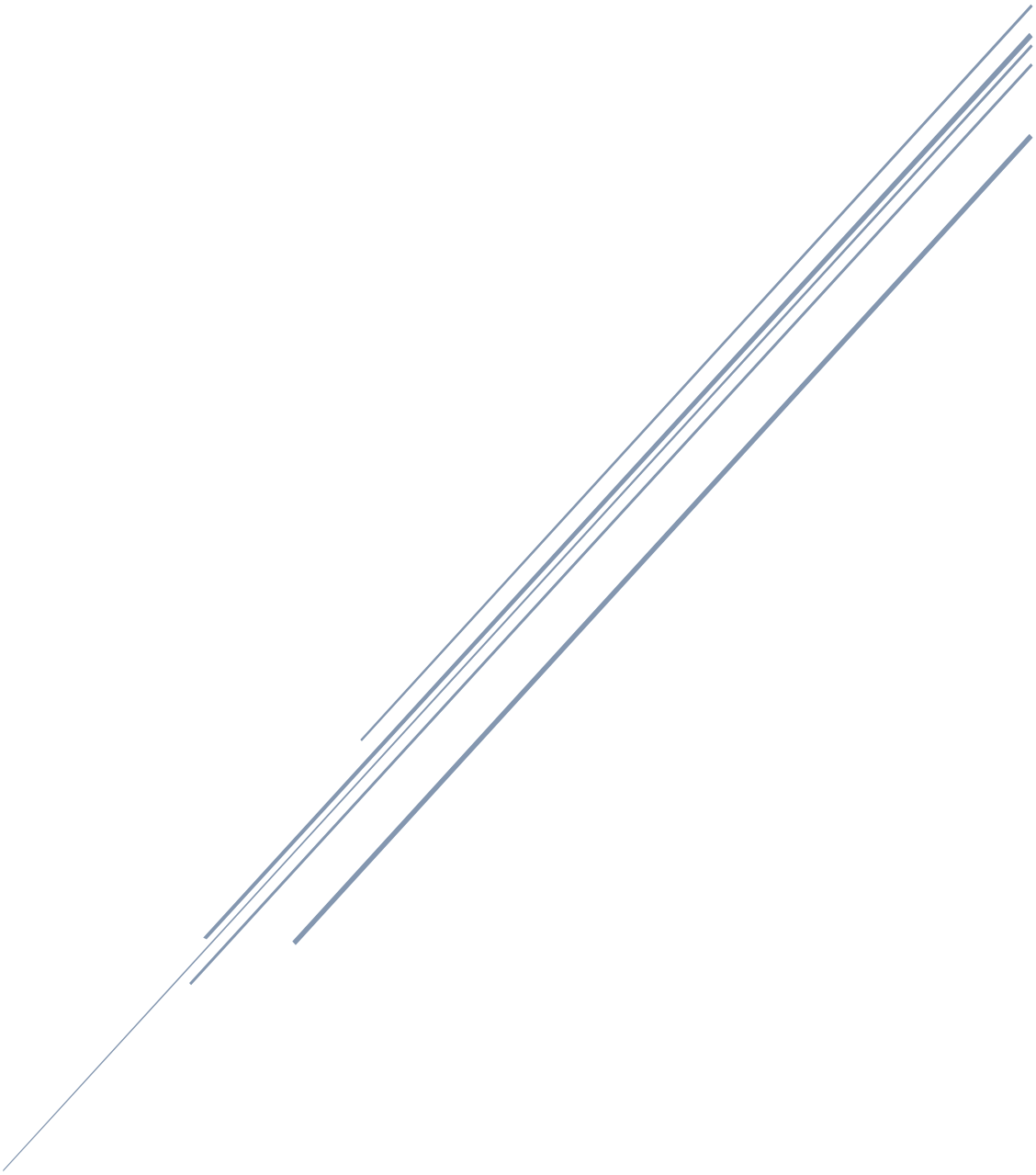
jb systems
industrial automation

BIJLAGENBOEK

Communicatie-architectuur PLC's en Unity

22-12-2016



Jordy Scholte
Haagse Hogeschool Delft

Inhoudsopgave

1 Bijlage A: Afstudeerplan

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2016-2.1 (start uiterlijk 29 augustus 2016)
Startdatum uitvoering afstudeeropdracht: 29-08-2016
Inleverdatum afstudeerdossier volgens jaarrooster: 22 december 2016

Studentnummer: 12019763
Achternaam: dhr Scholte (*) *weghalen niet van toepassing*
Voorletters: J.L.
Roepnaam: Jordy
Adres: Schimmelpenninckstraat 55
Postcode: 3119TP
Woonplaats: Schiedam
Telefoonnummer: 06-18958513
Mobiel nummer:
Privé emailadres: jordy@scholteweb.nl

Opleiding:
Locatie: Delft (*) *weghalen niet van toepassing*
Variant: voltijd

Naam studieloopbaanbegeleider: A. Andrioli
Naam begeleidend examinator: W.F.C. Wieringa
Naam tweede examinator: J. van Peski

Naam bedrijf: JB Systems
Afdeling bedrijf: R&D
Bezoekadres bedrijf: Westlandseweg 190
Postcode bezoekadres: 3131HX
Postbusnummer:
Postcode postbusnummer:
Plaats: Vlaardingen
Telefoon bedrijf: +31 (0)10 460 80 60
Telefax bedrijf: +31 (0)10 460 80 00
Internetsite bedrijf: www.jbsystems.nl

Achternaam opdrachtgever: dhr Vaandrager (*) *weghalen niet van toepassing*
Voorletters opdrachtgever: M.
Titulatuur opdrachtgever:
Functie opdrachtgever: R & D Engineer
Doorkiesnummer opdrachtgever: 06-25290435
Email opdrachtgever: mva@jbsystems.nl

Achternaam bedrijfsmentor: dhr Vaandrager (*) *weghalen niet van toepassing*
Voorletters bedrijfsmentor: M.
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor: R & D Engineer
Doorkiesnummer bedrijfsmentor: 06-25290435
Email bedrijfsmentor: mva@jbsystems.nl

NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder:

Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Ontwikkelen van een communicatie-architectuur tussen PLC's en simulaties

Opdrachtomschrijving *(toelichtende tekst verwijderen)*

1. Bedrijf

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 zal worden uitgevoerd op de afdeling R&D van de vestiging Vlaardingen, één van de drie vestigingen in Nederland.

2. Probleemstelling

JB Systems ontwikkelt 3D simulaties van op te leveren installaties. Deze worden gebruikt om te tonen aan klanten, voor trainingen, troubleshooting of testing. Voor deze doeleinden kan het ook nuttig zijn om deze simulaties op diverse manieren te laten interacteren met fysiek materieel, met name PLC's. Op dit moment wordt daarbij gebruik gemaakt van OPC en WinMOD. Dit blijkt echter foutgevoelig en dus instabiel te zijn. Het gevoel heerst dat dit wordt veroorzaakt door de diverse tussenstappen die nodig zijn bij deze oplossingen.

