 3 is de opsomming van deze communicatiemogelijkheden uiteengezet. Hierbij wordt gelijk opgemerkt wat de grootste tekortkomingen zijn en of er merkwaardigheden zijn die kunnen worden verwacht bij implementatie. Omdat er nog geen tests zijn uitgevoerd en de technische specificaties van sommige pakketten schaars is, is deze lijst samen gesteld aan de hand van eerste indrukken en bijvoorbeeld reviews van andere gebruikers.

Nr.	Naam	Opmerking
1	Snap7 [44]	Alleen Siemens PLC's
2	Sharp7 [45]	C# port van Snap7
3	S7.NET+ [46]	Alleen Siemens PLC's
4	LibNoDave [47]	Low-level library voor Siemens S7-200, 300 en 400 series. Vanwege de beschikbaarheid van abstractere libraries die met meer S7-types werken valt deze af.
5	LibOpenSRTP [48]	Library voor GE Fanuc PLC's. Dit type valt onder een could-have eis, en kan eventueel met andere, veelzijdigere libraries worden aangesproken. Valt hierdoor af.
6	DotNetSiemensPLCToolBoxLibrary [49]	Alleen Siemens S7-300/400 en S5 PLC's
7	CSP Ethernetdriver [50]	Oud Allen Bradley protocol, maar wordt voor backwards compatibility nog steeds ondersteund op nieuwere generaties.

8	AB Ethernet Protocol Driver [51]	Zelfde als 4, maar betaald en recenter geüpdatet en bevat commerciële support.
9	NModbus4 [52]	Kan op Siemens S7 PLC's alleen geïmplementeerd worden met wijzigingen aan PLC code [53]. Gebruikt als target framework .NET 4.0, en is dus niet te gebruiken met Unity. Valt hierdoor af.
10	OPC UA [54]	Server vereist, maar laatste KepServerEx versies (welke aanwezig is bij opdrachtgever) ondersteunen UA. Werkt met nagenoeg elke PLC. [21, 20, 22]
11	Experior [55]	Ondersteuning voor nagenoeg elk type PLC. Is betaald, maar bevat wel commerciële support. Wellicht te uitgebreid, omdat ook simulatie onderdeel is van het pakket.
12	ASComm.NET [56]	Ondersteuning voor nagenoeg elk type PLC. Is betaald, maar bevat wel commerciële support.

Tabel 3: Gevonden communicatiemogelijkheden

Zoals te zien valt een aantal oplossingen direct af, vanwege bijvoorbeeld incompatibiliteit met Unity. De overgebleven oplossingen in Tabel 3 zijn aan de hand van beschikbare literatuur en documentatie geanalyseerd, waarmee er een idee kan worden verkregen hoe deze geïmplementeerd kunnen worden in de simulaties. Voor de oplossingen op deze resulterende shortlist is in hoofdstuk 9 een implementatieontwerp gemaakt.

8.4 Aanvullende meetgegevens

Een aantal van deze oplossingen is door afstudeerder al in een eerder project gebruikt waarbij snelheidsmetingen van de libraries zijn uitgevoerd. Hoewel dit geen identieke situatie was, kan het interessant zijn om te refereren naar de resultaten die hieruit zijn voortgevloeid [57]. De hierbij gemeten snelheid is het resultaat van een cyclus van het versturen en ontvangen van 6 Reals tussen een LabVIEW applicatie en een Siemens S7-1200 PLC. Met deze resultaten kan alvast een schatting worden gemaakt over de prestaties van de libraries wanneer deze worden geïmplementeerd in de simulaties.

Omdat de transmissiesnelheid niet lineair schaalt met het aantal waardes valt hier niet voor te corrigeren. Wel is er gecorrigeerd voor een éénrichtingstransmissie, zoals gesteld in eis 1. Deze snelheden zijn getoond in Tabel 4.

Protocol	Aanvullende software	Hoogst gemeten snelheid
OPC	KepServerEX 5	40Hz
OPC	DataFEED server	52Hz
OPC UA	KepServerEX 5	72Hz
OPC UA	DataFEED server	84Hz
Modbus/TCP		48Hz
SNAP7		268Hz

Tabel 4: Snelheidsindicatie per protocol

9 Sprint 4: Ontwerp en oplossingsrichting

In deze sprint is voor de in het literatuuronderzoek gevonden technieken een implementatieontwerp opgesteld. Deze laten zien hoe de software kan worden geïntegreerd in Unity. Hieruit kan voor een aantal eisen worden vastgesteld of de betreffende communicatiemogelijkheid hieraan conformeert. Aan de hand van beschikbare documentatie zijn de technieken verder omschreven en uitgewerkt. Verderop in deze sprint is een overzicht te vinden waarin een schatting wordt gemaakt in hoeverre de communicatieoplossingen voldoen aan de eisen.

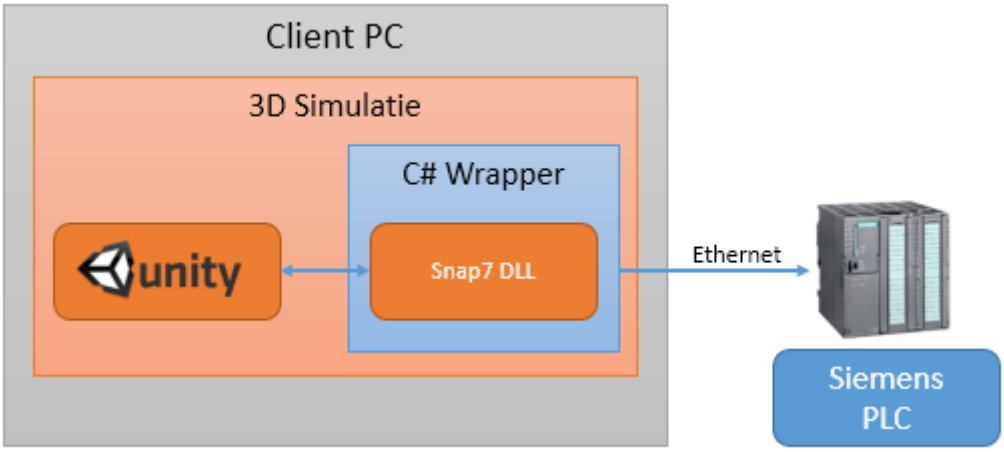
9.1 Ontwerp per gevonden oplossing

Voor de gevonden oplossingen is aan de hand van beschikbare documentatie een ontwerp gemaakt waarin wordt getoond hoe deze worden geïmplementeerd in Unity. Daarnaast wordt uitgelegd hoe de oplossingen werken, wat je ermee kan en of er tijdens het ontwerp nog bijzonderheden zijn tegengekomen.

9.1.1 Snap7

Snap7 is een library om met Siemens S7 PLC's te communiceren. Dit gebeurt door middel van Siemens' eigen S7-protocol, wat door PLC's gebruikt wordt om te communiceren met onder andere de TIA software, HMI's en andere PLC's. De library bestaat uit een DLL geschreven in C++ en wrappers om deze te kunnen implementeren in veelgebruikte programmeertalen. De library stelt zogenaamde client-, partner- en serverfuncties ter beschikking. De meest eenvoudige manier om een PC-applicatie met een PLC te laten communiceren is door middel van de client-functies. De partner- en serverfuncties zijn vooral handig om een PC te doen voorkomen als een PLC aan andere apparaten zoals HMI's. Voor dit project zullen dus voornamelijk de client-functies worden gebruikt.

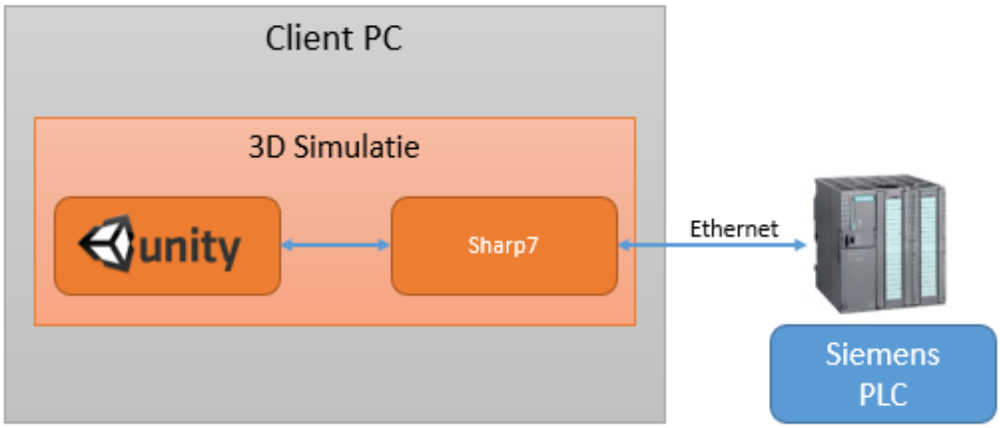
In Figuur 10 is te zien hoe de library wordt geïntegreerd in de applicatie en hoe het communicatiepad tussen simulatie en PLC verloopt.



Figuur 10: Schematische weergave Snap7-communicatie

9.1.2 Sharp7

Sharp7 is de C# port van Snap7. Deze library is uitgekomen tijdens de afstudeeropdracht, waardoor de werking en integratie van Snap7 al was uitgevoerd. De library bestaat uit een enkele C# klasse welke als los bestand bij een Visual Studio project kan worden gevoegd. De wrapper die bij Snap7 wordt gebruikt is één op één te vervangen door deze klasse: Alle functienamen en functiesyntax zijn hetzelfde. Omdat er in deze library C#-eigen functies worden gebruikt is er geen overhead door DLL's en zou het in theorie efficiënter moeten werken.

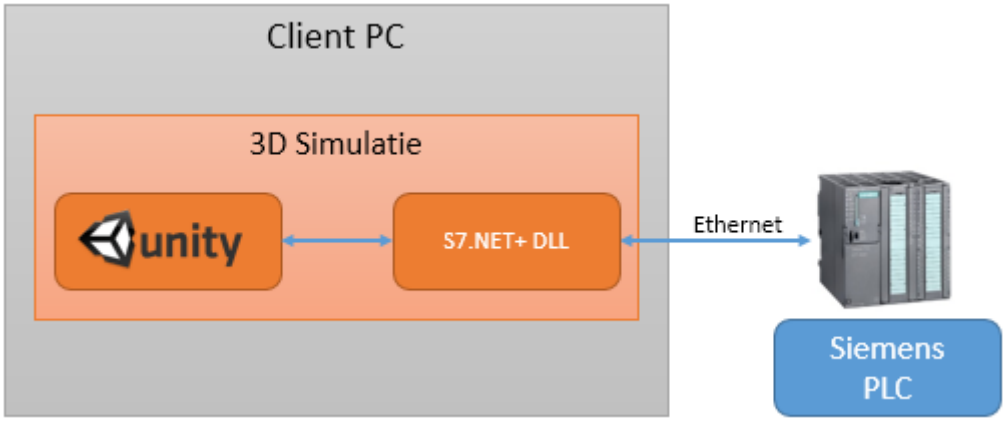


Figuur 11: Schematische weergave Sharp7

9.1.3 S7.NET+

De S7.NET+ lijkt qua functionaliteit erg op Snap7: Het maakt gebruik van het S7 protocol om te communiceren met Siemens PLC's. Het grootste verschil is dat S7.NET+ is geschreven in C# en dat de resulterende DLL rechtstreekst daarom direct, zonder wrappers, kan worden geïmplementeerd in een Visual Studio project. Figuur 12 laat zien hoe deze library wordt geïntegreerd in de simulatie.

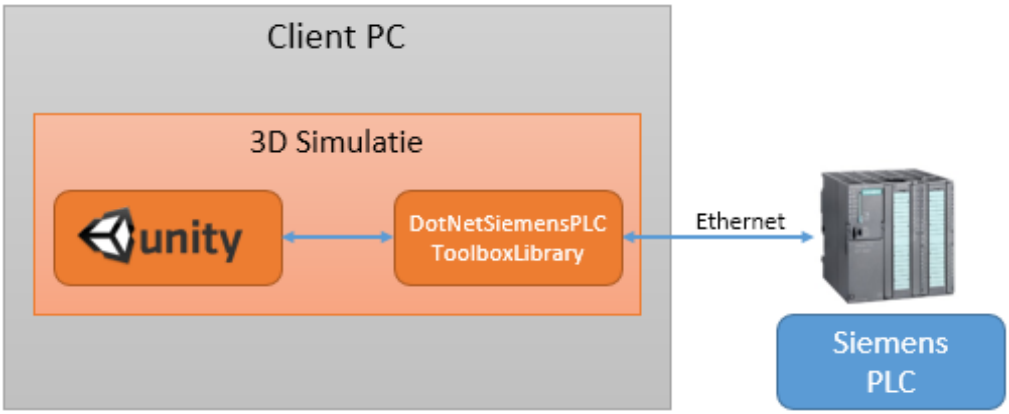
Uit de documentatie blijkt ook dat de mogelijkheden van S7.NET+ wat minder zijn dan die van Snap7, voornamelijk omdat er geen vergelijkbare partner- en serverfuncties aanwezig zijn. [58]



Figuur 12: Schematische weergave S7.NET+ communicatie

9.1.4 DotNetSiemensPLCToolBoxLibrary

De DotNetSiemensPLCToolBoxLibrary is een wat oudere C# library die ondersteuning levert voor oudere Siemens S7-300/400 en S5 PLC's. Het, volgens de documentatie, ontbreken van ondersteuning voor nieuwere S7 PLC's is een groot gemis, maar de library kan nuttig blijken wanneer er met oudere hardware gewerkt moet worden. Daarnaast is het handig dat deze library in native C# is ontwikkeld en dus zonder omwegen kan worden gebruikt in Unity projecten. De library heeft ook geen afhankelijkheden op .NET frameworks hoger dan 2.0. Deze directe integratie leidt tot de schematische weergave zoals te zien in Figuur 13.



Figuur 13: Schematische weergave DotNetSiemensPLCToolBoxLibrary

De DotNetSiemensPLCToolBoxLibrary is zoals reeds beschreven vooral nuttig bij gebruik van oudere Siemens hardware. Aangezien het gebruik van gediscontinueerde S5 PLC's niet interessant is, en er voor de oudere S7 PLC's (S7-300/400) voldoende, moderne libraries zijn, is DotNetSiemensPLCToolBoxLibrary niet geïmplementeerd en zal deze ook niet getest gaan worden.

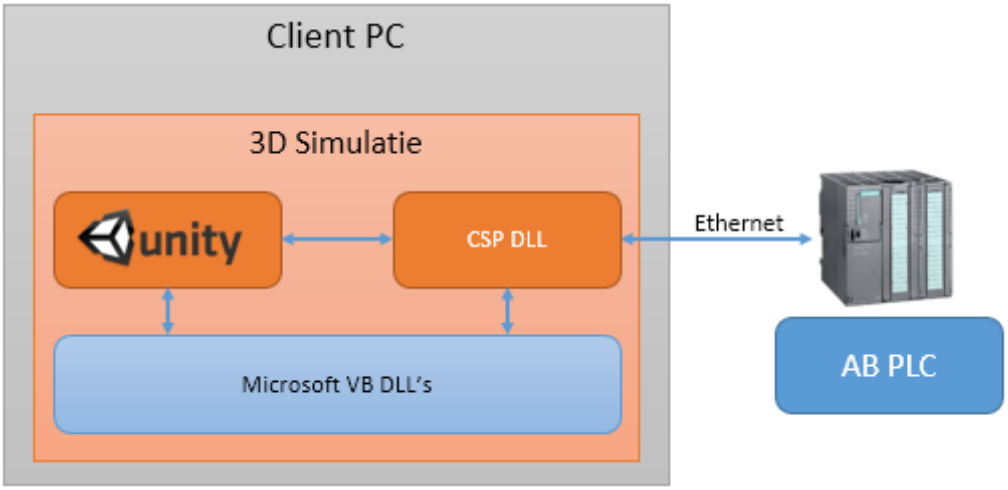
9.1.5 CSP Ethernetdriver

CSP (AB Ethernet) is een oud protocol van Allen Bradley/Rockwell. Dit wordt nog steeds gebruikt op PLC's van deze merken om backwards compatibility te waarborgen. De CSP ethernetdriver is een in VB.NET ontwikkelde library waarmee deze PLC's kunnen worden benaderd.

Door het verwijderen van alle verwijzingen naar Windows Forms uit de applicatie kan van de ABEthernet klasse een DLL worden gecompileerd. Deze kan worden geïmporteerd in Unity. Het nadeel hiervan is dat de volgende bestanden tevens moeten worden geïmporteerd in Unity om het geheel werkend te krijgen:

1. Microsoft.VisualBasic.dll
2. System.Deployment.dll

De resulterende constructie is getoond in Figuur 14.