3. Doelstelling van de afstudeeropdracht

Het doel van de opdracht is dat de 3D simulatiesoftware (Unity) zodanig met Siemens PLC's kan communiceren dat de virtuele apparatuur zich gedraagt als de fysieke apparatuur, en dat manipulaties aan fysieke hardware terug te zien zijn in de virtuele hardware en vice versa. Dit moet betrouwbaarder en sneller werken dan het nu doet.

4. Resultaat

Na het voltooien van de opdracht is er een architectuur beschikbaar waarmee de simulatiesoftware kan communiceren met Siemens PLC's. Met deze architectuur wordt een proof-of-concept ontwikkeld. Deze architectuur moet stabieler zijn dan de huidige oplossingen. Indien er meerdere oplossingen worden gevonden of ontwikkeld moeten hiervan de voor- en nadelen worden bestudeerd en gedocumenteerd.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

Oriëntatiefase:
Doorlooptijd: 1 week
Mijlpaal: Plan van Aanpak
Activiteiten:
- Verhelderen van de opdracht
- Risico's analyseren
- Planning opstellen

Definitiefase:
Doorlooptijd: 5 weken
Mijlpaal: Definitiestudie
Activiteiten:

- Inlezen in aanwezige documentatie
- Analyseren van de huidige situatie
    - Huidige problemen analyseren
    - Analyseren stabiliteit
    - Analyseren snelheid
    - Analyseren schaalbaarheid
- Verbeterpunten in kaart brengen
- Bepalen systeemeisen

Ontwerpfase:

Doorlooptijd: 5 weken
Mijlpaal: Ontwerprapport
Activiteiten:

- Uitwerken van één of meerdere alternatieven
- Advies geven over alternatieven
- Bepalen van de oplossingsrichting in overleg met opdrachtgever
- Ontwerpen van nieuwe architectuur

Realisatiefase:

Doorlooptijd: 6 weken
Mijlpaal: Test- en implementatierapportage
Activiteiten:

- Verkrijgen van software, licenties etc.
- Installatie en configuratie van verschillende componenten
- Testen
- Overdracht aan opdrachtgever

6. Op te leveren (tussen)producten

- Plan van aanpak
- Definitiestudie
- Ontwerprapport
- Testrapportage
- Implementatiehandleiding

7. Te demonstreren competenties en wijze waarop

G1 (Praktische aspecten hanteren in (internationale) projecten)
Deze beroepstaak wordt aangetoond in het plan van aanpak door middel van de planning en de risicoanalyse.

A1 (Analyseren van het probleemdomein)
Deze beroepstaak wordt aangetoond aan de hand van een analyse van de huidige situatie in de definitiestudie.

C10 (Ontwerpen van een systeemarchitectuur)
Deze beroepstaak wordt aangetoond aan de hand van het ontwerprapport

.

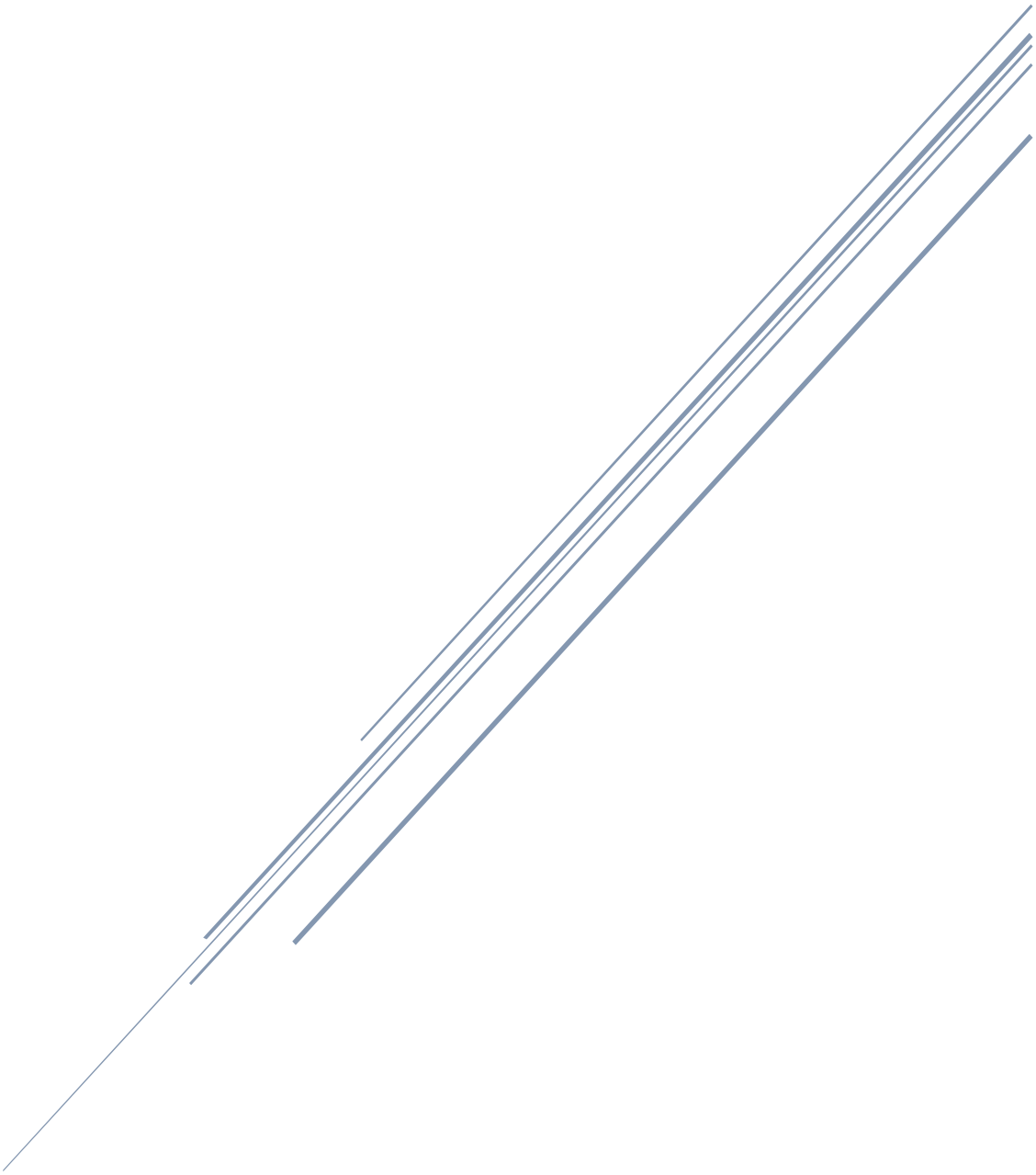
D18 (Testen van software systemen)
De juiste en verbeterde werking van het systeem zal worden aangetoond in het testrapport.