Figuur 14: Schematische weergave CSP Ethernetdriver

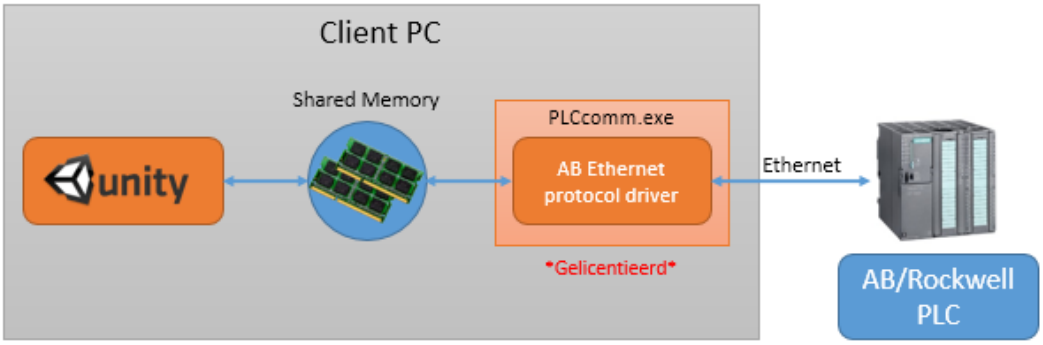
Helaas zijn deze DLL's verschillend per Windows versie. Dit zorgt voor een slechte portabiliteit van de library. Ook neemt hierdoor de implementatietijd toe ten opzichte van andere libraries. Omdat er nog

alternatieven zijn voor de communicatie met Allen-Bradley PLC's wordt deze library verder niet meer behandeld.

9.1.6 AB Ethernet Protocol Driver

De AB Ethernet Protocol Driver is een commerciële driver voor Visual Studio, gemaakt door Parijat Controlware, Inc. Deze driver kan communiceren met Allen Bradley/Rockwell SLC-5, PLC-5, ControlLogix, CompactLogix, and MicroLogix PLCs. Hiermee omvat het nagenoeg het hele Allen Bradley/Rockwell assortiment aan general purpose PLC's. De library leest volgens de documentatie een snelheid van 100 achtereenvolgende registers in 40 milliseconden.

Helaas bleek dat de library is gecompileerd voor het .NET 4.0 framework en dus niet compatibel is met Unity. Omdat de library toch interessant kan zijn wanneer communicatie met AB/Rockwell apparatuur nodig is, zou het eventueel mogelijk zijn om deze te gebruiken in combinatie met een losstaande applicatie. Met een shared memory driver (zie 9.1.9) kan deze communiceren met de Unity simulatie. Dit zou leiden tot een constructie zoals getoond in Figuur 15.



Figuur 15: Schematische weergave AB Ethernet Protocol

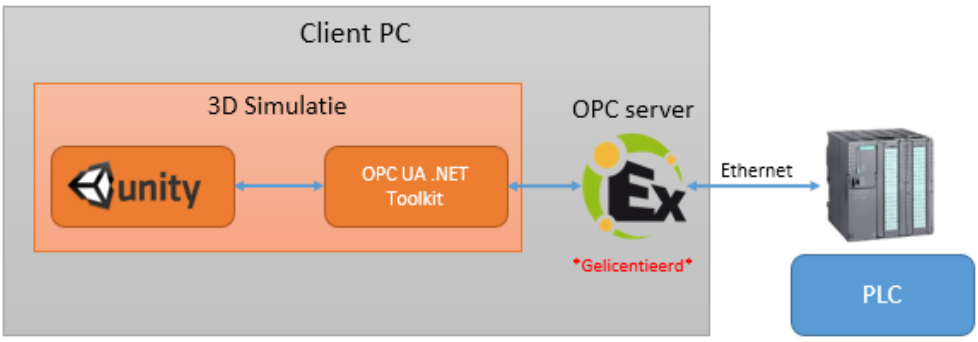
Vanwege de .NET 4.0 afhankelijkheid en omdat er voor AB/Rockwell nog andere alternatieven als CSP Ethernetdriver en ASComm.NET beschikbaar zijn, is deze library verder niet geïmplementeerd en getest. In de rest van het project wordt de AB Ethernet Protocol Driver ook niet meer behandeld.

9.1.7 OPC UA

OPC UA (Unified Architecture) is een vernieuwde variant van OPC DA. Vanwege het afstappen van DCOM en opensource karakter is het eenvoudiger om op meerdere platforms te implementeren. De OPC Foundation heeft zelf een toolkit uitgebracht waarmee OPC UA Clients kunnen worden ontwikkeld binnen de .NET omgeving. Deze werkt helaas niet met het .NET 3.5 framework, maar er zijn vier andere leveranciers van OPC toolkits die aangeven wel compatibel te zijn met .NET 3.5:

- Unified Automation .NET Based OPC UA Client SDK [59]
- Prosys OPC UA .NET SDK [60]
- Softing OPC UA .NET Server and Client Development Toolkits [61]
- Siemens SIMATIC OPC UA Toolkit [62]

Deze drie toolkits bestaan uit een aantal DLL's welke in Unity moeten worden geïmporteerd. Aangezien deze zijn ontwikkeld voor C# kunnen deze worden geïmplementeerd zoals getoond in Figuur 16.



Figuur 16: Schematiscche weergave OPC UA communicatie

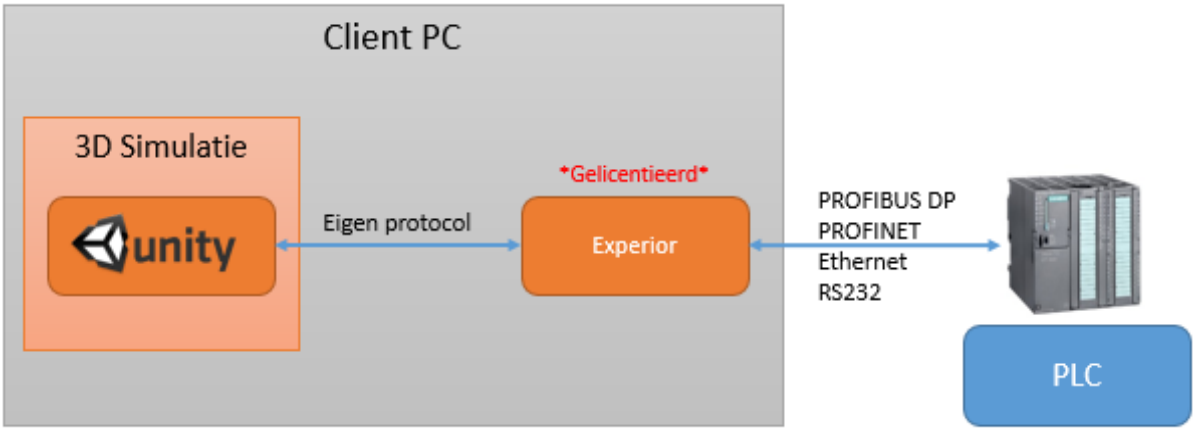
9.1.8 Experior

Om uit te kunnen vinden wat de mogelijkheden zijn van de Experior software is er contact opgenomen met de ontwikkelaar, Xcelgo. Naar hun eigen zeggen was de software geschikt om op hoge snelheid te communiceren met op zijn minst de volgende controllers:

- Siemens S7 classic en TIA
- Beckhoff
- Allen Bradley
- B&R

Helaas hadden ze geen statistieken voorhanden over doorvoersnelheden, omdat ze dit nooit zelf hebben geanalyseerd.

Na een telefonisch gesprek met de distributeur voor de Benelux is tot de conclusie gekomen dat Experior niet is waar naar wordt gezocht; De software moet als zelfstandige applicatie draaien naast Unity, in plaats van geïmporteerd te worden als library. Hierdoor zal er zelf een protocol moeten worden geschreven tussen deze twee (Figuur 17). Daarnaast zitten er in Experior zelf al zeer uitgebreide simulatie-mogelijkheden, waardoor dit pakket eigenlijk te omvangrijk is voor het doel. Tot slot kost een developer-licentie €25.000,-, wat het duurder maakt dan WinMOD en waardoor het direct afvalt.



Figuur 17: Schematische weergave Experior

Experior wordt in de rest van het project niet meer behandeld.

9.1.9 ASComm.NET

ASComm.NET is een commerciële library die wordt ontwikkeld door Automated Solutions. Deze library bevat drivers voor oude en moderne Allen-Bradley/Rockwell PLC's , Siemens S7 PLC's, General Electric PLC's en Modbus drivers. Hiermee is het mogelijk om met nagenoeg elke PLC te communiceren. De software is geschreven in C# en wordt geleverd als DLL in combinatie met een aantal applicaties om bijvoorbeeld een licentie toe te voegen of een I/O-structuur te genereren.

Omdat de library bestaat uit een C# DLL is de integratie in Unity hiervan hetzelfde als bij Sharp7 en S7.NET+. De documentatie is zeer uitgebreid, en op supportvragen werd, ondanks dat er nog een trial werd gebruikt, binnen enkele uren gereageerd. Dit bied een voordeel boven de andere oplossingen, omdat professionele ondersteuning en ontwikkeling zeker preferabel zijn.

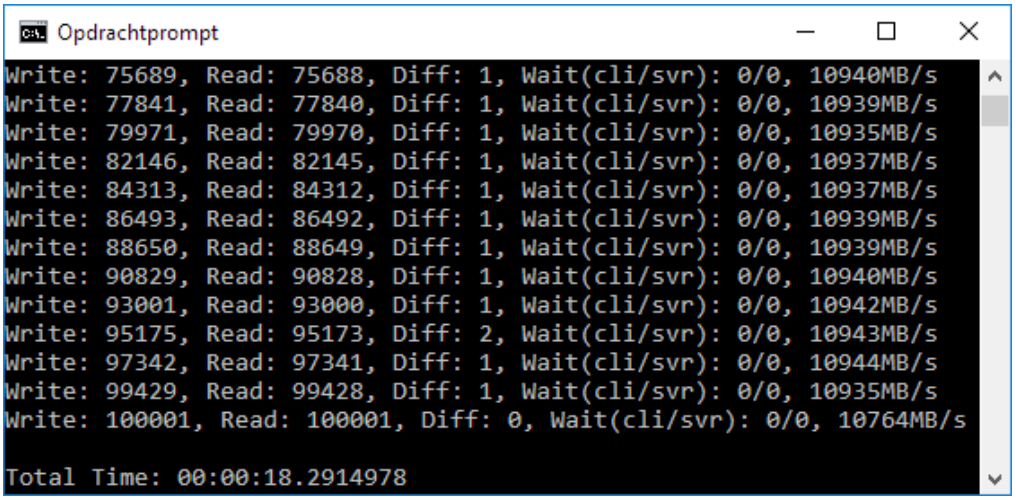
9.2 Aanvullende technieken

Tijdens het zoeken naar communicatieoplossingen zijn verder een aantal zaken aan het licht gekomen welke performanceverbeteringen kunnen opleveren bij zowel de nieuwe als de bestaande communicatieoplossingen. Deze technieken op zich verzorgen geen communicatie met PLC's, maar kunnen wel andere communicatiemogelijkheden helpen bij het voldoen aan de eisen. Later in het project zal moeten blijken of het echt nodig is deze technieken te implementeren.

9.2.1 Vervangen tekstbestanden door shared memory

Het is ook mogelijk om libraries die niet te integreren zijn in Unity te gebruiken door middel van externe applicaties. Dit is al gedaan met de OPC en S7.NET protocollen voorafgaand aan het afstudeeronderzoek. Een losse applicatie verzorgt de communicatie met de PLC. Deze moet gestart worden met of voorafgaand aan de simulatie. Hierbij werd gebruik gemaakt van tekstbestanden om tussen de simulatie en de communicatie-applicatie gegevens uit te wisselen. Dit is traag en foutgevoelig.

Het is ook mogelijk om deze tekstbestanden te vervangen door shared memory, ongeveer zoals gebeurt bij WinMOD/Y200. Shared memory is vele malen sneller dan tekstbestanden, aangezien deze laatste afhankelijk zijn van de doorvoersnelheid van de harde schijf. Shared memory haalt met gemak snelheden van meerdere gigabytes per seconde, zoals te zien in Figuur 18.

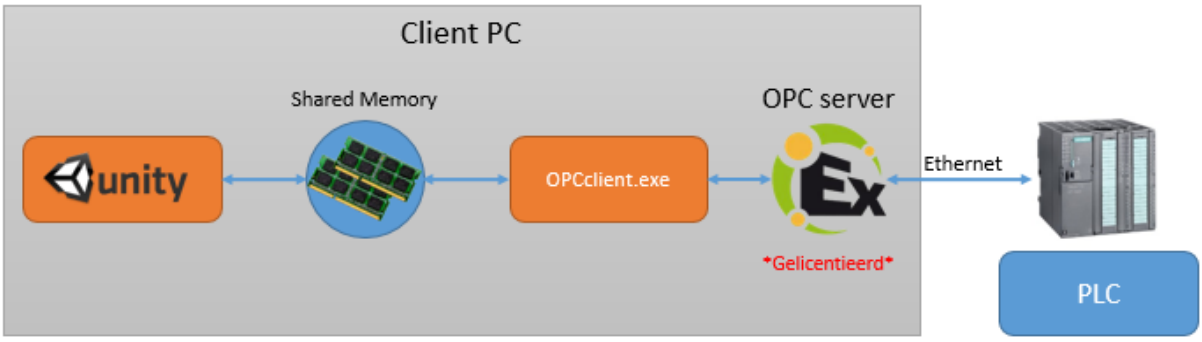


Figuur 18: Snelheidstest shared memory

Voor deze tests is gebruik gemaakt van een shared memory library die geschikt is voor .NET 3.5 [63]. Hierdoor is deze ook bruikbaar binnen Unity. Dit zijn niet de snelheden waarmee met de PLC kan

worden gecommuniceerd; Deze worden beperkt door het daadwerkelijk gebruikte protocol, zoals OPC. Door gebruik te maken van shared memory wordt de snelheid in ieder geval niet meer beperkt door de snelheid van lezen en schrijven naar tekstbestanden.

Door bijvoorbeeld de tekstbestanden uit hoofdstuk 6.1 te vervangen door shared memory zou het schematisch overzicht eruit zien als in Figuur 19. De bottleneck is hierbij naar rechts verlegd: De OPC client en server zijn nu de beperkende factor in de communicatiesnelheid.

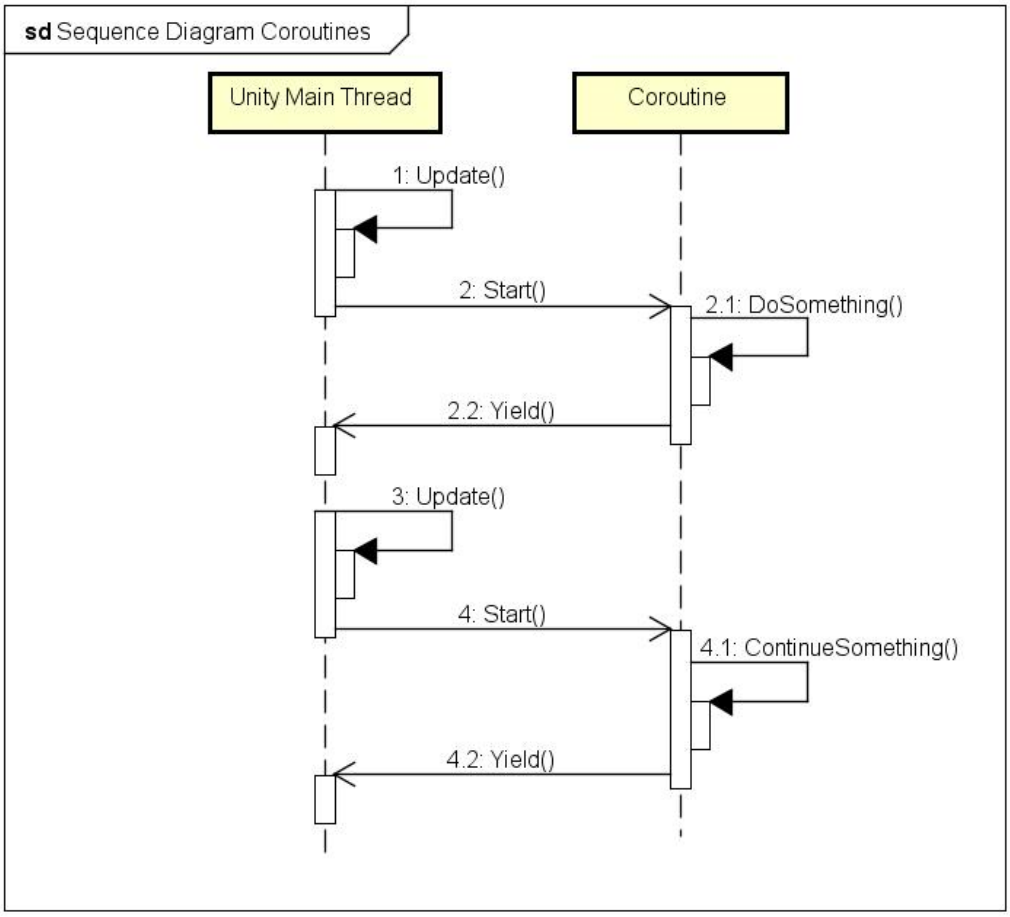


Figuur 19: Schematisch overzicht OPC met Shared memory

Op deze manier kan shared memory ook nuttig blijken bij communicatieprotocollen welke niet te implementeren zijn in C#, of in Unity3D vanwege de oude .Net versie [17].

9.2.2 Threading

Standaard zijn in Unity geen threads beschikbaar. Wel is er de beschikking over zogenoemde coroutines [64]. Hiervoor moet echter in de code van de coroutine worden aangegeven wanneer deze gepauzeerd kan worden, zodat de main thread verder kan. Coroutines worden dus niet parallel uitgevoerd, maar om-en-om, vergelijkbaar met context switching. Figuur 20 toont hoe een coroutine gedeeltelijk wordt uitgevoerd, om daarna de processortijd weer aan de main thread te geven (yielding). Nadat deze een update heeft uitgevoerd mag de coroutine weer verder met waar hij mee bezig was.