2 Bijlage B: Plan van Aanpak

PLAN VAN AANPAK

Communicatie-architectuur PLC's en Unity

22-12-2016



Jordy Scholte
Haagse Hogeschool Delft

Versiemanagement

Versie	Datum	Wijzigingen	Reviewer
1.0	2016-09-01	Initiële versie	M. Vaandrager

Inhoudsopgave

3 Inleiding

Dit plan van aanpak is geschreven naar aanleiding van de afstudeeropdracht bij JB Systems Automatisering. Deze stage is een onderdeel van de opleiding Technische Informatica aan de Haagse Hogeschool te Delft.

JB Systems is gespecialiseerd in het ontwikkelen van automatiseringsoplossingen voor klanten in de machinebouw, offshore en procesindustrie. De duur van de afstudeeropdracht is 17 weken. De opdracht is als volgt geformuleerd: “Ontwikkel een communicatie-architectuur tussen PLC’s en 3D-simulaties in Unity”.

In dit document zal de aanpak van het afstudeerproject worden beschreven. Om te beginnen is er in het eerste hoofdstuk terug te vinden wat deze opdracht inhoudt. Hieronder vallen de aanleiding, probleemstelling en doelstelling. Daarna is in het tweede hoofdstuk beschreven wat de gewenste resultaten en producten zijn.

Deze geven een gedetailleerder beeld van de opdracht met wat er vooral wel en niet onder de opdracht valt. Daarna wordt beschreven welke methode er zal worden gebruikt bij het onderzoek en het realiseren van het proof-of-concept.

Bij het opstellen van het afstudeerplan zijn er beroepscompetenties geformuleerd waarvan in het uiteindelijke afstudeerplan moet worden bewezen dat aan deze competenties is voldaan. Deze zijn, met verdere toelichting, beschreven in hoofdstuk 8.

Tevens wordt er in de risicoanalyse bekeken welke onvoorziene zaken de voortgang van het project kunnen belemmeren en hoe deze kunnen worden voorkomen of worden verkleind.

Tot slot is de planning voor de komende zeventien weken opgesteld. Hierin is in een timeline terug te vinden wat er per week en sprint moet gebeuren en wanneer de opleverdeadlines zijn. Ook de data van de opleiding zijn hier, voor zover bekend, in terug te vinden. Al deze data zijn onderverdeeld in fases, welke zijn onderverdeeld in sprints.

4 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.

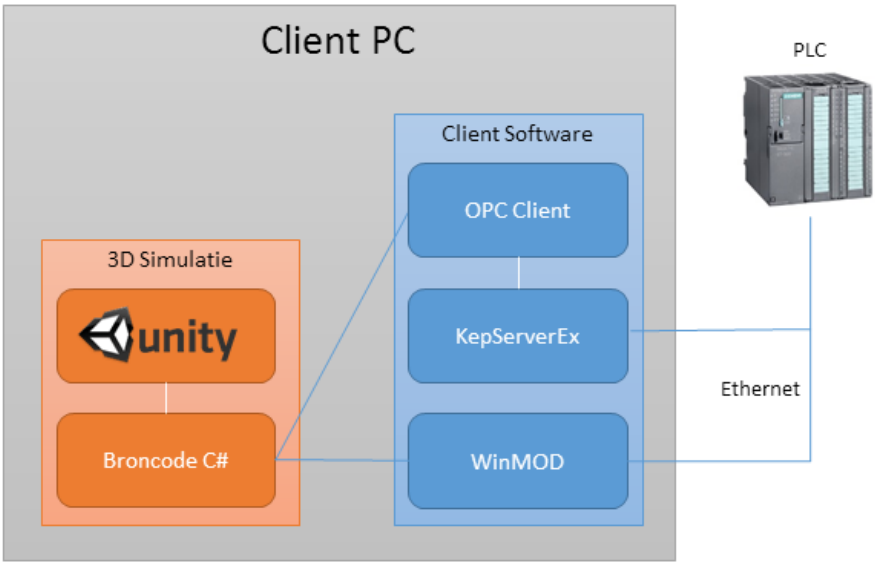
4.1 Aanleiding en huidige situatie

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.

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 gebuikte 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 1. Hierin is een onderverdeling gemaakt tussen de simulatie ("3D Simulatie") en de losse softwarepakketten ("Client Software").



Figuur 1: Schematische weergave huidige situatie

4.2 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.

4.3 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.

4.4 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 om welke geen last heeft van deze problemen. De onderzoeksvraag kan dus als volgt worden 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.

5 Afbakening en randvoorwaarden

De afstudeer periode duurt in zijn geheel 17 weken. Per week wordt er gemiddeld 40 uur aan de opdracht gewerkt. Dit houdt dus in dat het totaal aantal besteedde uren aan de opdracht rond de 680 ligt.

Zoals gesteld in het vorige hoofdstuk moet er worden gezocht naar een methode waarmee de communicatie tussen Unity-simulaties en PLC's kan worden verzorgd. Een volledig nieuwe architectuur is echter niet verplicht. In het mogelijke, doch onwaarschijnlijke geval dat de huidige situatie de beste blijkt, is dat ook een voor de opdrachtgever acceptabele conclusie, mits goed onderbouwd.

In de scope van het project bevinden de simulatie PC en de PLC zich binnen dezelfde fysieke ruimte, op hetzelfde netwerk. Dit netwerk mag zo eenvoudig mogelijk worden opgezet, tenminste bestaande uit een PC, een PLC, een netwerkswitch en bekabeling. De simulaties zelf worden gerenderd in Unity; Dit is ook een voorwaarde waar niet vanaf te wijken is.

Het onderzoek en implementatie zullen plaatsvinden op twee locaties van JB Systems: Het hoofdkantoor te Vlaardingen en het kantoor te Delft. De dagen waarin deze locaties worden afgewisseld zal voor het grootste deel samenlopen met de agenda van de opdrachtgever.

6 Op te leveren (tussen)producten

In de loop van de afstudeerperiode worden er een aantal producten opgeleverd. Deze lijst staat niet vast en hier kan in de loop van het project nog van worden afgeweken.

1. Plan van aanpak, met daarin onder andere een opdrachtomschrijving, voorlopige planning en risicoanalyse
2. Een definitiestudie/eisendocument, met daarin de in samenspraak met de opdrachtgever vastgestelde eisen en wensen
3. Een ontwerprapport, met daarin onder andere een overzicht van gevonden methoden, oplossingen en mogelijkheden, waarmee een aanbeveling kan worden gedaan
4. Eén of meerdere werkende proof-of-concepts
5. Testresultaten en –rapportage, met daarin de resultaten van de vergelijking tussen de nieuwe en de oude situaties
6. Een implementatiedocument/-handleiding met daarin de belangrijkste stappen en handvatten om de gekozen implementatie door de voeren in andere projecten
7. Een procesverslag van de volledige afstudeeropdracht

7 Ontwikkelmethodiek

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 SCRUM

Aan de hand van deze projectkenmerken zijn een aantal eisen opgesteld om een ontwikkelmethode te kiezen. In Figuur 2 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 2: Selectiematrix methodieken

Met behulp van deze matrix is de keuze gevallen op SCRUM. Deze methode heeft veruit de meeste plusjes.

7.1 Scrum

Scrum is een Agile ontwikkelmethode die werkt met sprints. Deze sprints duren 1 tot 4 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 in te passen in de huidige situatie waar één persoon de ontwikkeling op zich neemt. Hieronder volgen de aanpassingen welke worden 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.
3. De verschillende binnen Scrum gedefinieerde rollen worden door de afstudeerder uitgevoerd. De opdrachtgever kan in dit geval worden gezien als product-owner.

Met deze aanpassingen moet het geen probleem zijn om binnen dit relatief kleine project Scrum toe te passen.

Om het Scrum-proces te beheren zal er gebruik gemaakt gaan worden van de software Icescrum. Dit is een web-based applicatie waarmee de voortgang van het project, de backlogs en de sprints kunnen worden bijgehouden. Er is overwogen om een algemeen projectmanagementsysteem als Redmine te gebruiken, echter zal een tool als Icescrum naar verwachting meer dwingen om de Scrummethode nauwgezet te volgen.

8 Beroepstaken en competenties

Bij het opstellen van het afstudeerplan zijn er een aantal beroepscompetenties opgenomen waar aan het einde van het traject aan moet worden voldaan.

NR.	**Titel, manier van aantonen en relevantie**
G1	**Praktische aspecten hanteren in (internationale) projecten**
	Deze beroepstaak wordt aangetoond in het plan van aanpak door middel van de planning en de risicoanalyse.
	Deze competentie is relevant omdat de ontwikkeling van simulaties een continu proces is. Hierbij moet rekening gehouden worden met wensen van andere personen en projectgroepen. Daarnaast kunnen de resultaten van de opdracht grote gevolgen hebben voor deze projecten.
A1	**Analyseren van het probleemdomein**
	Deze beroepstaak wordt aangetoond aan de hand van een analyse van de huidige situatie in de definitiestudie.
	Om te weten wat er moet gebeuren, moet eerst worden bepaald wat de huidige problemen precies zijn en waardoor deze worden veroorzaakt.
C10	**Ontwerpen van een systeemarchitectuur**
	Deze beroepstaak wordt aangetoond aan de hand van het ontwerprapport.
	Deze competentie is relevant omdat het doel van de opdracht een verbeterde architectuur is
D18	**Testen van software systemen**
	De juiste en verbeterde werking van het systeem zal worden aangetoond in het testrapport.
	Om te kunnen bewijzen dat de nieuwe architectuur een verbetering is ten opzichte van de oude, zullen er kwantitatieve tests moeten worden bedacht en uitgevoerd die bijvoorbeeld snelheidswinst of hogere stabiliteit laten zien.

9 Risicoanalyse

In dit hoofdstuk worden de risico's die zich tijdens dit project kunnen voordoen behandeld. Daarbij wordt aangegeven wat de kans en impact zijn van de betreffende situatie. Aan de hand hiervan wordt elk risico met een kleur en cijfer ingeschaald. In Tabel 1 is de zien wat deze kleuren inhouden. Hoe hoger het cijfer, hoe meer gevaar het betreffende risico oplevert mocht deze zich voordoen.

Risicoschaal				
Kans	Groot	3	6	9
	Middel	2	4	6
	Klein	1	2	3
		Klein	Middel	Groot
		Impact		

Tabel 1: Risicoschaal

Risico	**Niet alle aspecten van de opdracht zijn helemaal duidelijk**	
Gevolg	Bepalen van haalbaarheid is moeilijk of niet mogelijk	
Kans		Middel
Kans vermindering	Vroeg in het project zorgen dat alle eisen duidelijk zijn	
Impact		Middel
Impactvermindering	Kiezen van de juiste ontwikkelmethode	
Plan B		

Risico	**Best werkende methode is te duur**	
Gevolg	Opdrachtgever gaat niet akkoord met voorstel	
Kans		Klein
Kans vermindering	Van te voren dure software en hardware proberen te vermijden, zodat er achteraf geen verassingen ontstaan	
Impact		Middel
Impactvermindering		
Plan B	Terugvallen op de één na beste oplossing	

Risico	**Doel wordt te snel/gemakkelijk behaald**	
Gevolg	Te veel tijd over, waarschijnlijk onvoldoende werk voor een goede beoordeling	
Kans		Klein
Kans vermindering	Goede afbakening opstellen en tijdig overleg plegen	
Impact		Groot
Impactvermindering	Tijdig aangeven bij betreffende personen en naar oplossingen zoeken	
Plan B	Aanvullen van opdracht met nieuwe deelopdrachten	