Figuur 20: Sequentiediagram coroutines in Unity

Deze manier van werken is om enkele nadelen niet bruikbaar voor de communicatie tussen Unity en PLC's. Ten eerste is het gebruik van coroutines minder efficiënt dan context switching door het besturingssysteem, omdat laatstgenoemde op een minder abstracte laag functioneert. Ten tweede vindt er geen daadwerkelijke parallellisatie plaats over meerdere processorkernen, waarvan er in elke moderne machine wel meerdere aanwezig zijn.

Het grootste nadeel is echter het feit dat het switchen in de code moet worden geprogrammeerd. Wanneer er gebruik gemaakt wordt van een third-party library waarbij de code niet kan worden aangepast, gaan coroutines dus niet werken. Zelfs als de code van een library wel kan worden aangepast, is het erg veel werk om deze uit te pluizen en op de juiste plekken yields aan te brengen.

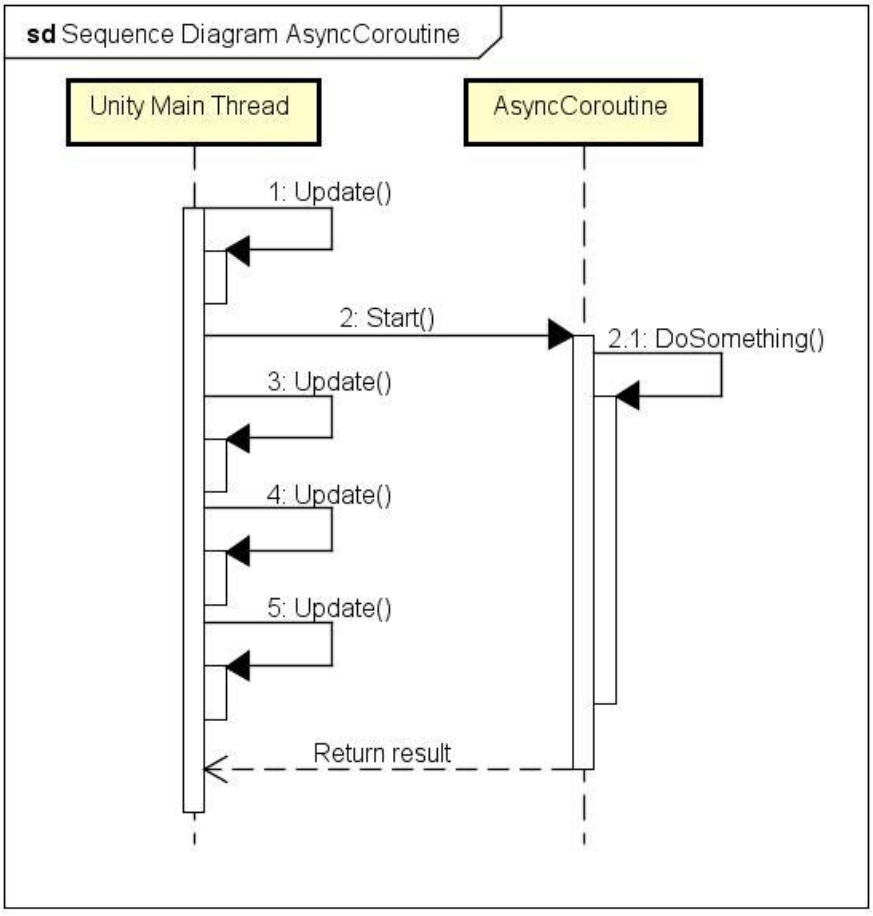
Een goed voorbeeld van een probleem dat dit oplevert is wanneer er een time-out optreedt: Netwerkcommunicatie wordt door de netwerkstack van het OS afgehandeld. Zolang een coroutine aan het wachten is op een respons van de netwerkstack kan er niet worden geyield. Hierdoor blijft de gehele simulatie hangen gedurende de time-out interval. De context switch van het OS zou dit wel goed kunnen afhandelen, omdat deze invloed heeft op zowel de simulatie als de netwerkstack.

Het is mogelijk om in Unity gebruik te maken van C# threads. Unity is echter niet thread-safe en het aanspreken van de Unity-API vanuit een thread is daarom vragen om problemen [65]. De API zal namelijk requests vanuit een andere thread weigeren, wat resulteert in exceptions of locks. Er is echter een redelijk onbekende extensie beschikbaar in de Unity Assetstore genaamd Thread Ninja [66]. Deze extensie voegt zogenoemde AsyncCoroutines toe welke daadwerkelijk parallel worden uitgevoerd.

Ook zijn er speciale functies toegevoegd waarmee de thread tijdelijk terugkeert naar de main thread om zodoende de Unity-API aan te kunnen spreken. Het gebruik van deze extensie lijkt erg op het gebruik van gewone coroutines en is daarom erg eenvoudig te implementeren.

Dit is getest door vanuit een AsyncCoroutine verbinding te maken met een niet-bestaande PLC. Onder normale omstandigheden blijft hierdoor de verbindingspoging time-outs opleveren. Het aantal FPS van de simulatie nadert vervolgens nul, omdat de frameverversing blijft wachten op de verbindingspoging. Indien deze coroutine met behulp van Thread Ninja wordt veranderd in een AsyncCoroutine blijft het aantal FPS op peil en hebben time-outs hier geen invloed meer op. Dit betekent dat communicatie en frameverversing parallel aan elkaar worden uitgevoerd.

Voor grote en/of zware berekeningen, onvoorspelbare functieaanroepen of tijdkritische zaken is Thread Ninja een uitstekende toepassing. Figuur 21 laat zien hoe een AsyncCoroutine wordt uitgevoerd parallel aan de main thread van Unity. In vergelijking met Figuur 20 is te zien dat de AsyncCoroutine niet onderbroken hoeft te worden en dat de main thread doorgaat met uitvoeren zonder te wachten op yields van de coroutine.



Figuur 21: Sequentiediagram asynchrone coroutine voorbeeld

9.3 Eerste inschatting nakoming van eisen

In onderstaande Tabel 5 staan de gevonden mogelijkheden afgezet tegen de verwachting in hoeverre deze voldoen aan de eisen uit hoofdstuk 8.2. Dit is op basis van een eerste inschatting. De drie huidige situaties zijn hierbij niet meegenomen. In de uiteindelijke vergelijkingstabel na de tests (Sprint 8) zullen deze wel te zien zijn.

	Eis:	Snap7	Sharp7	S7.NET+	CSP Ethernet driver	OPC UA	ASComm. NET
1	*Snelheid*	+	+	+	O	O	+
2	*Prijs*	+	+	+	+	+	+
3	*Siemens PLC's*	+	+	+	+	+	+
4	*Gangbare PLC's*	-	-	-	-	+	+
5	*Alle PLC's*	-	-	-	-	+	+
6	*Jitter*	O	O	O	O	O	O
7	*Latency*	O	O	O	O	O	O
8	*Stabiliteit*	O	O	O	O	O	O
9	*Complexiteit*	+	+	+	+	+	+
10	*Unity3D*	+	+	+	+	+	+
11	*Geen aanpassing PLC code*	+	+	+	O	O	O
12	*FPS*	O	O	O	O	O	O

Tabel 5: Mogelijkheden afgezet tegen eisen

De informatie om deze tabel samen te stellen komt voort uit de websites en documentatie van de betreffende libraries die staan gerefereerd in bovenstaande hoofdstukken. Voor sommige eisen, zoals 6, 7, 8 en 12 kan met deze informatie geen inschatting worden gemaakt, waardoor deze worden aangegeven als onbekend. In een latere sprint zal uit meetresultaten moeten blijken of de libraries aan deze eisen voldoen.

Eis 1, 2, 3, 4, 5, 9, 10 en 11 zijn ingevuld als schatting op basis van de informatie die bij de libraries zit, waarbij deze informatie voor waar wordt aangenomen. Bij sommige libraries zit namelijk een hoedanige hoeveelheid drivers en communicatieprotocollen dat er van uit mag worden gegaan dat deze werken met elk type PLC, tevens gezien de onmogelijke opgave om het tegendeel te bewijzen.

Enkele overige zaken die als 'onbekend' zijn ingevuld zullen in de testsprints proefondervindelijk moeten worden vastgesteld.

10 Sprint 5: Voorbereiding implementatie

In sprint 5 is zoveel mogelijk zorg gedragen voor een soepele implementatie van de uit sprint 4 overgebleven oplossingen. Hiervoor is een testomgeving opgezet en zijn alle benodigde softwareproducten verkregen. Ook worden de in deze sprint aangetroffen problemen behandeld.

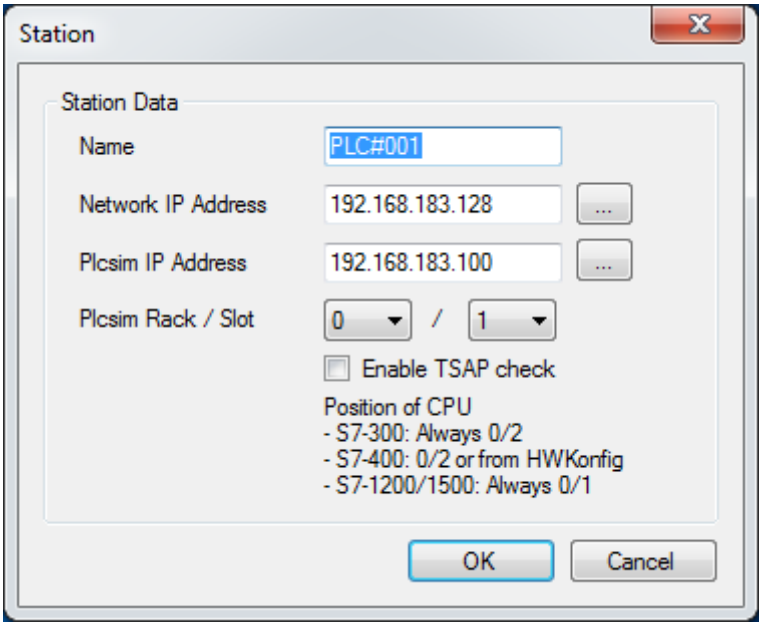
10.1 Opzetten testomgeving

Om te kunnen testen of de communicatie-architectuur werkt is het handig om kleine tests uit te voeren op de Client PC zelf, zonder externe hardware aan te koppelen en te configureren. Siemens stelt hiervoor de software 'PLCSIM' beschikbaar, zodat het gedrag van programma's die zijn geschreven in het TIA Portal kunnen worden bekeken door middel van een soft-PLC. Dit is ideaal om snel te kunnen controleren of een communicatie-architectuur werkt met Siemens apparatuur. Er hoeft in dat geval geen fysieke apparatuur te worden aangesloten en geconfigureerd. Om dit te bewerkstelligen zijn echter wel een aantal stappen nodig, omdat de PLCSIM software standaard alleen door het TIA Portal kan worden benaderd.

10.1.1 Beschikbaar stellen PLCSIM aan externe software

Volgens PLCSIM krijgt een soft-PLC een eigen IP adres, dat kan worden ingesteld bij het schrijven van het PLC programma in het TIA Portal. Het is echter niet mogelijk om buiten de Siemens software om iets te doen met dit IP adres; Zo is het niet eens mogelijk dit adres te pingen.

Op internet is het programma NetToPLCSim [6] gevonden, dat de soft-PLC koppelt aan een netwerkinterface. Het zorgt er kortgezegd voor dat de soft-PLC bereikbaar wordt via het IP adres van de machine waar deze op draait. Zo is in Figuur 22 te zien hoe het IP adres van de soft-PLC (192.168.183.100) wordt gekoppeld aan de netwerkadapter met IP adres 192.168.183.128, waardoor de soft-PLC te bereiken is via dit laatste adres.

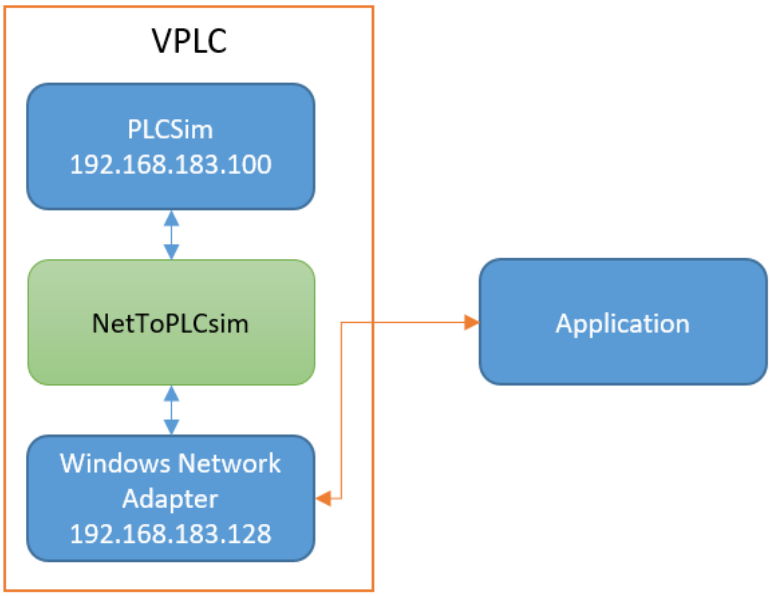


Figuur 22: Configuratie van NetToPLCSim

In Figuur 23 is schematisch weergegeven hoe de communicatie vervolgens verloopt. Een applicatie maakt verbinding met de VPLC op dezelfde manier als met een fysieke PLC en ziet hiertussen ook geen verschil. De VPLC bestaat echter uit drie belangrijke componenten:

- De netwerkadapter van de PC waar de software op draait
- het programma NetToPLCsim
- de PLCSim software van Siemens.

De applicatie stuurt alle verkeer naar het IP adres van de host PC (*192.168.183.128*). Vanaf hier wordt het door NetToPLCsim doorgestuurd naar de PLCSim software. Antwoorden worden over dezelfde weg weer teruggestuurd naar de applicatie. Deze hele route is voor de applicatie onzichtbaar, waardoor het lijkt alsof er met een echte, fysieke PLC wordt gecommuniceerd. Tevens kunnen al deze componenten zich op dezelfde PC bevinden. In dat geval kan de applicatie verbinding maken met localhost, of kan er een IP adres worden toegevoegd aan de netwerkadapter.



Figuur 23: Schematische weergave communicatiepad virtuele PLC

Aan het gebruik van een soft-PLC in combinatie met NetToPLCSim zitten wel een aantal limitaties. Zo is het alleen mogelijk om bepaalde datagebieden te bereiken [67]. In Tabel 6 is getoond welke gebieden gelezen of geschreven kunnen worden. Tijdens een korte test is gebleken dat sommige andere functies zoals het stoppen van de PLC of het uitlezen van zijn serienummer niet werken, terwijl het uitlezen van het ordernummer juist weer wel werkt.

Gebied	Lezen	Schrijven
Data Blocks (DB)	Ja	Ja
Flags (M)	Ja	Ja
Outputs (O)	Ja	Ja
Inputs (I)	Ja	Ja
Timer (T)	Ja	Ja
Counter (C)	Ja	Ja
Peripheral Inputs (PI)	---	---
Peripheral Outputs (PO)	---	---

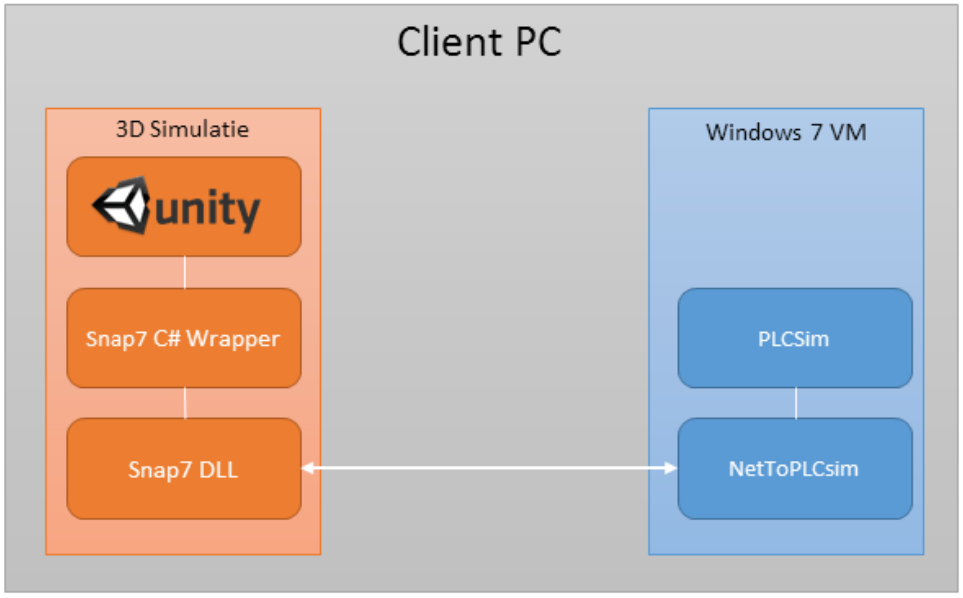
Tabel 6: Beschikbare datagebieden soft-PLC

Voor functionaliteitstesten en om een snelle check te doen of de implementatie van een communicatiearchitectuur werkt, werkt het echter naar behoren.