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	

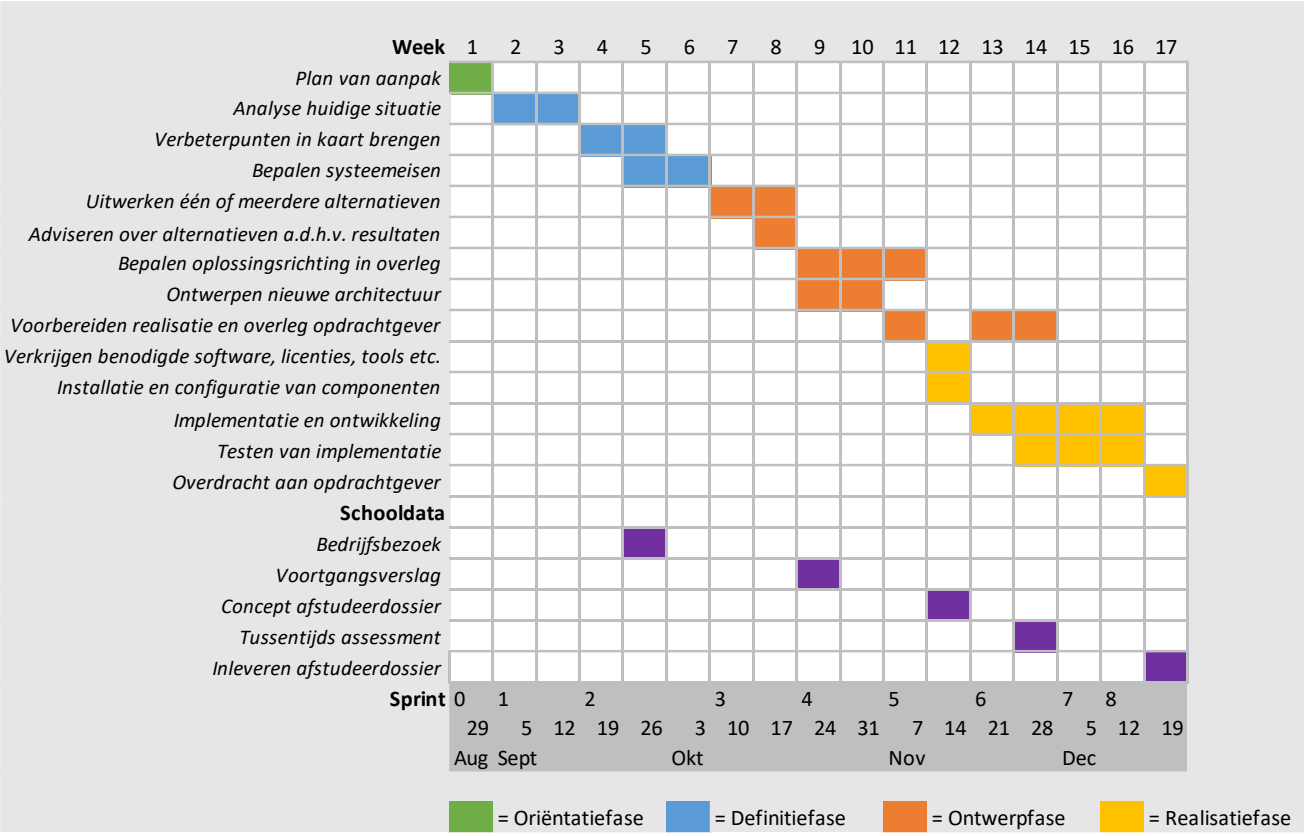
10 Planning

Voor de komende zeventien weken is een initiële planning gemaakt. In Figuur 3 is een timeline van deze planning te zien. Het project is onderverdeeld in vier fasen: De oriëntatiefase, definitiefase, ontwerpfase en realisatiefase. Deze zijn met kleurcoderingen herkenbaar gemaakt in de timeline.

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 over het algemeen twee tot drie weken.

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 niet bekend. Daarom zijn de verwachte weken waarin deze data zullen vallen aangegeven.



Figuur 3: Planning in een timeline

Het inleveren van het afstudeerdossier vindt plaats in de laatste week op uiterlijk 22 December 2016.

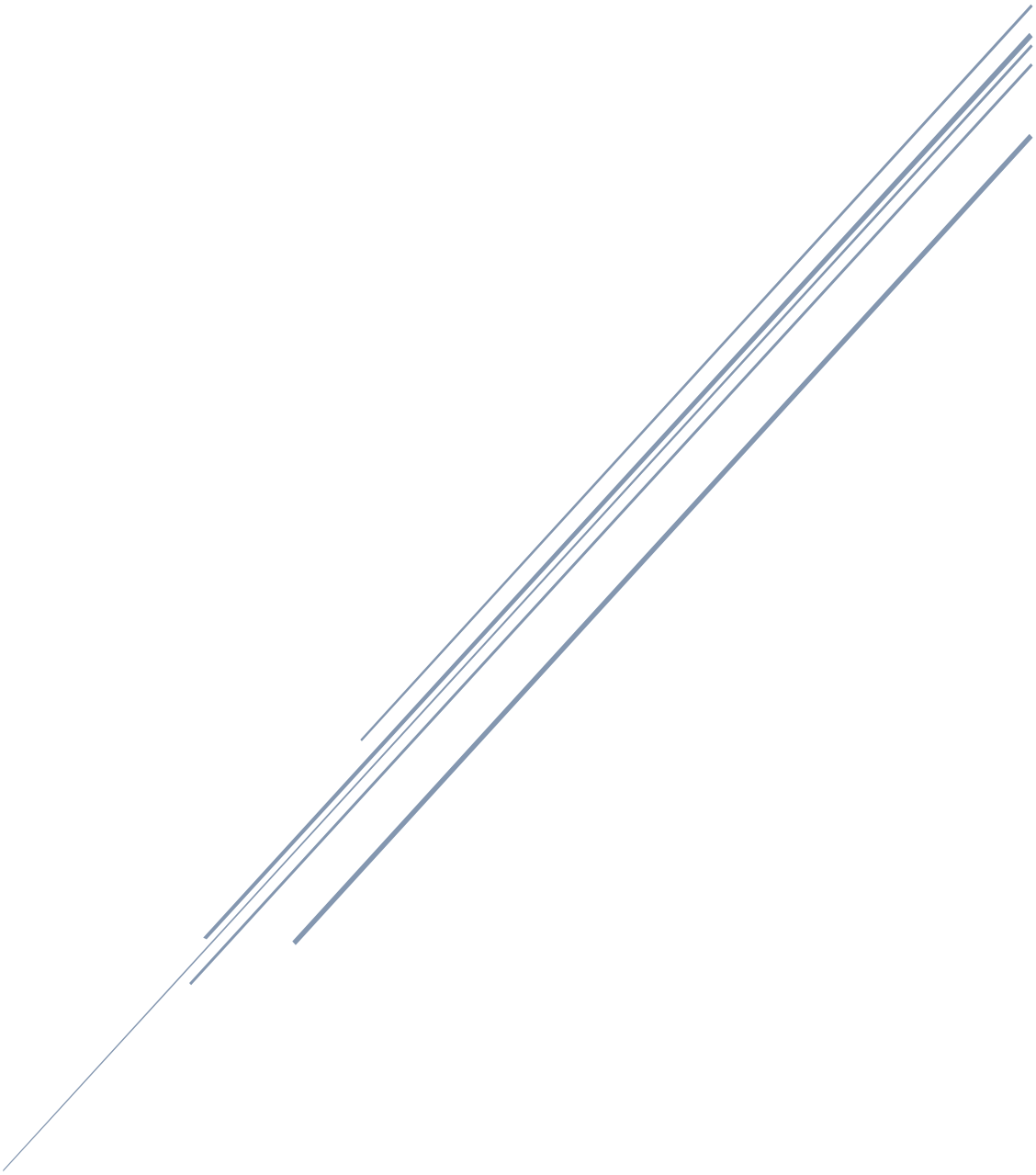
3 Bijlage C: Definitiestudie

jb systems
industrial automation

DEFINITIESTUDIE

Communicatie-architectuur PLC's en Unity

22-12-2016



Jordy Scholte
Haagse Hogeschool Delft

Inhoudsopgave

1 Inleiding

In dit vooronderzoek is onderzoek gedaan naar hoe de op dit moment gebruikte communicatietechnieken opgezet zijn. Daarna zijn een aantal hoofdeisen opgesteld waaraan alternatieve oplossingen moeten voldoen. Ze zijn opgesteld in samenspraak met de opdrachtgever en kunnen worden gebruikt om gevonden communicatiemogelijkheden bij voorbaat uit te sluiten. Het is goed mogelijk dat deze eisen nog veranderen, omdat het op dit moment nog niet duidelijk is of er een communicatietechniek bestaat die aan alle eisen voldoet.

In een later stadium zullen de detaileisen worden bepaald. Tevens bevat dit document een overzicht van reeds gevonden mogelijkheden. Aan de hand van aanwezige documentatie zijn deze vergeleken met de gestelde eisen, en is een overzicht gemaakt waarin te zien is hoe geschikt elk stuk software naar verwachting zal presteren.

In het verdere onderzoek zullen de prestaties van de communicatiemogelijkheden worden gekwantificeerd met behulp van een testopstelling.

De onderzoeksvraag welke ten grondslag ligt aan dit onderzoek is als volgt te omschrijven:

> *“Welke manieren van communicatie zijn mogelijk tussen Unity3D en een PLC?”*

Hieruit valt het volgende doel op te maken:

> *“Creëer een overzicht van alle communicatiemogelijkheden tussen Unity3D en een PLC. Ontwerp en implementeer een beter alternatief voor de bestaande oplossing”*

4 Analyse Huidige Situatie

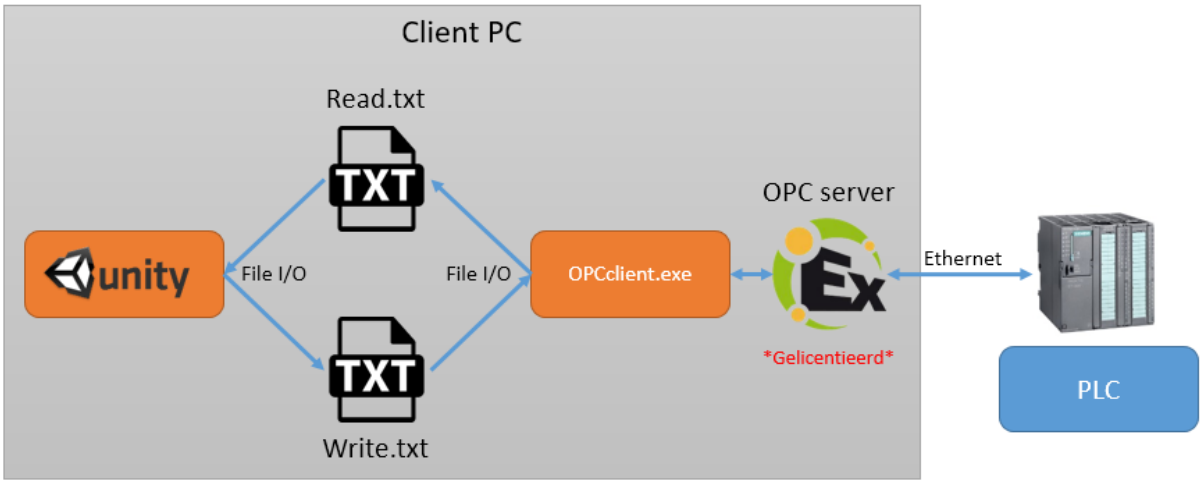
Voordat er kan worden bepaald wat de eisen voor nieuwe communicatiemogelijkheden in moeten houden moet er een definitieanalyse worden uitgevoerd. Hieronder valt een analyse van de huidige situatie en de problemen die daarbij komen kijken. Aan de hand van de hieruit voortkomende resultaten kan er gekeken 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. Vlak voor de opdracht is er ook, bij wijze van experiment, een keer gebruik gemaakt 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.

4.1 OPC DA

OPC is een industriële communicatiestandaard [1] 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 [2], 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 [3]. Deze library komt in de vorm van een DLL bestand. Het is echter niet gelukt deze direct in Unity te integreren. 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 4). Deze losse applicatie is vervolgens de OPC-client die gebruik maakt van de bovengenoemde DLL die communiceert met de OPC-server, in dit geval KepServerEX 5 [4].



Figuur 4: 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 [5]
3. Toevoegen OPC Client in PLC en tags 'openbaar maken'
4. OPC server aanmaken in KepServer en tags toevoegen
5. OPC client aanmaken die leest en schrijft van tekstbestanden
6. Unity laten lezen en schrijven van tekstbestanden

4.1.1 Voordelen

Het grootste voordeel van OPC is de wijdverbreide ondersteuning binnen de industrie, waardoor het samenwerkt met nagenoeg elk gangbaar merk en type PLC [6, 5, 7]. Als het eenmaal werkt, werkt het naar behoren. OPC werkt op dit moment voor de huidige toepassingen snel genoeg.