Het enige grote nadeel aan deze manier van werken is dat het downloaden van nieuwe software op de VPLC regelmatig crasht als NetToPLCsim is gestart. Het herstarten van alle siemens applicaties wil af en toe werken, maar de enige zekere oplossing hiervoor is het opnieuw opstarten van de computer.

10.2 Problemen met Siemens Software en Windows 10

De TIA Portal V13 software van Siemens wordt officieel niet ondersteund op Windows 10 [68]. Het is toch geprobeerd om te installeren zodat alle software op één laptop kon blijven staan. Het lukt om de TIA portal te installeren onder Windows 10 Professional, maar helaas werkt het niet vlekkeloos; De TIA Portal loopt regelmatig vast, en de License Manager en PLCSIM applicaties starten überhaupt niet op. Siemens biedt op het moment van schrijven nog steeds geen officiële ondersteuning voor Windows 10, ook niet op V14 welke tijdens het project werd uitgebracht. Daarom is er gekozen om de software van Siemens te installeren op een Windows 7 Pro virtuele machine in VMware Workstation [11]. In Figuur 24 is de voorheen genoemde Snap7-library genomen als voorbeeld van het pad dat wordt gevolgd voor de communicatie tussen de simulatie en PLCSim.



Figuur 24: Communicatie met virtuele PLC in PLCSIM

Deze opstelling heeft gedurende het onderzoek naar behoren gewerkt.

11 Sprint 6: Implementatie en initiële tests

In sprint 6 zijn de eerste integratie- en functionele tests uitgevoerd, door de betreffende libraries toe te voegen aan een Unity-project. Dit toont in eerste instantie aan of een library überhaupt werkt binnen de Unity-omgeving, maar ook wat het implementatiegemak is. Daarnaast wordt er ervaring opgedaan met hoe de libraries in elkaar steken, hoe deze worden aangesproken en welke data er mee kan worden verkregen. Eventuele tijdens de implementatie tegengekomen problemen worden tevens behandeld.

11.1 Installation test

Tijdens deze eerste test, welke plaatsvindt tijdens de implementatie van de oplossingen, wordt gekeken of de betreffende oplossing daadwerkelijk kan worden geïmplementeerd in Unity, zoals beschreven in het hoofdstuk “Ontwerp per gevonden oplossing” uit sprint 4. De korte tests uitgevoerd in deze sprint zullen betrekking hebben op de volgende eisen uit hoofdstuk 0:

Nr.	Beschrijving	Prioriteit
9	Nieuwe oplossingen moeten uit minder componenten (software, connecties, licenties etc.) bestaan dan de huidige, ter verlaging van de complexiteit van het geheel en de vermindering van kans op problemen.	Should
10	De software moet vanuit Unity benaderbaar zijn (te integreren in C#/.NET applicatie)	Must

Tabel 7: Geteste eisen Sprint 6

Indien een oplossing geïmplementeerd kan worden in Unity op de manier die staat beschreven in hoofdstuk 0 betekent dat dat er aan eis 9 en 10 is voldaan, zoals beschreven in het testrapport [Bijlage E]. Hiermee is de eerste compatibiliteitstest [69] voltooid.

11.2 Verloop en resultaten implementatie

Hieronder staat voor de libraries die zijn overgebleven uit sprint 4 hoe de implementatie in Unity is verlopen. Eventuele problemen hierbij en hun oplossingen zijn ook beschreven.

11.2.1 Snap7

De implementatie van Snap7 heeft iets meer om handen dan de meeste DLL’s, omdat deze geschreven is in C++. Hierdoor wordt deze door Unity behandeld als een zogenoemde ‘Native Plugin’ [70], waar wat meer haken en ogen aan zitten dan bij ‘Managed Plugins’ [71]. Bij de implementatietest wilde Unity de DLL niet herkennen totdat deze in exact de juiste map (/Assets/Plugins) werd geplaatst en in de Unity editor werd aangegeven met welke platforms en architecturen deze DLL compatibel is.

Daarnaast gaf de editor een foutmelding over een ‘API Compatibility level’. Een korte zoektocht op Google leverde op dat er in Unity onder Edit->Preferences->Player een optie staat die goed moet worden gezet. Dit betreft het *API Compatibility level* welke standaard op *.NET 2.0 Subset* staat, om zo de resulterende applicatie kleiner te kunnen houden [72]. Door deze te wijzigen in *.NET 2.0* werd deze foutmelding opgelost.

Met behulp van een C#-wrapper kan de DLL worden geïmplementeerd in Unity scripts. Deze wrapper is bij de download van Snap7 meegeleverd. Om de library werkend te krijgen zijn verder geen stappen benodigd, zoals het activeren van licenties. Daarnaast is de learning-curve vrij vlak waardoor de eerste implementatie binnen een dag kon worden uitgevoerd.

De implementatie geschiedde precies zoals beschreven in 9.1.1. Aan eis 9 en 10 is dus voldaan.

Qua functionaliteit zal er vooral worden gefocust op de client-functies die deze library biedt. Deze functies zullen voor de meeste doeleinden al voldoende mogelijkheden bieden om gegevens uit te wisselen met een PLC. Met deze functies kan in ieder geval worden gelezen en geschreven uit en naar datagebieden, datablokken, inputs, outputs, merkers, timers en counters. Daarnaast kunnen er controlefuncties worden uitgevoerd zoals hotstart, coldstart, stop, het uitlezen van systeemstatus en het uitlezen van systeeminformatie.

11.2.2 Sharp7

Sharp7 is een C# port van Snap7, waardoor deze functioneel exact hetzelfde is. De functieaanroepen zijn gelijk aan die in de Snap7-Wrapper, waardoor deze één-op- één zijn te vervangen. Omdat Sharp7 wordt geleverd als een enkel C# bestand (.cs) is deze gelijk in Unity te importeren zonder verdere instellingen. Wel moet ook bij Sharp7 het *API Compatibility Level* worden verhoogd naar .NET 2.0. Aan eis 9 en 10 is in ieder geval voldaan.

Het gebruiksgemak is wellicht wat hoger dan dat van Snap7, omdat er niet met een DLL gewerkt hoeft te worden. De C# code wordt pas gecompileerd in Unity, waardoor er ook geen problemen zijn met conflicterende .NET frameworks. Dit geeft ook een hogere toekomstbestendigheid voor wanneer bijvoorbeeld Unity, het OS of het platform wordt aangepast. Daarnaast is de library gemakkelijker te debuggen en aan te passen mocht dit nodig zijn.

11.2.3 S7.NET+

In een eerder project is er bij de opdrachtgever al geprobeerd om S7.NET, de voorganger van S7.NET+, te implementeren in Unity. Door onbekende redenen is dit toentertijd niet gelukt. S7.NET+ is een fork hiervan welke verder is doorontwikkeld en recenter is bijgewerkt.

De library komt in de vorm van een C# DLL en is dus een managed plugin in Unity. De library is gecompileerd tegen het .NET 3.5 framework en is dus te gebruiken in Unity. Ook deze library voldoet aan eis 9 en 10.

De library biedt iets minder functies dan Snap7 en Sharp7. Het is mogelijk om inputs, outputs, merkers, datablokken, timers en counters uit te lezen en te beschrijven. Verder kan er het een en ander aan systeeminformatie worden verkregen, zoals CPU-type.

11.2.4 OPC UA

De implementatie van OPC UA in Unity is helaas niet gelukt. Het blijkt dat alle vier de libraries genoemd in 9.1.7 gebruik maken van dezelfde core DLL welke problemen heeft met Unity. Het is mogelijk om de DLL's van de libraries toe te voegen aan Unity, en het lukt ook om de simulatie te compileren. Pas tijdens het draaien van de simulatie wordt er een exception gegooid. Alle vier de libraries geven namelijk exact dezelfde foutmelding wanneer er in de code wordt geprobeerd een verbinding op te zetten:

```
ServiceResultException: Error establishing a connection.
Opc.Ua.Bindings.TcpAsyncOperation`1[System.Int32].End (Int32 timeout)
```

Het verhogen van het *API Compatibility Level* haalde hier niets uit. Er is geprobeerd te onderzoeken waar deze exception vandaan komt. Allereerst is hiervoor de code uit het Unity-project gebruikt in een standalone Visual Studio-project. Hier werkt het opzetten van de connectie wel, zelfs als .NET framework 3.5 wordt geselecteerd.

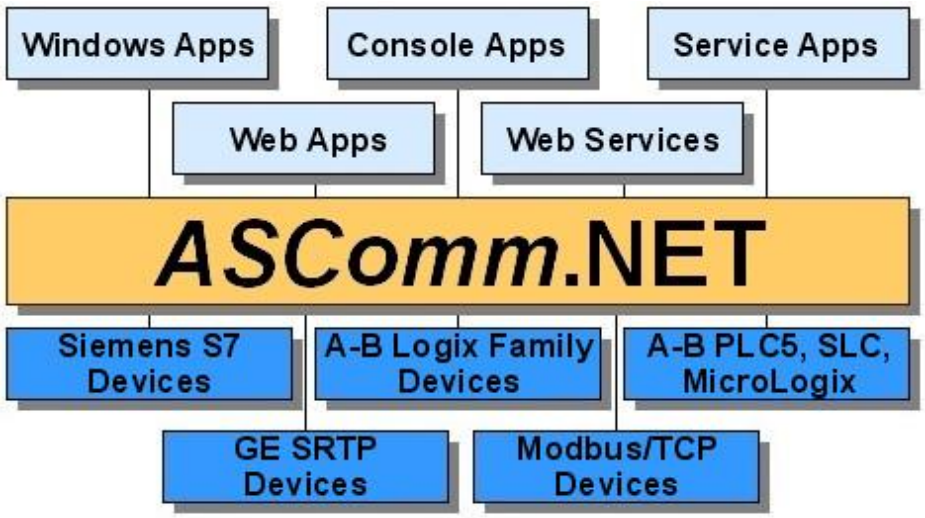
Er is hierna verder gezocht op internet naar deze foutmelding, en of deze voorkomt in combinatie met Unity. Deze zoektocht levert alleen een gesprek op het Unity-forum op waarin hetzelfde probleem met de core-library wordt besproken. [73] Voor de zekerheid is nog geprobeerd om dezelfde stappen te volgen die door de forumleden is voorgesteld, maar ook dit mocht niet baten.

De manier waarop de TCP-verbinding wordt geopend lijkt niet in goede aard te vallen bij de Unity-engine. Dit is waarschijnlijk te wijten aan de Mono-implementatie van het .NET framework, zoals beschreven in 7.1.4. Dit is echter niet 100% door afstudeerder vast te stellen, aangezien het debuggen van framework-libraries ver buiten het kennisbereik ligt en buiten de scope van het project valt. Hierdoor heeft de afstudeerder na een week besloten deze optie te laten vervallen omdat de libraries niet voldoen aan eis 10. Het is mogelijk om OPC UA te implementeren met *Shared Memory* zoals beschreven in 9.2.1. Omdat er echter goede alternatieven zijn voor OPC UA (WinMOD en ASComm.NET) is hier niet mee verdergegaan.

Aan eis 9 is ook niet voldaan; Een separate OPC Server is nog steeds nodig, en bij gebruik van shared memory is er nog steeds een tussenstap zoals getoond in 6.1.

11.2.5 ASComm.NET

De ASComm.NET library biedt de mogelijkheid om te communiceren met een breed scala aan PLC's. Aan de library kunnen losse drivers worden toegevoegd voor specifieke hardware, welke vervolgens met dezelfde methoden zijn aan te spreken. Deze drivers worden los aangeboden, maar kunnen ook worden aangeschaft als pakket. Figuur 25 laat de structuur zien van een applicatie welke gebruik maakt van ASComm.NET en welke onderdelen en drivers er beschikbaar zijn.



Figuur 25: Integratiemogelijkheden ASComm.NET [74]

Voor het onderzoek en de tests is gebruik gemaakt van de trial-versie van ASComm.NET. Deze bevat alle drivers die beschikbaar zijn. In geval er tot aanschaf van het softwarepakket wordt overgegaan, zullen niet al deze drivers nodig zijn. Zo wordt er bijvoorbeeld nooit gebruik gemaakt van Modbus RTU. Voor een redelijke prijs is er een driver suite beschikbaar met de volgende drivers:

- A-B ControlLogix, CompactLogix, & GuardLogix via Ethernet
- A-B PLC5, SLC500, and MicroLogix families via Ethernet
- A-B Micro800 Series (Micro820 & Micro 850) via Ethernet
- GE SRTP Master via Ethernet
- Modbus/TCP Master via Ethernet

- Siemens S7-200, S7-300, S7-400, S7-1200, & S7-1500 via Ethernet

Voor de redelijke prijs van $2.499,- voor één developer licentie, of $5.497,- voor drie licenties, biedt deze suite een grote veelzijdigheid aan ondersteunde hardware. Met deze developer licenties is het mogelijk om licentie-vrije runtime-applicaties te maken. In tegenstelling tot bijvoorbeeld WinMOD, hoeft er dus niet voor elke simulatie-release een licentie te worden aangeschaft. Tevens is de prijs voor drie developer licenties al flink lager dan één enkele WinMOD licentie.

Voor licentie-vrije runtime-applicaties moet wel aan een aantal voorwaarden worden voldaan [75]. Omdat er nog enige twijfel was of het gebruik van de software door JB Systems hieraan voldeed, is er aan de support van Automated Solutions een use-case voorgelegd. In deze use-case worden er voor een klant twintig identieke machines gebouwd. Elk van deze machines wordt geleverd met een op maat gemaakte 3D-simulatie welke gegevens kan uitwisselen met deze machine. Er zijn dus ook twintig kopieën van deze simulatie, maar slechts ontwikkeld voor deze ene klant. Op de vraag of deze situatie ook voldoet aan de voorwaarden voor licentie-vrije runtimedistributie, is er door de support bevestigd dat dit inderdaad het geval is, zolang de simulaties worden gebundeld met de machines, en niet los worden verkocht.

Integratie van ASComm.NET in Unity verloopt vrij soepel. Het enige probleem waar tegenaan is gelopen is dat het pakket niet wil werken in de 64-bits editor van Unity. Zodra er met de library een communicatiekanaal wordt geopend loopt de gehele Unity-editor vast zonder een duidelijke foutmelding.

Hierop is gepoogd in Visual Studio 2015 een losstaande console-applicatie te maken met de ASComm library, en deze exclusief voor 64-bits platforms te compileren. Aangezien deze applicatie wel functioneerde lijkt het er sterk op dat het probleem ligt bij Unity, en niet bij ASComm.NET.

Hierna is geprobeerd om met behulp van een stacktrace en memory-viewer in WinDBG [12] te volgen op welk punt de simulatie precies vastloopt. Hieruit is gebleken dat er zich een heap-corruption voordoet in ntdll.dll. Het vermoeden bestaat dat Unity iets probeert te doen buiten zijn eigen geheugenbereik, wellicht omdat er ergens een 32-bits geheugenadres wordt gebruikt in plaats van 64-bits. Vanaf dit punt bevindt het debuggen zich in een gebied dat zeer ver buiten de vaardigheden en kennis van de afstudeerder ligt. De opdrachtgever ziet het verder niet als probleem om gebruik te maken van de 32-bits Unity-editor, waarop is besloten deze kwestie te laten rusten.

De implementatie in Unity verloopt eenvoudig; De enige stappen bestaan uit het toevoegen van AutomatedSolutions.ASComm.dll aan de *Assets* map, en ook hier weer het verhogen van het *API Compatibility Level*. De documentatie en voorbeeldapplicaties zijn erg duidelijk, waardoor de library een lage leercurve kent. Het probleem met de 64-bits versie van Unity niet meegerekend, heeft de totale implementatie om tot een eenvoudige lees- en schrijfopdracht te komen minder dan een dag geduurd. Hiermee is voldaan aan eis 9 en 10.

12 Sprint 7: Testplan en testapplicatie

In sprint 7 is gefocust op het ontwikkelen van de testapplicaties en het uitvoeren van tests. Allereest is hiervoor aan de hand van de projecteisen een testplan opgesteld. Hierin staat aangegeven hoe met de testapplicatie de eisen kunnen worden gecontroleerd. Daarna wordt uitgelegd wat de ontwerpoverwegingen waren bij het ontwikkelen van de testapplicaties/-simulaties.

12.1 Testplan

Zoals beschreven in hoofdstuk 0 zijn de eisen in Tabel 8 van toepassing op de gekozen oplossingen, minus eis 9 en 10 welke al in Sprint 6 zijn behandeld.

Nr.	Beschrijving	Prioriteit
1	Snelheid van communicatie = 100Hz, bij 100 Reals per datastroom, één richting	Should
2	Oplossing mag niet duurder zijn dan huidige duurste mogelijkheid (WinMOD ~14€)	Must
3	De gekozen oplossing moet werken met Siemens S7 PLC's	Must
4	De gekozen oplossing moet werken met gangbare merken en types PLC's	Should
5	De gekozen oplossing moet universeel inzetbaar zijn voor alle types PLC	Could
6	De jitter mag niet meer dan 50ms bedragen	Must
7	De latency van één datatransmissie mag niet meer dan 10ms bedragen	Must
8	De software moet langer (meer dan achtenveertig uur) zonder blijven draaien dan de huidige oplossingen zonder vast te lopen	Must
11	Het is zeer gewenst om voor de het opzetten van de communicatie geen aanpassingen te hoeven doen aan de code op de PLC	Should
12	De gekozen oplossing mag geen merkbare invloed uitoefenen op de vernieuwingssnelheid (frames per second) van de simulatie	Must

Tabel 8: Eisen uit hoofdstuk 7

Met een aantal tests kan worden nagegaan of de oplossingen hieraan voldoen. Hieronder volgt een opsomming van de type tests die worden uitgevoerd, het doel van de tests en de eisen waarop deze betrekking hebben. Een aantal testtypes zijn bij elkaar genomen omdat de overlap hiertussen groot genoeg is, of omdat een bepaalde test meerdere eisen valideert.

12.2 Compatibility/functional

Tijdens deze tests wordt gekeken of de betreffende oplossing inderdaad te gebruiken is binnen de Unity-omgeving en wordt zover mogelijk gecontroleerd of deze werkt met de genoemde PLC's types. Kortgezegd wordt hiermee de vraag *"Is de oplossing aan de praat te krijgen?"* beantwoord. [69] Dit is de meest eenvoudige test welke kan worden uitgevoerd, desalniettemin erg belangrijk, omdat bij het niet slagen hiervan de implementatie geen doorgang kan vinden.

Deze tests zullen betrekking hebben op eis 3, 4, 5, 9, 10 en 11. De test zelf kan worden gezien als continu proces tijdens de eerste implementatie van de betreffende oplossing. In hoofdstuk 0 is reeds aan de hand van de betreffende documentatie van de oplossingen beschreven hoe deze geïmplementeerd moeten worden. Indien dit slaagt betekent dat in ieder geval dat aan de eisen 9 en 10 is voldaan.

Eisen 3, 4 en 5 zijn moeilijker te bewijzen, maar indien een softwarepakket werkbaar te krijgen is met ten minste een Siemens S7 PLC wordt eis 3 als voldaan beschouwd. Softwarepakketten welke beloven meerdere merken te ondersteunen zullen worden getest met een Allen-Bradley PLC, een Schneider Electric PLC en een Beckhoff PLC. Welke types dit betreft hangt af van wat er op het moment van testen beschikbaar is in het lab bij JB Systems.

Of een oplossing voldoet aan eis 11 zal gaandeweg tijdens het eerste testtraject duidelijk worden. Dit is echter een should-have eis, waardoor het pass/fail-moment aan enige interpretatie onderhevig is; Wanneer er bijvoorbeeld in de PLC-software alleen ergens een vinkje hoeft te worden geplaatst om een communicatieoplossing werkend te krijgen, betekent dit niet dat deze oplossing wordt afgeschreven. Wanneer er echter meerdere regels of blokken code moeten worden veranderd gaat dit wel spelen en kan de eis als 'niet voldaan' worden beschouwd.

12.3 Performance en reliability

Bij de performance- en reliabilitytests zal worden gemeten of de oplossingen voldoen aan de gestelde snelheids- en stabiliteitseisen [69, 76]. Dit zijn de eisen 1, 6, 7, 8 en 12. Voor de uitvoer van deze test is het volgende plan opgesteld:

Voor het testen van de stabiliteit zullen de oplossingen 48 uur achtereen worden uitgevoerd in het lab van JB Systems. Indien in deze 48 uur de applicatie niet vastloopt, geen ongewenst gedrag vertoont en de PLC waarden niet van elkaar afwijken, wordt de oplossing als stabiel bestempeld. Indien een van deze situaties zich toch voordoet moet er worden onderzocht of dit te wijten is aan de communicatieoplossing, of aan andere zaken als Unity of het OS. Dit kan worden vastgesteld met behulp van de logfiles. Met deze test wordt eis 8 bewezen.

Voor het testen van eis 1 wordt de snelheid gemeten door 100 keer per seconde een 100-tal Reals uit te lezen, er 1 bij op te tellen, en deze weer weg te schrijven. Wanneer de resulterende getallen worden gedeeld door het aantal seconden dat de applicatie heeft gedraaid, kan hieruit de snelheid worden bepaald. Dit getal moet naderen richting 100. Indien de Reals van elkaar afwijken na afloop van de test betekent dat er een aantal schrijfacties zijn overgeslagen. Dit is ook een indicatie van instabiliteit.

Het testen van gemiddelde latency en jitter (respectievelijk eis 7 en 6) kan ook redelijk eenvoudig worden bekeken. In Unity is het mogelijk om met de *profiler* te zien wat de verwerkingstijd van diverse onderdelen is. Hierin is de scripttijd het meest relevant. Als deze onder de 10ms blijft betekent dit dat de latency ook onder de 10ms ligt. Eventuele pieken kunnen hier ook in worden bekeken.

Het aantal frames per seconde moet minimaal 30 zijn. Dit is het algemeen geaccepteerde minimum voor vloeiend ogende beelden. Unity kan bijhouden wat de FPS is. Indien deze een getal hoger dan 30 aangeeft is aan deze eis voldaan.

12.4 Testapplicatie

De code voor het opvragen en wegschrijven van de data kan in Unity op een aantal plekken in het script worden gezet [77]:

FixedUpdate()
Deze functie wordt aangeroepen op gezette tijden welke door de programmeur worden bepaald. In Unity staat de frequentie van deze functie op 50 keer per seconde. Hiermee is de frequentie van de functie niet afhankelijk van de framerate.

Er moet wel worden opgemerkt dat zodra een FixedUpdate lang duurt, vanwege bijvoorbeeld een timeout of zware berekening, dat het volgende frame pas wordt gerenderd zodra de FixedUpdate klaar is. De FixedUpdate heeft dus wel invloed op de framerate.

Update() en LateUpdate()
De functies Update en LateUpdate worden eens per frame aangeroepen. Dit betekent dat de frequentie variabel is en afhankelijk van de framerate. Anderzijds is de framerate afhankelijk van de

logica die wordt uitgevoerd binnen deze functies, wanneer hier bijvoorbeeld zware berekeningen worden uitgevoerd.

Omdat het doel van de tests is om de maximale snelheid te bepalen, is de afhankelijkheid van de framerate ongewenst. Het is dus logischer om de communicatielogica in de FixedUpdate te plaatsen, zodat er met de hand een hogere snelheid kan worden ingesteld, terwijl de framerate gelijk blijft. Helaas is in dit geval wel de framerate afhankelijk van de communicatielogica.

12.4.1 Ontwerp

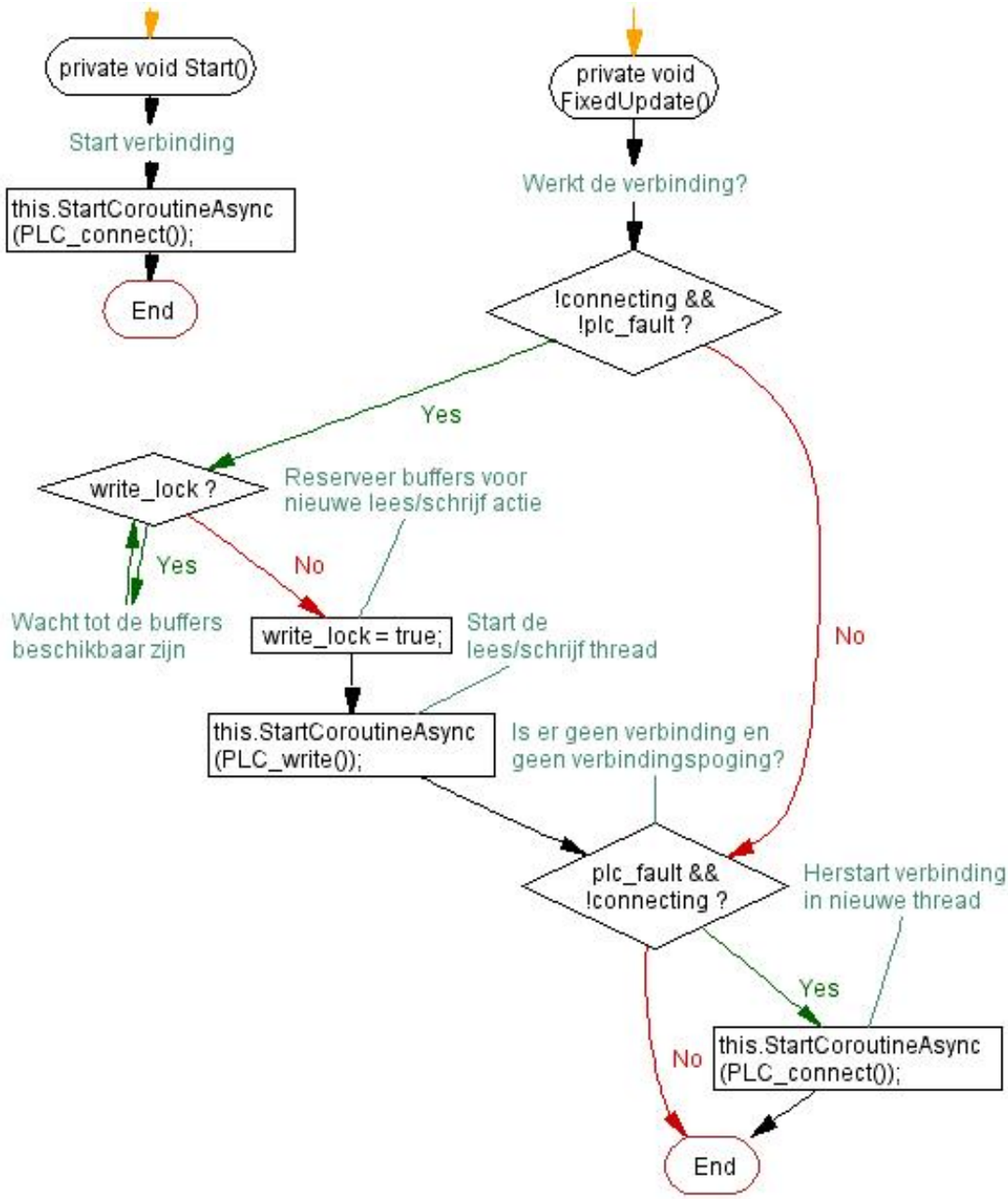
Om er zeker van te zijn dat de communicatie geen invloed heeft op de framerate en vice versa, wordt het communicatiegedeelte van de simulatie in een aparte thread uitgevoerd met behulp van ThreadNinja. De werking van deze threads is al beschreven in het hoofdstuk Threading. Threads kunnen worden gestart door de functie *this.StartCoroutineAsync()* aan te roepen. Dit wordt echter niet gedaan voor de ASComm.NET library, omdat deze van zichzelf al beschikt over asynchrone lees en schrijfacties, en bovendien zelfstandig de connectie kan herstellen mocht deze uitvallen. Voor de andere libraries wordt threading gebruikt, zodat wanneer er bijvoorbeeld een time-out optreedt, de simulatie niet blijft hangen.

Daarnaast wordt er gebruik gemaakt van een Tick-Tock model. Dit is gedaan omdat eis 1 stelt dat communicatiesnelheid wordt gemeten in één richting, dus alleen lezen of alleen schrijven. Tevens moet worden voorkomen dat er meer dan één extra thread wordt gestart, omdat er anders een raceprobleem ontstaat omdat er twee of meer threads de PLC en buffers willen aanspreken. Hiervoor is het wel van belang dat alle game-logica wordt uitgevoerd vóórdat de thread wordt gestart, omdat er anders een raceprobleem kan ontstaan tussen de game-logica en de communicatiethread.

Figuur 26 toont de globale werking van de testsimulaties: Zodra de simulatie start zal er een thread worden geopend waarin de connectie met de PLC wordt gestart. Dit wordt in een aparte thread gedaan zodat de framesnelheid niet instort wanneer er een timeout optreedt.

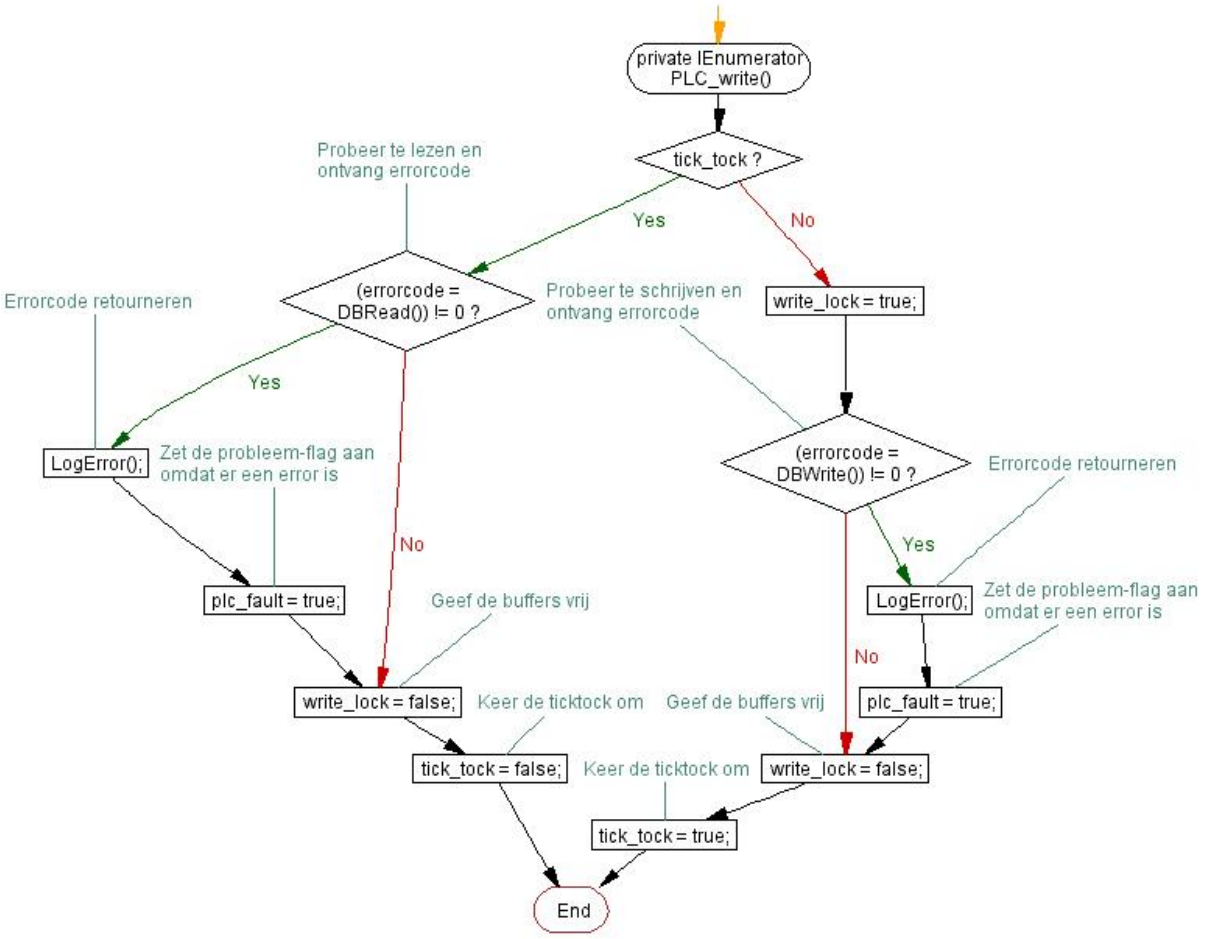
De applicatie zal vervolgens in de FixedUpdate kijken of er op dat moment een connectie wordt opgezet (connecting) en of dat er een probleem is met een bestaande verbinding (plc_fault). Zo niet, dan wordt er gewacht totdat de buffers beschikbaar zijn voor schrijven (write_lock). Zodra deze beschikbaar zijn worden ze gereserveerd voor een nieuwe communicatiethread die direct wordt opgezet. Deze thread zal de buffers vanzelf weer vrijgeven.

Als er ergens tijdens dit proces een probleem optreedt met de verbinding (plc_fault) en er wordt nog geen nieuwe verbinding opgezet (connecting), dan zal de connectiethread opnieuw worden gestart.



Figuur 26: Flowchart communicatie-applicatie

Als FixedUpdate is afgelopen blijft de communicatiethread (Figuur 27) in de achtergrond doorgaan, als deze is aangemaakt. Deze thread zal data naar de PLC schrijven, of deze uitlezen, afhankelijk van hetgeen in de voorgaande thread is uitgevoerd. Dit is een tick-tock systeem, om te voorkomen dat er tegelijkertijd wordt gelezen en geschreven. Tevens zorgt het om-en-om lezen en schrijven ervoor dat de uitvoer van de thread niet te lang duurt, en deze doorgaans klaar zal zijn voordat de volgende Update plaatsvindt. Voor elke lees- en schrijf actie worden de buffers gereserveerd (write_lock), zodat er geen race-conditions ontstaan en er in FixedUpdate maar één thread tegelijk wordt gestart. Wanneer een lees- of schrijfactie mislukt wordt de plc_fault flag geset, zodat tijdens de eerstvolgende update een nieuwe verbinding wordt opgezet.