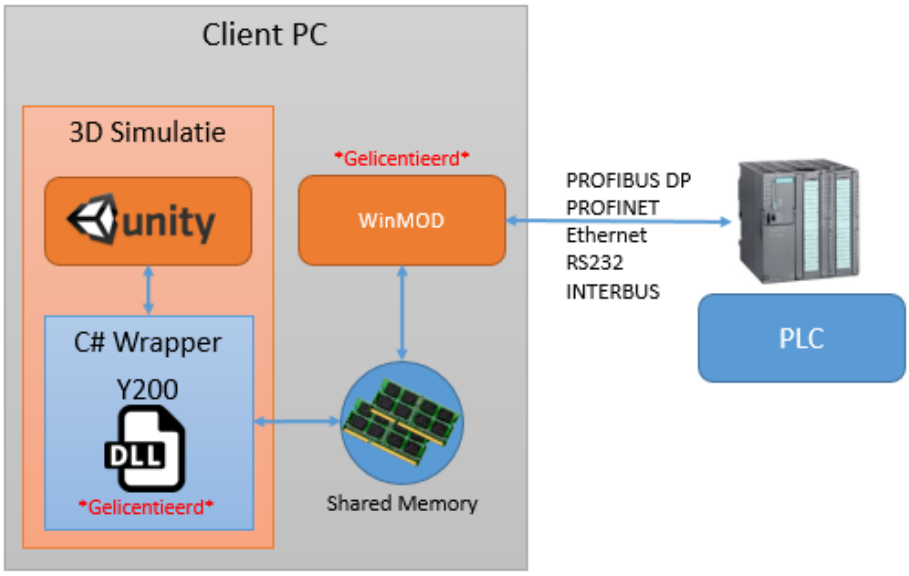
4.1.2 Nadelen

Het gevoel heerst binnen JB Systems dat het een hoop werk is om aan de praat te krijgen; Er zijn veel tussenstappen benodigd en het opzetten van de Server en het instellen van de PLC neemt ook tijd in beslag. Daarnaast moeten er voor de server voor elk afzonderlijk merk PLC licenties aangeschaft worden [5]. Tevens leert de ervaring dat de OPC-oplossing vrij broos en gevoelig voor programmeerfouten is, en de algehele stemming is dat OPC het liefst niet voor dit doeleinde zou moeten worden gebruikt. Dat zou natuurlijk ook kunnen liggen aan het gebruik van de tussenapplicatie met tekstbestandjes welke gebruikt worden voor de OPC verbinding. Tot slot is OPC op het moment snel genoeg, maar is de snelheid nooit gekwantificeerd, en is het dus onduidelijk of het voor toekomstige projecten geschikt zou zijn.

4.2 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 [8, 9]. Voor WinMOD is een losstaande C++ DLL beschikbaar waarmee externe applicaties kunnen communiceren met WinMOD door middel van een shared memory driver [10]. 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 5: Schematische weergave WinMOD/Y200-communicatie

Voor het opzetten van dit systeem zijn grofweg de volgende stappen benodigd [11]:

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

4.2.1 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 elk denkbare methode [9, 11].

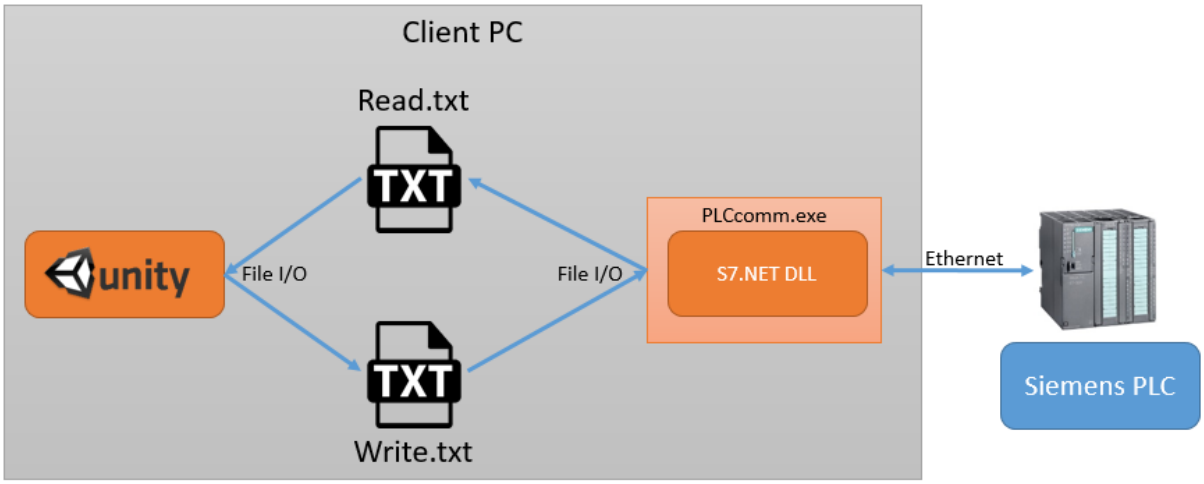
4.2.2 Nadelen

Het implementeren van WinMOD is vrij omslachtig. Allereerst is het een uitgebreid softwarepakket wat tijd kost om te configureren [10]. Dit is zelfs meer werk dan bij OPC. Ook het gebruik van een USB-dongle voor het overdragen van de licentie [12] wordt ervaren als een ongewenste extra handeling. Daarnaast is de prijs een hekel punt: Een licentie loopt al gauw richting de €14.000,- wat voor kleinere projecten vaak niet is te verantwoorden.

4.3 S7.NET

S7.NET is een gratis verkrijgbare opensource library [13] waarmee het Siemens S7 protocol [14, 15] 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 [16]. 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 [17], maar is op dit moment nog niet bruikbaar. Hier is daarom tevens gebruikt 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 6) .



Figuur 6: Schematische weergave S7.NET-communicatie

4.3.1 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. Vanwege de overbodigheid van het vertalen van en naar andere protocollen zou het in theorie de snelste manier van communiceren moeten zijn.

4.3.2 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 [13]. Het is daarom niet zeker dat eventuele nieuwe types PLC's zullen worden ondersteund, hoewel het S7-protocol waarschijnlijk niet snel door Siemens zal worden vervangen. 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.

4.4 Conclusie Analyse

De huidige gebruikte communicatiemogelijkheden hebben allemaal gemeen dat ze uit meerdere tussenstappen bestaan. Dit brengt extra complexiteit met zich mee, wat op zijn beurt de kans op bugs, fouten en problemen vergroot [18, 19, 20, 21]. 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.

Daarnaast voegen ook licenties extra complexiteit aan het geheel toe. Dit is met name merkbaar bij WinMOD, waar een licentie volgens de opdrachtgever al gauw €14.000,- kost en moet worden gewerkt met een usb-dongle om de licentie te activeren [12]. Daarnaast is het wel eens voorgekomen dat een gelicentieerde applicatie besloot dat de licentie niet meer geldig was. Hierdoor worden licenties beschouwd als een point-of-failure binnen een systeem. Tevens zijn licenties niet altijd even makkelijk schaalbaar, omdat het hele traject van aanschaf, installatie en configuratie voor elke simulatie weer opnieuw moet worden uitgevoerd.

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.

5 Hoofdeisen

In overleg met de opdrachtgever zijn er een aantal eisen opgesteld. Deze top-level eisen voor de op te leveren architectuur zijn als volgt:

1. Snelheid van communicatie = 100Hz, bij 100 Reals per datastroom, één richting
2. Oplossing mag niet duurder zijn dan huidige duurste mogelijkheid (WinMOD ~14€)
3. De gekozen oplossing moet in ieder geval werken met Siemens S7 PLC's, maar het is zeer gewenst dat deze ook compatible is met de bij JB Systems meest gebruikte merken.
4. De maximale latency van één datatransmissie mag niet meer dan 10ms bedragen
5. De software moet langer zonder vastlopers blijven draaien dan de huidige oplossingen
6. Nieuwe oplossingen moeten uit minder softwarecomponenten of –lagen bestaan dan de huidige, ter verlaging van de complexiteit van het geheel en de vermindering van kans op problemen.
7. De software moet vanuit Unity benaderbaar zijn (te integreren in C#/.NET applicatie)
8. Het is zeer gewenst om voor de het opzetten van de communicatie geen aanpassingen te hoeven doen aan de code op de PLC

Deze eisen zijn gebaseerd op de ervaringen van de opdrachtgever met de huidige communicatiepaden.