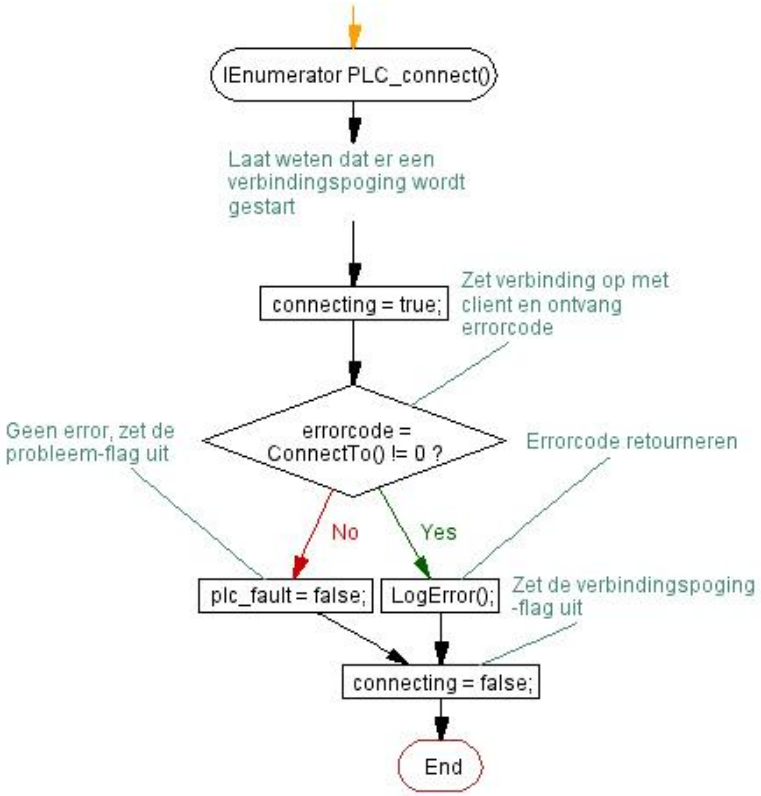


Figuur 27: Flowchart communicatiethread

De connectiethread (Figuur 28) wordt uitgevoerd wanneer de plc_fault flag aan staat en de connecting flag uit. Deze thread zit vrij eenvoudig in elkaar. Zodra deze thread wordt gemaakt wordt allereerst de connecting flag geset zodat er maar één verbindingspoging tegelijk plaatsvindt. Als het maken van de verbinding vervolgens slaagt wordt zowel de connected als de plc_fault flag uitgezet. Hierdoor zal in FixedUpdate de communicatie weer worden hervat.

Als er een foutmelding terugkomt bij de verbindingspoging wordt de error gelogd naar de console en de logfile, en wordt alleen de connecting flag uitgezet. Hierdoor kan er bij de volgende FixedUpdate weer een nieuwe verbindingspoging worden gestart.

Deze tests leggen alleen een benchmark voor de betreffende libraries en drivers. Wanneer de PLC of de PC zeer zwaar belast worden, kunnen de resultaten afwijken.



Figuur 28: Flowchart connectiethread

Deze applicatiestructuur is voor iedere communicatieoplossing hetzelfde, behalve voor ASComm.NET. De enige verschillen zijn de manieren waarop bijvoorbeeld de verbinding wordt opgezet of de data wordt uitgelezen. Deze oplossingsafhankelijke functies kunnen eenvoudig in het hierboven beschreven skelet worden geplaatst. De uiteindelijke code kan worden teruggevonden in [Bijlage G].

13 Sprint 8: Testvervolg en resultaten

In het lab van JB Systems was bij aanvang van de testsprint helaas geen Beckhoff PLC meer beschikbaar. Er waren nog wel een Siemens S7 1516-3 PN/DP PLC, een Allen-Bradley ControlLogix 5561 PLC en een Schneider Electric M340 PLC beschikbaar. De Siemens-specifieke libraries zullen worden getest met de S7 PLC. De ASComm.NET library zal worden getest met alle drie de PLC's. De code is terug te vinden in [Bijlage G]. De FixedUpdate is hierbij ingesteld op 10ms zodat het starten van lees- en schrijfacties op 100Hz wordt uitgevoerd.

Door na de testperiode de waardes op de PLC te delen door het aantal seconden dat Unity heeft gedraaid, is er een frequentie te bepalen. Deze frequentie geeft aan hoe vaak per seconde er een lees- én schrijfactie worden uitgevoerd, omdat in de testapplicaties voor elke verhoging van de PLC waardes zowel een lees- als een schrijfactie wordt verricht. Eis 1 stelt dat de frequentie van éénrichtingstransmissies moet worden bepaald. Door de lees- en schrijffrequentie te verdubbelen wordt de gemiddelde éénrichtingssnelheid bepaald.

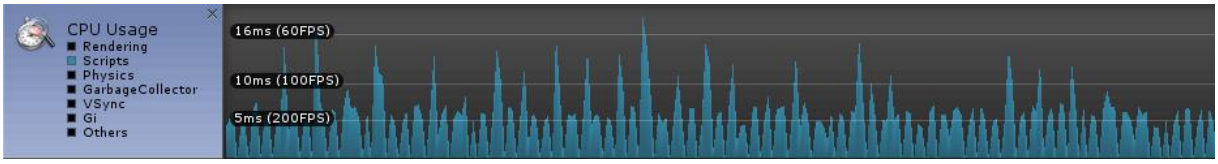
13.1 Resultaten per library

13.1.1 S7.NET+

Vanwege drukte in het lab bij JB Systems, is de S7.NET+ library helaas maar 24 uur in plaats van 48 uur getest. Hieruit is de volgende data verkregen.Gedurende de testperiode zijn er in de log geen foutmeldingen geregistreerd door Unity. Ook hebben de Reals op de PLC allen dezelfde waarde. Er is dus geen afwijking of drift ontstaan in de communicatie. Hierdoor mag worden gezegd dat de library voldoet aan eis 8.

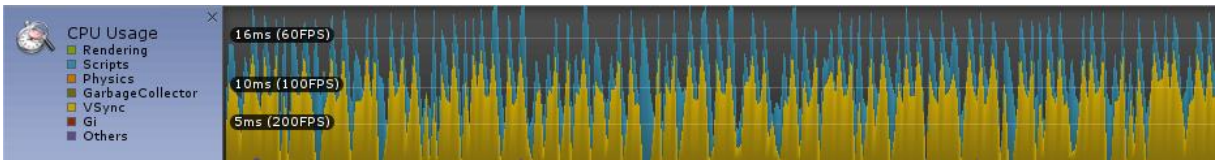
Na de testperiode was de waarde van de Reals op de PLC 4.271.651 en Unity's totale runtime eindigde op 85.868,08 seconde. Het delen van deze twee waarden levert een frequentie van 49,75Hz op. Omdat in deze test elke cycle wordt gelezen én geschreven resulteert dit voor een éénrichtingstransmissie in 99,49Hz. Hiermee voldoet de library aan eis 1.

De gemiddelde scripttijd ligt onder de 10ms, zoals te zien in Figuur 29. Dit betekent dat de communicatielatency hier ook onder ligt. Hiermee is aan eis 7 voldaan. Qua jitter zijn er enkele uitschieters richting de 20ms, maar nooit boven de 50ms. Hiermee is ook aan eis 6 voldaan.



Figuur 29: S7.NET+ Script timing

De gehele framesnelheid ligt tussen de 60 en de 100 FPS. Dit is ruim voldoende en voldoet aan eis 12.



Figuur 30: S7.NET+ Frame timing

13.1.2 Snap7

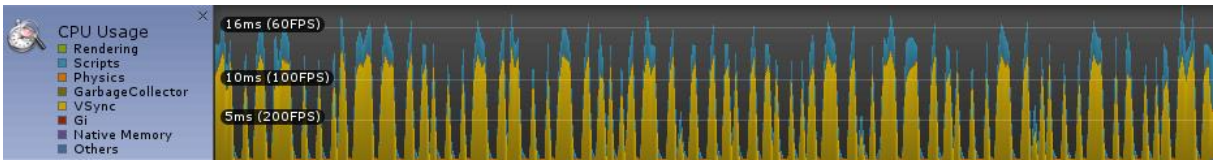
De Snap7 library is gedurende 42,5 uur getest in het lab. Gedurende deze periode zijn de Reals opgelopen tot 7.619.244 en heeft Unity 153.118,2 seconden geregistreerd. Dit resulteert in een

frequentie van 49,76Hz voor de lees-bewerk-schrijf cycles. Een enkele lees- of schrijfactie heeft hiermee een frequentie van 99,52Hz.

In Figuur 31 is te zien dat de gemiddelde scripttijd onder de 5ms ligt en dus voldoet aan eis 7, omdat de gemiddelde latency onder de 10ms ligt. Snap7 is daarmee iets efficiënter dan S7.NET+, maar heeft wel een aantal hogere pieken die richting de 40ms gaan. Dit valt nog onder de jitter-eis van 50ms, dus Snap7 voldoet hieraan.

Figuur 31: Snap7 Script timing

De gemiddelde framesnelheid van de gehele simulatie ligt rond de 100FPS. Helaas is er af en toe een enkele uitschieter net onder de 30. Er mag echter gesteld worden dat de speelbaarheid van de simulatie niet negatief wordt beïnvloed door de communicatie, omdat dit slecht om enkele frames gaat.

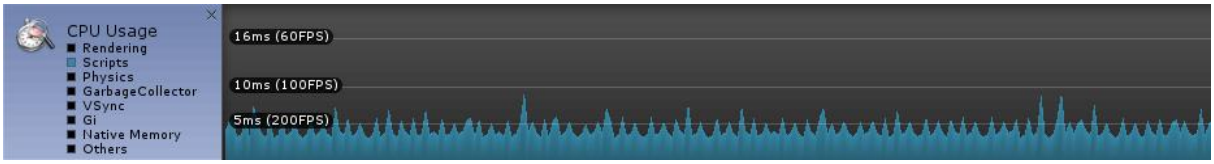


Figuur 32: Snap7 Frame timing

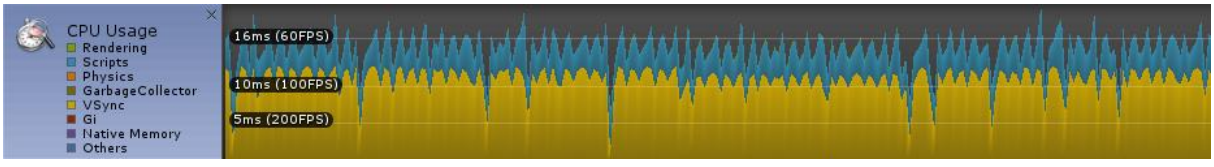
In de logs waren 5 read-errors aangetroffen. De simulatie is hierdoor niet vastgelopen, dus de applicatie kan de communicatie hervatten zoals verwacht in hoofdstuk 12.4. Dat de Reals op de PLC zijn toegenomen tot 7.619.244 betekent dat er in totaal 15.238.488 lees- en schrijfacties zijn geweest. Dit komt neer op een foutmarge van 0,00003% en is dus irrelevant.

13.1.3 Sharp7

De Sharp7 Library is net iets langer dan 48 uur zonder foutmeldingen getest in het lab. Hierbij zijn de Reals op de PLC opgelopen tot 8.393.613 en heeft Unity 173098.2 seconden geregistreerd. Dit komt neer op een frequentie van 48,49Hz. De éénrichtingstransmissiesnelheid komt hiermee op 96,98Hz. Eis 1 en 8 zijn hiermee voldaan.In Figuur 33 is te zien dat de gemiddelde scripttijd onder de 5ms ligt. Figuur 34 toont aan dat de gemiddelde framesnelheid rond de 70FPS ligt. Aan eis 6, 7 en 12 is hiermee voldaan.



Figuur 33: Sharp7 Script timing



Figuur 34: Sharp7 Frame timing

Waarom deze snelheid een aantal Hertz lager ligt dan de eerder geteste libraries is onbekend. Om te kijken of de library zelf de bottleneck is, is de snelheid van de FixedUpdate verhoogd naar 200Hz. Dit is gedurende een half uur getest. Door weer de waarde van de Reals te delen door de looptijd van de simulatie kwam hier een snelheid van 96,92Hz uit. De éénrichtingstransmissie komt hiermee op 193,84Hz uit. Het lijkt er dus op dat de library zelf geen moeite heeft met snelheden hoger dan 100Hz. Ook aan eis 1 is dus voldaan.

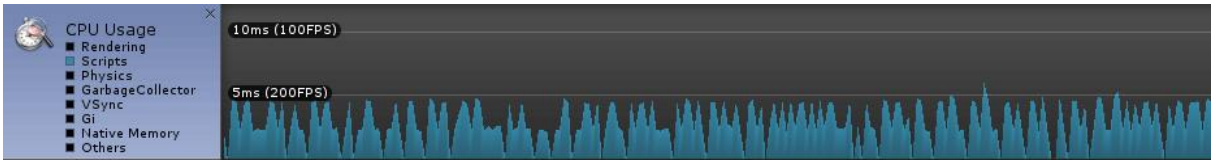
Het is lastig te verklaren waarom de frametijd hoger ligt dan die van Snap7, en waarom de uiteindelijke gemiddelde snelheid een paar Hertz lager ligt. Waarschijnlijk zit er in de .NET-gebaseerde Sharp7 library meer overhead dan in de Snap7 library welke is geschreven in C++. Allebei de libraries voldoen echter aan de eisen.

13.1.4 ASComm.NET i.c.m. Siemens

De Siemens driver van de ASComm.NET library is over een weekend heen getest voor bijna 70 uur. Gedurende deze tijd zijn er geen fouten of verbindingsproblemen voorgekomen en zijn de waardes van de verschillende Reals niet gaan afwijken. De library voldoet dus aan eis 8.

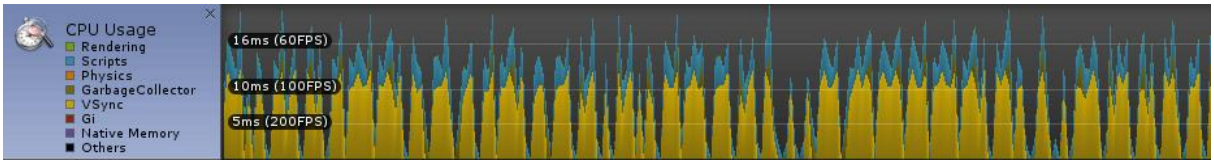
De uiteindelijke waarde van de Reals was 12.563.599, na een door Unity gemeten looptijd van 251.275 seconden. Hiermee is de frequentie vast te stellen op 49,999Hz. Hiermee is ASComm meteen al de library met de laagste afwijking. De éénrichtingssnelheid komt neer op 99,999Hz. Aan eis 1 is dus voldaan.

Figuur 35 laat zien dat de gemiddelde scripttijd en dus latency onder de 5ms ligt. Er zijn hierbij geen opvallende uitschieters. Hiermee is voldaan aan eis 6 en 7.



Figuur 35: ASComm Siemens Script timing

Figuur 36 laat zien hoe de gemiddelde framesnelheid tussen de 80 en de 100FPS ligt. Aan eis 12 is hiermee voldaan.

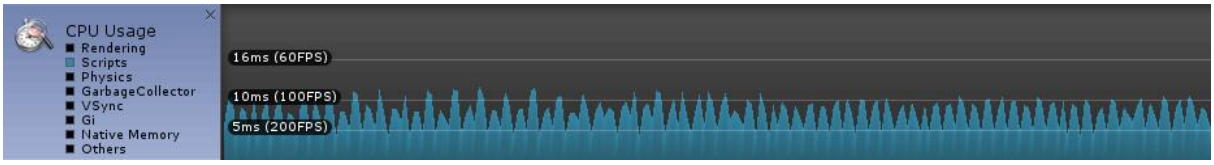


Figuur 36: ASComm Siemens Frame timing

13.1.5 ASComm.NET Allen-Bradley Logix driver

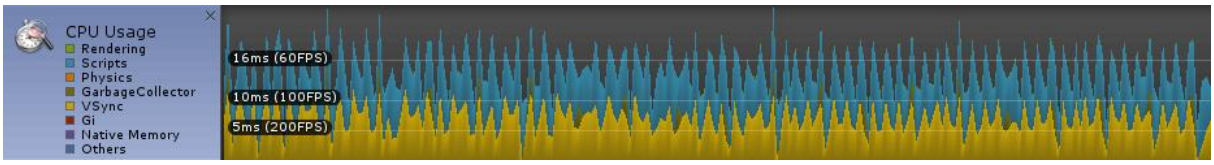
De Allen-Bradley Logix driver van de ASComm.NET library is getest gedurende 45 uur. Hierbij zijn geen fouten aangetroffen in de logs. De waarde van de Reals na deze periode was opgelopen tot 4.458.240 na een door Unity gemeten runtime van 161.948 seconden. Het ontbreken van verschillen tussen de Reals zorgt er voor dat er is voldaan aan eis 8.

De lees- en schrijffrequentie is vast te stellen op 27,53Hz. De éénrichtingstransmissie komt hiermee op 55,06Hz. Dit is net iets meer als de helft van de vereiste 100Hz, waarmee dus niet aan eis 1 is voldaan. Aan eis 6 en 7 is wel voldaan, omdat de gemiddelde latency onder de 10ms ligt en geen hoge uitschieters bevat.



Figuur 37: ASComm A-B Script timing

In Figuur 38 is te zien hoe de gemiddelde frametijd rond de 70FPS ligt, waarmee aan eis 12 is voldaan.



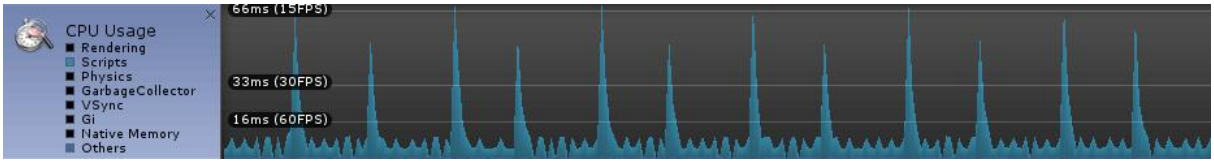
Figuur 38: ASComm A-B Frame timing

13.1.6 ASComm.NET Modbus (Schneider/Modicon)

Wegens drukte in het lab was het niet mogelijk om de Modbus functionaliteit van de ASComm.NET library te testen in combinatie met een Schneider PLC. Hierdoor is er met het softwarepakket Modbus Slave [5] getest, wat draait op dezelfde PC als waar de simulatie op draait. Door het elimineren van netwerkcomponenten zal dit systeem sneller werken dan met een fysieke PLC. In dit programma zijn 100 Floats aangemaakt in het zogenoemde Holding Register. Dit is waar variabelen zouden worden opgeslagen in de PLC [78]. Helaas heeft de gratis versie van Modbus Slave een maximale runtime van 10 minuten, hoewel dit genoeg is om te testen of de library werkt en om een *baseline* voor de snelheid te bepalen.

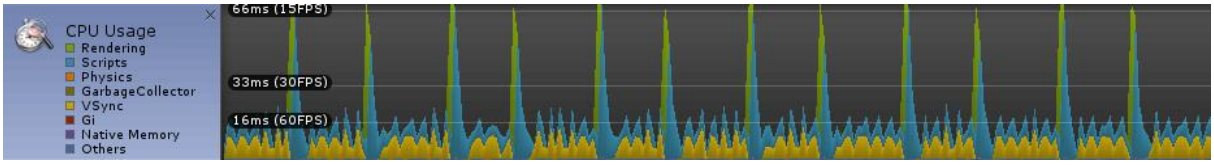
Vervolgens is er met Unity de test gedraaid. Hieruit bleek na bijna 10 minuten runtime dat Modbus bij lange na niet aan eis 1 voldoet. Na 555,0818 seconden dat de simulatie liep waren de Floats opgelopen tot 6142. Dit komt neer op een snelheid van 11,07Hz. Door dit om te rekenen naar éénrichtingstransmissie komt de snelheid uit op 22,13Hz. De simulatie draaide wel stabiel en zonder afwijkingen in de Floats. Gezien de magere snelheid van 22,13Hz, wat neerkomt op een cycletijd van 45,19ms, kan worden gezegd dat er niet is voldaan aan eis 1.

Kijkend naar de script- en frametijden vallen er een aantal dingen op. Allereerst zijn in de scripttijd in Figuur 39 hoge pieken te zien. Deze vinden plaats op het moment dat er daadwerkelijk een transmissie is geweest. Gezien de hoogte en onderlinge afstand van deze pieken wordt er niet aan eis 6 en 7 voldaan.



Figuur 39: ASComm Modbus Script timing

De gemiddelde framesnelheid ligt rond de 60FPS, waarmee dus wel voldaan is aan eis 12 (Figuur 40).



Figuur 40: ASComm Modbus Frame timing

13.2 Vergelijking van resultaten

In is een overzicht weergegeven van de gemeten snelheden van de libraries.

Library	Snelheid	Opmerkingen
S7.NET+	99,49Hz	
Snap7	99,52Hz	
Sharp7	96,98Hz (193,84Hz)	Kort getest op dubbele snelheid
ASComm.NET Siemens driver	100Hz	Reëel 99,999Hz
ASComm.NET Modbus driver	22,13Hz	Getest met virtuele PLC
ASComm.NET A-B driver	55,06Hz	

In Tabel 9 zijn de geteste libraries afgezet tegenover de bestaande communicatieoplossingen. In de één na laatste regel van de tabel is het totaal aantal plusjes per library getoond. Daaronder is er een totaalscore gegeven aan de oplossingen. Deze is gebaseerd op een getal dat is gehangen aan de eisen prioriteit. Zo is een must-have eis 4 punten waard, een should-have eis 3, een could-have 2 en won't-have 1. Hoewel ASComm.NET niet voor elke driver aan eis 1 en 7 voldoet krijgt deze toch een plusje, omdat eis 1 een should-have eis is en omdat minstens één andere driver wel aan deze eisen voldoet.

	Eis:	Snap7	Sharp7	S7.NET+	ASComm.NET	OPC DA*	WinMOD Y200*	S7.NET*
1	*Snelheid*	+	+	+	+	-	+	-
2	*Prijs*	+	+	+	+	+	-	+
3	*Siemens PLC's*	+	+	+	+	+	+	+
4	*Gangbare PLC's*	-	-	-	+	+	+	-
5	*Alle PLC's*	-	-	-	+	+	+	-
6	*Jitter*	+	+	+	+	-	+	-
7	*Latency*	+	+	+	+	-	+	-
8	*Stabiliteit*	+	+	+	+	+	+	+
9	*Complexiteit*	+	+	+	+	-	-	-
10	*Unity3D*	+	+	+	+	-	+	-
11	*Geen aanpassing PLC code*	+	+	+	+	+	+	+
12	*FPS*	+	+	+	+	+	+	+
	Totaal +	10	10	10	12	7	10	5
	Met Score	37	37	37	42	24	35	19

\+ Voldoet aan de eis
\- Voldoet niet aan de eis
* Oplossing uit huidige situatie

Tabel 9: Vergelijkingstabel met resultaten

14 Conclusie en aanbevelingen

In dit hoofdstuk wordt een conclusie getrokken uit de projectresultaten. Tevens worden aan de hand hiervan aanbevelingen gedaan.

14.1 Conclusie

De doelstelling welke voor aanvang van het project is opgesteld betrof het ontwikkelen van een architectuur waarmee Unity-simulaties gegevens uit een PLC kunnen halen. Hiertoe is bij aanvang van het project de volgende onderzoeksvraag opgesteld:

> *"Welke snelle en stabiele manieren van communicatie zijn mogelijk tussen Unity3D en een PLC?"*

Gekeken naar de oplossingen die zijn gevonden in het onderzoek kan worden geconcludeerd dat er zeker voldoende mogelijkheden zijn om het doel van het project te bewerkstelligen.

Het tweede gedeelte van de onderzoeksvraag, waarin wordt gekeken naar snelheid en stabiliteit, is daarna beantwoord door het vergelijkend onderzoek en de tests die met de verschillende oplossingen zijn uitgevoerd. Hieruit is naar voren gekomen dat er voldoende mogelijkheden zijn welke aan de performance-eisen voldoen.

14.2 Aanbevelingen

Kijkend naar de resultaten uit hoofdstuk 13 is er één oplossing welke aan alle eisen voldoet. Dit betreft de ASComm.NET library. Hiermee is het mogelijk om met nagenoeg elk type PLC te kunnen communiceren. Dit kan enorm schelen in de *'Time to market'* omdat de ontwikkelaars slechts één API hoeven te leren.

Het verdient dan ook de aanbeveling om van ASComm.NET één of meerdere licenties aan te schaffen. Deze licenties zijn voordeliger dan een enkele WinMOD licentie en er hoeft daarna niet voor deliverables te worden betaald. Dit heeft als bijkomend voordeel dat er na aanschaf nog 12 maanden professionele support wordt geleverd, al moet hierbij worden vermeld dat zelfs bij het gebruik van de trial-versie het geen probleem was om vragen te stellen aan de support.

Daarnaast moet er een eervolle vermelding voor de Sharp7 library worden gemaakt; Hoewel deze gratis library alleen werkt met Siemens S7 PLC's biedt deze veruit de meeste mogelijkheden. Implementatie is zeer eenvoudig en probleemloos omdat de library wordt aangeleverd als C#-broncode. Dit is een pluspunt ten opzichte van werken met DLL's qua debugging en platform-interoperabiliteit. Hoewel de Snap7 en S7.NET+ libraries dezelfde score krijgen in Tabel 9, zal Sharp7 hierdoor toch de voorkeur genieten. Daarnaast biedt Sharp7 (en Snap7) een aantal extra functies ten opzichte van S7.NET+.

Tot slot is het geen slecht idee om gebruik te maken van de WinMOD-Y200 combinatie, mits er voor het betreffende project reeds een WinMOD simulatie is ontwikkeld. Deze methode heeft namelijk vóór dit afstudeertraject al bewezen een snelle en stabiele oplossing te zijn, met als enig minpunt de prijs.

15 Projectevaluatie

In dit hoofdstuk wordt het afstudeertraject geëvalueerd. Deze evaluatie is op te delen in een productevaluatie en een procesevaluatie. Als laatste worden de beroepscompetenties geëvalueerd.

15.1 Productevaluatie

Bij aanvang van het afstudeerproject zijn er een aantal producten vastgesteld welke tijdens het project moesten worden opgeleverd. Dit betreft een aantal documenten en een proof-of-concept van de communicatieoplossingen.

De proof-of-concept is aan het einde van het traject opgeleverd en hier is uit gebleken dat de gevonden en gekozen communicatieoplossingen inderdaad aan de eisen voldoen. Hiermee is dus het uiteindelijke projectdoel behaald en zijn de onderzoeksvragen beantwoord. De opdrachtgever kan zelfs een keuze maken voor de beste oplossing per scenario, omdat sommige oplossingen betere specificaties hebben dan gesteld in de eisen, welke zijn vastgesteld in de definitiestudie [Bijlage C].

De opgeleverde testopstellingen zijn beschreven aan de hand van package-diagrammen, flowcharts en voorbeeldcode [Bijlage D]. Hierbij is tevens een universeel inzetbaar *blanco* script opgeleverd waarmee eventuele toekomstige communicatieprotocollen kunnen worden geïntegreerd [Bijlage G]. Hier kunnen de functies van de library worden geplaatst samen met de bewerkingen die op de data moeten worden uitgevoerd. Het script zelf zorgt vervolgens voor de parallellisatie en een zelfherstellende connectie. Daarnaast is er een voorbeeld opgeleverd waarmee met behulp van shared-memory libraries kunnen worden geïntegreerd welke niet compatibel zijn met Unity, hoewel dit buiten de scope van het project viel.

Voor de gekozen softwarepakketten zijn tests ontworpen en uitgevoerd, waarvan het traject is vastgelegd in het testrapport [Bijlage E]. Deze tests zijn gemakkelijk te repliceren en te verifiëren. De software hiervoor is opgeleverd aan de opdrachtgever. De tests tonen aan of de betreffende communicatieoplossing aan de gestelde eisen voldoet.

15.2 Procesevaluatie

In het plan van aanpak [Bijlage B] is een planning gemaakt welke tevens te zien is in het hoofdstuk Planning. Daarnaast is er gewerkt volgens de Scrum methodiek. Aan deze twee zaken zijn een aantal dingen op te merken.

Allereerst heeft de gebruikte Scrum methode een aantal wijzigingen ondergaan om deze geschikt te kunnen maken voor de specifieke projectkenmerken. Voorbeelden hiervan zijn het gebruik van Scrum in een éénpersoonsteam en het opleveren van producten anders dan software. Deze manier van werken bleek prima te werken. Er is gedurende het gehele project geen tijdsnood geweest en alle beloofde producten zijn opgeleverd.

Er heeft wel meer overlap gezeten in de onderdelen van de planning dan was voorzien. Dit is echter zonder problemen opgevangen. Door een aantal zaken naar een vorige of volgende sprint te schuiven was het mogelijk om wat elasticiteit aan te brengen in de planning, terwijl de betreffende sprint kon worden afgerond.

Deze overlap is te wijten aan twee zaken. Allereerst zijn er later in het traject nog een aantal softwarepakketten gevonden, terwijl er al was begonnen aan implementatie en initiële tests van de reeds gevonden libraries. Ook kwam in de loop van het project de Sharp7 library uit. Het zou zonde zijn om deze niet mee te nemen in de vergelijking. Deze is dus in een latere sprint behandeld dan de eerdere libraries.

Daarnaast is er helaas veel tijd besteed aan het wekend krijgen van oplossingen die eigenlijk niet werkbaar bleken. Een goed voorbeeld hiervan is de ASComm.NET library onder de 64-bits versie van Unity. Hier is bijna een hele week ingestoken om deze werkend te krijgen, terwijl dit gewoon een limitatie lijkt te zijn van het in Unity gebruikte Mono-framework.

Ook is er tegen beter weten in te lang doorgegaan met de CSP Ethernetdriver. Het was veel werk om deze aan de praat te krijgen, terwijl de library achteraf helemaal geen portabiliteit bleek te hebben. Dit is dus zonde van de tijd geweest.

Scrum kan gelukkig goed omgaan met veranderende situaties en eisen, waardoor het gemakkelijk was het project bij te sturen. De opdrachtgever is hierbij op de hoogte gehouden met behulp van een interne backlog-spreadsheet en door elke week een voorgangsgesprek te houden. Daarnaast is er met andere stagiairs en afstudeerders nagenoeg elke dag een korte scrum-meeting gehouden waarbij een ieder elkaar op de hoogte hield van de voorgang en aangetroffen problemen. Dit alles zorgde voor een natuurlijk verloop van het proces.

15.3 Competenties en beroepstaken

Voor aanvang van het afstudeerproject zijn in het afstudeerplan [Bijlage A] een viertal beroepscompetenties uitgekozen waar in de loop van het project extra op wordt gefocust. Van deze competenties wordt in dit hoofdstuk beschreven hoe deze tijdens het project zijn aangetoond. Er is uitgegaan van één competentie per projectfase, hoewel het onvermijdelijk is dat er wat overlap optreedt, omdat er in iteraties is gewerkt.

15.3.1 Praktische aspecten hanteren in (internationale) projecten (G1)

Bij dit project is er rekening gehouden met de praktische aspecten. Dit is met name naar voren gekomen in de eerste fase, waarin het Plan van aanpak [Bijlage B] is opgeleverd. Hierin is de volledige opdrachtomschrijving uitgewerkt, is er een risicoanalyse opgesteld en is er een ontwikkelmethodiek gekozen.

De opdrachtomschrijving bevat de aanleiding, probleemstelling, doelstelling en gewenste resultaten. Dit vormt een consistent geheel waaruit het doel en verloop van de afstudeeropdracht duidelijk naar voren komt.

15.3.2 Analyseren van het probleemdomein (A1)

Om te weten welke veranderingen er moeten worden doorgevoerd, zal er eerst moeten worden gekeken wat de huidige problemen precies omvatten. Hierom is in de tweede fase van het project, de definitiefase, het probleemdomein geanalyseerd. Hiervoor zijn ten eerste gesprekken gehouden met de opdrachtgever en enkele ontwikkelaars welke zich bezighouden met de simulaties. Er is hierbij achterhaald met welke middelen en softwarepakketten er werd gewerkt en welke problemen hierbij naar boven kwamen.

Deze zaken zijn door de afstudeerder vervolgens verder bestudeerd om kennis op te doen over de werking en het gebruik hiervan. Dit is gedaan door middel van zowel zelfstudie als door korte sessies (crashcourses) met werknemers van JB Systems. Dit heeft inzicht gegeven in de problemen die werden ondervonden voorafgaand aan de afstudeeropdracht.

Dit inzicht in het probleemdomein is vervolgens uitvoerig gedocumenteerd en is vertaald naar eisen voor het nieuw te ontwikkelen ontwerp. Deze informatie is samengekomen in de definitiestudie [Bijlage C].

15.3.3 Ontwerpen van een systeemarchitectuur (C10)

Aan de hand van de eisen die zijn voortgekomen uit de definitiefase is een ontwerp voor meerdere mogelijke systeemarchitecturen gemaakt. Relevant hierbij is dat voorafgaand aan dit ontwerp tevens een onderzoek is gedaan naar de mogelijke manieren om de communicatie met PLC's te bewerkstelligen. Er is zowel bij dit onderzoek als bij de ontwerpkeuze rekening gehouden met zaken als open- versus geslotenheid, werking met en aansluiting op bestaande systemen en voorkeuren van de opdrachtgever.

De uiteindelijke ontworpen architecturen hebben allen gemeen dat de werken binnen het .NET 3.5 framework en beschikken over één of meerdere gangbare communicatieprotocollen binnen de automatiseringsindustrie (bijvoorbeeld S7, AB Ethernet, Modbus). De onderlinge verschillen zijn echter dermate groot dat er niet één oplossing is welke 100% aan de eisen voldoet, hoewel ASComm hierbij zeer sterk in de buurt komt.

De opgeleverde architecturen bieden echter ook een keuze aan de opdrachtgever, zodat deze een oplossing kan kiezen aan de hand van projecteisen. Elk project is natuurlijk anders, en de ene eis kan hierbij zwaarder wegen dan de ander. Hier is op dit moment geen rekening mee te houden, maar de opgeleverde architecturen bieden een helder overzicht waarmee de opdrachtgever een weloverwogen keuze kan maken. Het traject en de resultaten uit deze fase zijn beschreven in het ontwerprapport [Bijlage D].

15.3.4 Testen van softwaresystemen (D17)

In de laatste fase van het project, de realisatiefase, heeft de nadruk gelegen op het aantonen van de beroepscompetentie *'Testen van softwaresystemen'*. De opgeleverde proof-of-concepts zijn op een aantal punten getest. Ten eerste om kijken of eventuele specificaties van libraries waargemaakt konden worden, en ten tweede om vergelijkend bewijs te hebben tussen de verschillende oplossingen.

Voor aanvang van het testen zijn aan de hand van de eisen een aantal tests opgesteld. Hiermee was het mogelijk de gevonden oplossingen aan de eisen te toetsen. Helaas was het voor sommige eisen lastig of onmogelijk om een test te ontwikkelen die binnen het redelijke uitvoerbaar is.

De resultaten van deze fase zijn vastgelegd in het testrapport [Bijlage E].

16 Verklarende woordenlijst en afkortingen

Woord	Omschrijving
Context Switch	Het opslaan en herstellen van de staat van een proces of taak, zodat deze later kan worden hervat.
Coroutine	'Parallelle' taak waarbij de context switching door de programmeur moet worden gepland.
Jitter	Een variërende (negatieve) afwijking in de transporttijd van datapakketten. Ook wel *packet delay variation* genoemd.
Localhost	Loopbackinterface: Dit wordt in netwerken gebruikt om te verwijzen naar het eigen systeem, de 'lokale host'.
Real	Andere benaming voor een 32-bit IEEE floating-point getal, vaak gebruikt bij PLC's.
Thread	Een echte parallelle taak waarbij context switching door het OS of de omgeving wordt afgehandeld op instructieniveau.
Thread-safe	Wanneer multithreaded software geen data aanspreekt op een manier die race condities, locks of andere problemen kan opleveren.
Tick-tock	Een constructie waarbij steeds om-en-om wordt afgewisseld tussen twee handelingen.

Afkorting	Verklaring
API	Application Programming Interface
CPU	Central Processing Unit
DB	Data Block
DLL	Dynamic Link Library
FPS	Frames Per Second
HIL	Hardware-In-Loop
HMI	Human-Machine Interface
MSVC	MicroSoft Visual C++
MSVS	MicroSoft Visual Studio
OLE	Object Linking and Embedding
OPC	OLE for Process Control
OPC DA	OPC Data Access
OPC UA	OPC Unified Architecture
OS	Operating System
PLC	Programmable Logic Controller
SCADA	Supervisory Control And Data Acquisition
TIA	Totally Integrated Automation
VM	Virtual Machine

17 Bibliografie

[1] A. Coppens, M. Jacobs, T. Jacobs, R. Niels en N. Verhoeven, „Proeven van onderzoek - De methodenkaart in de beroepspraktijk van ICT en Media," 04 2015. [Online]. Available: http://www.ralphniels.nl/pubs/jacobs-proevenvanonderzoekboek.pdf. [Geopend 09 09 2016].

[2] Astah, „Change Vision, Inc.," [Online]. Available: http://astah.net/. [Geopend 12 10 2019].

[3] Microsoft, Inc., „Microsoft Office," [Online]. Available: https://products.office.com/nl-nl/home. [Geopend 29 08 2016].

[4] Microsoft, Inc., „Microsoft Windows 10," [Online]. Available: https://www.microsoft.com/nl-nl/windows/get-windows-10. [Geopend 29 08 2016].

[5] Witte Software, „Modbus Slave," 2016. [Online]. Available: http://www.modbustools.com/modbus_slave.html. [Geopend 09 12 2016].

[6] T. Wiens, „NetToPLCSim," 2015. [Online]. Available: http://nettoplcsim.sourceforge.net/. [Geopend 14 09 2016].

[7] Siemens AG, „SIMATIC S7-PLCSIM," Siemens AG, [Online]. Available: http://w3.siemens.com/mcms/simatic-controller-software/en/step7/simatic-s7-plcsim/pages/default.aspx. [Geopend 14 09 2016].

[8] Siemens AG, „TIA Portal Software," Siemens AG, [Online]. Available: http://www.industry.siemens.nl/automation/nl/nl/industriele-automatisering/automatiserings-software/tia-portal-software/Pages/tia-portal-software.aspx. [Geopend 29 08 2016].

[9] Unity Technologies, „Unity3D," [Online]. Available: https://unity3d.com/. [Geopend 29 08 2016].

[10] „Visual Studio Community," [Online]. Available: https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx. [Geopend 29 08 2016].

[11] VMware, Inc, „Workstation for Windows," VMware, Inc, 2016. [Online]. Available: http://www.vmware.com/products/workstation.html?src=WWW_Workstation12.5_US_HPpromoLL_LearnMore. [Geopend 14 09 2016].

[12] Microsoft, Inc., „Debugging tools fot Windows," [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/hardware/ff551063(v=vs.85).aspx. [Geopend 10 11 2016].

[13] R. M. Iver, „Scrum Is Not Just for Software - A real-life application of Scrum outside IT.," Scrum Alliance, Westminster, 2009.

[14] F. Iwanitz en J. Lange, OPC Fundamentals, Implementation and Application, Laxmi Publications, 2010.

[15] X. Hong en J. Wang, „Using standard components in automation industry: A study on OPC Specification," Elsevier, Xi'an, 2005.

[16] Advosol Inc., „OPC DA .Net Client Development Component," [Online]. Available: http://www.advosol.com/pc-1-3-opcdanet.aspx. [Geopend 20 09 2016].

[17] A. Davis, „Unity and DLLs," 11 12 2015. [Online]. Available: http://www.what-could-possibly-go-wrong.com/unity-and-dlls/#compilingadlltouseinunity. [Geopend 20 09 2016].

[18] R. Hauwert, „The future of scripting in Unity," Unity Technologies, 20 05 2014. [Online]. Available: https://blogs.unity3d.com/2014/05/20/the-future-of-scripting-in-unity/. [Geopend 20 09 2016].

[19] Kepware, Inc., „KEPServerEX," Kepware, Inc., [Online]. Available: https://www.kepware.com/products/kepserverex/. [Geopend 05 01 2016].

[20] KepWare Technologies, „Price List PRICE-SS-UN-05-2016," 12 05 2016. [Online]. Available: https://www.kepware.com/products/pricelist.pdf. [Geopend 08 09 2016].

[21] Mesta Automation, „PLC-PC communication with C#: a quick resume about data exchange libraries," 11 04 2012. [Online]. Available: http://www.mesta-automation.com/plc-pc-communication-with-c-a-quick-resume-about-data-exchange-libraries/. [Geopend 01 08 2016].

[22] L. Tundong, C. Gangquan en P. Xiafu, „OPC Server Software Design in DCS," IEEE, Xiamen, 2009.

[23] Mewes & Partner, „WinMOD Systems for Periphery Simulation," [Online]. Available: http://winmod.de/en/index.php?page=peripherisimulation. [Geopend 21 09 2016].

[24] Mewes & Partner, „WinMOD Configurations," 2010. [Online]. Available: http://winmod.de/en/uploads/WinMOD-Configurations_5.1_en.pdf. [Geopend 21 09 2016].

[25] Mewes & Partner, „WinMOD Configuration Y200," Mewes & Partner, Hennigsdorf, 2014.

[26] Mewes & Partner, „WinMOD System Software V7.2 Manual," Mewes & Partner, Henningsdorf, 2015.

[27] Mewes & Partner, „WinMOD Configurations Data Sheets," 2016. [Online]. Available: http://winmod.de/en/index.php?page=winmod-konfigurationen. [Geopend 26 09 2016].

[28] Juergen1969, „S7.Net," Codeplex, 04 10 2009. [Online]. Available: https://s7net.codeplex.com/. [Geopend 20 09 2016].

[29] D. Nardella, „Siemens communications overview," Snap7, [Online]. Available: http://snap7.sourceforge.net/siemens_comm.html. [Geopend 20 09 2016].

[30] T. Wiens, „S7 Communication (S7comm)," Wireshark.org, 13 05 2016. [Online]. Available: https://wiki.wireshark.org/S7comm. [Geopend 20 09 2016].

[31] Unity Technologies, „Unity Roadmap,” [Online]. Available: https://unity3d.com/unity/roadmap. [Geopend 21 09 2016].

[32] S. H. Kan, Metrics and Models in Software Quality Engineering, Boston, MA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[33] V. Y. Shen, T.-J. Yu, S. M. Thebaut en L. R. Paulsen, „Identifying Error-Prone Software - An Empirical Study,” *IEEE Transactions On Software Engineering,* vol. 11, nr. 4, p. 8, 1985.

[34] T. M. Khoshgoftaar en J. C. Munson, „Identifying Software Development Errors Using Software Complexity Metrics,” *IEEE Journal On Selected Areas In Communications,* vol. 8, nr. 2, p. 9, 1990.

[35] B. Beizer, Software Testing Techniques, Dreamtech Press, 2002.

[36] H. v. Vliet, Software Engineering: Principles and Practice [PDF], Wiley: Chichester, 2007.

[37] Van Oord Offshore BV, „http://cdn.vanoord.com,” 04 2015. [Online]. Available: http://cdn.vanoord.com/sites/default/files/leaflet_bravenes_april2015_a4_lr.pdf. [Geopend 05 10 2016].

[38] Mono Project, „FAQ: General,” Mono Project, 2016. [Online]. Available: http://www.mono-project.com/docs/faq/general/. [Geopend 20 09 2016].

[39] C. Anagnostopoulos en A. Kalogeras, „An Industrial Simulator Utilizing a Gaming Platform,” IEEE, Patras, 2015.

[40] A. Lakshmi Sangeetha, B. Naveenkumar, A. Balaji Ganesh en N. Bharathi, „Experimental validation of PID based cascade control system through SCADA–PLC–OPC and internet architectures,” Elsevier, Chennai, 2012.

[41] T. Piiparinen, „Development of a PLC-based Control & Monitoring Device,” Helsinki Metropolia University of Applied Sciences, Helsinki, 2014.

[42] H. Schollier, „Implementatie van motion capture in operator training systemen,” eXpertisecentrum industriële automatisering Kortrijk, Kortrijk, 2016.

[43] S. Shujie, H. Lin, Z. Liaomo en H. Yi, „Design and simulation of the production line system based on multi-channel controller,” IEEE, Shenyang, 2015.

[44] D. Nardella, „Snap7,” [Online]. Available: http://snap7.sourceforge.net/. [Geopend 05 01 2016].

[45] D. Nardella, „Sharp7 Project overview,” 07 10 2016. [Online]. Available: http://snap7.sourceforge.net/sharp7.html. [Geopend 14 10 2016].

[46] D. Heiser, „S7.NET+,” 2016. [Online]. Available: https://github.com/killnine/s7netplus. [Geopend 12 09 2016].

[47] T. Hergenhahn, „LIBNODAVE,” Sourceforge, 22 05 2014. [Online]. Available: http://libnodave.sourceforge.net/. [Geopend 26 09 2016].

[48] T. Hergenhahn, „LibOpenSRTP,” Sourceforge, 21 03 2005. [Online]. Available: http://libopensrtp.sourceforge.net/. [Geopend 26 09 2016].

[49] dotnetprojects, „DotNetSiemensPLCToolBoxLibrary,” 2016. [Online]. Available: https://github.com/dotnetprojects/DotNetSiemensPLCToolBoxLibrary. [Geopend 12 09 2016].

[50] Archie, „Allen Bradley SLC 5/05 Ethernet Driver,” 01 04 2013. [Online]. Available: http://abethernet.sourceforge.net/. [Geopend 12 09 2016].

[51] Parijat Controlware Inc., „Rockwell/Allen-Bradley Ethernet Protocol .NET Communications Driver,” Parijat Controlware Inc., 12 08 2016. [Online]. Available: https://visualstudiogallery.msdn.microsoft.com/afd91e85-6cff-4974-9340-14b94d12c864. [Geopend 21 09 2016].

[52] D. Turin, „NModbus4,” GitHub, 29 06 2016. [Online]. Available: https://github.com/NModbus4/NModbus4. [Geopend 12 09 2016].

[53] Siemens AG, „SIMATIC Modbus/TCP via the integrated PN interface of the S7-300/400 CPU,” 28 07 2014. [Online]. Available: https://support.industry.siemens.com/cs/attachments/22660304/modbustcppncpu_en.pdf. [Geopend 01 09 2015].

[54] OPC Foundation, „Build OPC UA .NET applications using C#, VB.NET,” OPC Foundation, 2015. [Online]. Available: http://opcfoundation.github.io/UA-.NET/. [Geopend 05 09 2016].

[55] Xcelgo A/S, „Product – Experior,” Xcelgo A/S, [Online]. Available: http://xcelgo.com/experior/. [Geopend 09 09 2016].

[56] Automated Solutions, Inc., „ASComm.NET Allen-Bradley, Siemens S7, GE, & Modbus Drivers for .NET Developers,” 2016. [Online]. Available: http://automatedsolutions.com/products/dotnet/ascomm/. [Geopend 22 10 2016].

[57] J. Scholte, „Optimalisatie van PLC-PC communicatie met LabVIEW,” Haagse Hogeschool, Delft, 2016.

[58] D. Heiser, „s7netplus/Documentation/Documentation.pdf,” 12 06 2016. [Online]. Available: https://github.com/killnine/s7netplus/blob/master/Documentation/Documentation.pdf. [Geopend 03 10 2016].

[59] Unified Automation GmbH, „.NET Based OPC UA Client SDK,” Unified Automation GmbH, 2016. [Online]. Available: https://www.unified-automation.com/products/client-sdk/net-ua-client-sdk.html. [Geopend 05 10 2016].

[60] Prosys PMS Ltd, „OPC UA .NET SDK,” Prosys PMS Ltd, 2016. [Online]. Available: https://www.prosysopc.com/products/opc-ua-dotnet-sdk/. [Geopend 05 10 2016].

[61] Softing Industrial Automation GmbH, „OPC UA .NET Server and Client Development Toolkits,” Softing Industrial Automation GmbH, 2016. [Online]. Available: http://industrial.softing.com/en/products/software/opc-development-toolkits/opc-ua-net-

development-toolkit/opc-ua-net-server-client-toolkit-for-windows.html. [Geopend 05 10 2016].

[62] Siemens AG, „Programming an OPC UA .NET Client with C# for the SIMATIC NET OPC UA Server," Siemens AG, 09 04 2014. [Online]. Available: https://support.industry.siemens.com/cs/document/42014088/programming-an-opc-ua-net-client-with-c-for-the-simatic-net-opc-ua-server?dti=0&lc=en-WW. [Geopend 05 10 2016].

[63] J. Stenning, „C# SharedMemory," 25 07 2016. [Online]. Available: https://sharedmemory.codeplex.com/. [Geopend 14 10 2016].

[64] F. Zapata, „CoroutineScheduler," Unify Community, 30 01 2014. [Online]. Available: http://wiki.unity3d.com/index.php?title=CoroutineScheduler. [Geopend 24 10 2016].

[65] Unity Answers, „Threading in Unity," 26 10 2011. [Online]. Available: http://answers.unity3d.com/questions/180243/threading-in-unity.html. [Geopend 24 10 2016].

[66] C. Spike, „Thread Ninja - Multithread Coroutine," Unity Assetstore, 06 03 2014. [Online]. Available: https://www.assetstore.unity3d.com/en/#!/content/15717. [Geopend 07 10 2016].

[67] T. Wiens, „NetToPLCSim - Overview," [Online]. Available: http://nettoplcsim.sourceforge.net/nettoplcsim-s7o-en.html#Overview. [Geopend 15 09 2016].

[68] Siemens AG, „SIMATIC STEP 7 Basic V13 - System requirements," Siemens AG, 2016. [Online]. Available: https://www.industry.siemens.com/topics/global/en/tia-portal/controller-sw-tia-portal/simatic-step7-basic-tia-portal/system-requirements/pages/default.aspx. [Geopend 05 10 2016].

[69] G. J. Myers, T. Badgett en C. Sandler, „The Art of Software Testing," John Wiley & Sons, Chichester, 2011.

[70] Unity Technologies, „Native Plugins," Unity3D, 2016. [Online]. Available: https://docs.unity3d.com/Manual/NativePlugins.html. [Geopend 10 11 2016].

[71] Unity Technologies, „Managed Plugins," Unity Technologies, 2016. [Online]. Available: https://docs.unity3d.com/Manual/UsingDLL.html. [Geopend 10 11 2016].

[72] Unity Technologies, „Mono Compatibility," [Online]. Available: https://docs.unity3d.com/412/Documentation/ScriptReference/MonoCompatibility.html. [Geopend 27 09 2016].

[73] Unity Technologies, „OPC UA .NET SDK code don't run in Unity player and windows build," 30 09 2016. [Online]. Available: https://forum.unity3d.com/threads/opc-ua-net-sdk-code-dont-run-in-unity-player-and-windows-build.434086/. [Geopend 21 11 2016].

[74] Automated Solutions, Inc., „AutomatedSolutions.ASComm.chm," Automated Solutions, Inc., Santa Rosa, CA, 2016.

[75] Automated Solutions, Inc., „END USER LICENSE AGREEMENT FOR AUTOMATED SOLUTIONS ASComm.NET SOFTWARE,” Automated Solutions, Inc., 2016. [Online]. Available: http://automatedsolutions.com/license/agreements/ascomm/default.asp. [Geopend 26 10 2016].

[76] F. I. Vokolos en E. J. Weyuker, „Performance Testing of Software Systems,” ACM, Santa Fe, New Mexico, 1998.

[77] Unity Technologies, „Execution Order of Event Functions,” Unity Technologies, 2016. [Online]. Available: https://docs.unity3d.com/Manual/ExecutionOrder.html. [Geopend 22 11 2016].

[78] Modicon, Inc., „Modicon Modbus Protocol Reference Guide,” Modicon, Inc., 06 1996. [Online]. Available: http://modbus.org/docs/PI_MBUS_300.pdf. [Geopend 13 12 2016].

18 Figurenlijst

19 Tabellenlijst