5.1 Afbakening

Naast de systeemeisen is er nog een stukje afbakening. Deze zijn al in het plan van aanpak uitgewerkt [22], maar de voor dit document relevante randvoorwaarde is inmiddels verder gedefinieerd.

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.

6 Bibliografie

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

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

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

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

[5] 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].

[6] 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].

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

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

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

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

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

[12] Mewes & Partner GmbH, „Licencing Contract," 01 01 2010. [Online]. Available: http://winmod.de/en/uploads/01_2010_WinMOD%20Lizenzvertrag%20Standard_engl.pdf. [Geopend 20 09 2016].

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

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

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

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

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

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

[19] 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.

[20] 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.

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

[22] J. Scholte, „PLAN VAN AANPAK Communicatie-architectuur PLC’s en Unity,” Vlaardingen, 2016.

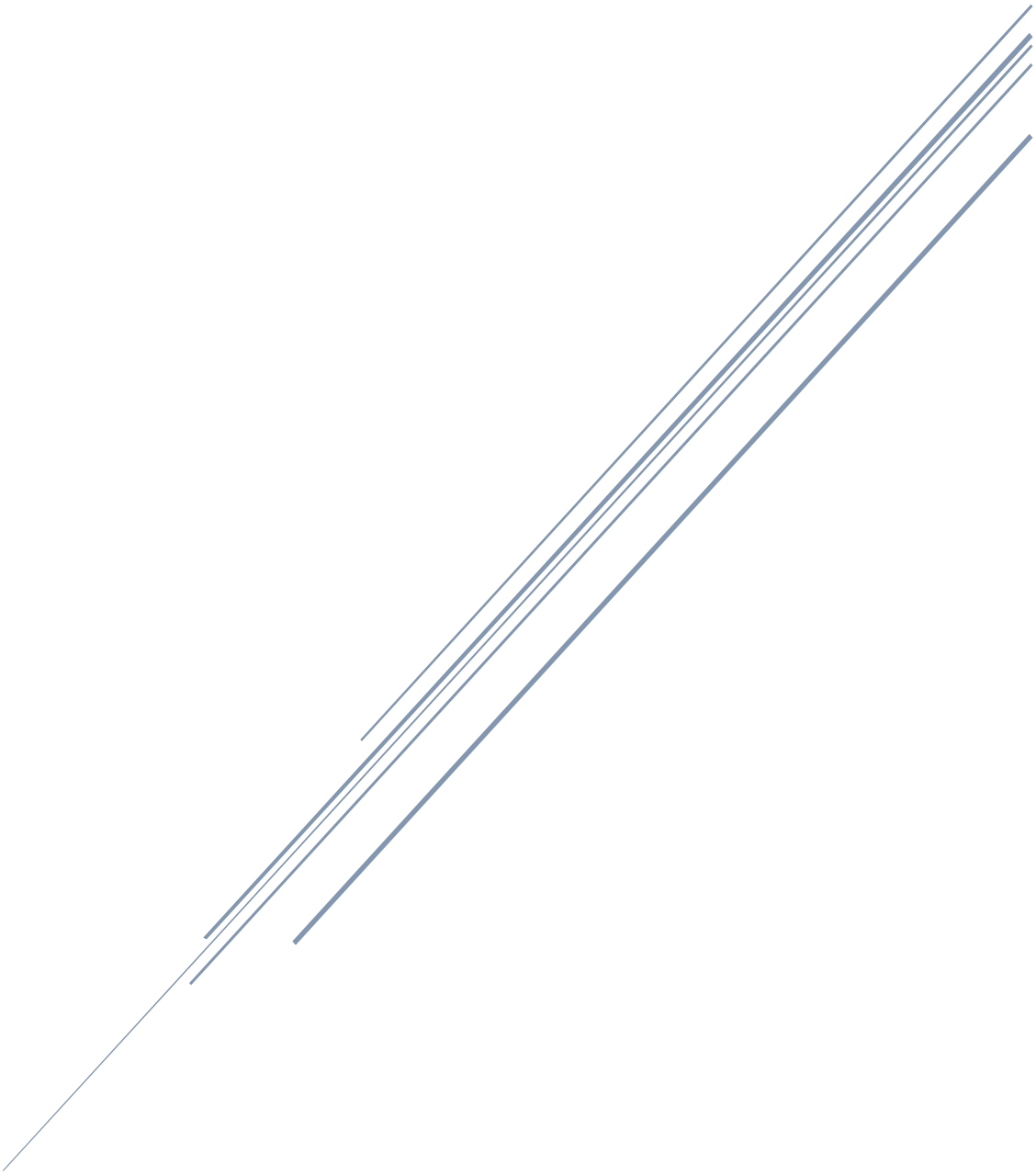
4 Bijlage D: Ontwerprapport

jb systems
industrial automation

ONTWERPRAPPORT

Communicatie-architectuur PLC's en Unity

22-12-2016



Jordy Scholte
Haagse Hogeschool Delft

Inhoudsopgave

1 Inleiding

Dit document bevat een overzicht van reeds gevonden mogelijkheden. Aan de hand van aanwezige documentatie zijn deze vergeleken met de gestelde eisen in de definitiestudie, en is er een overzicht gemaakt waarin te zien is hoe geschikt elk stuk software naar verwachting zal presteren.

Van elk deze mogelijkheden is verder uitgezocht hoe deze geïmplementeerd zouden kunnen worden in Unity3D. Dit wordt geïllustreerd aan de hand van schematische diagrammen, waarin onder andere onderdelen, afhankelijkheden en packages worden getoond.

In later onderzoek zullen de prestaties van de communicatiemogelijkheden worden gekwantificeerd met behulp van een testopstelling.

De onderzoeksvraag welke ten grondslag ligt aan dit onderzoek is als volgt te omschrijven:

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

Hieruit valt het volgende doel op te maken:

> *"Creëer een overzicht van alle communicatiemogelijkheden tussen Unity3D en een PLC. Ontwerp en implementeer een beter alternatief voor de bestaande oplossing"*

5 Eisen

In de Definitiestudie [1] zijn reeds een aantal hoofdeisen opgesteld. De communicatietechnieken die in dit document worden behandeld moeten hieraan voldoen. Deze hoofdeisen zijn verder uigesplitst in meer gedetailleerde eisen. Om een weging mee te geven aan de eisen zijn deze tevens voorafgaand aan het onderzoek ingedeeld volgens de MoSCoW-methode:

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.

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 2: Eisen

Deze eisen zijn gebaseerd op de ervaringen van de opdrachtgever met de huidige communicatiepaden.

5.1 Afbakening

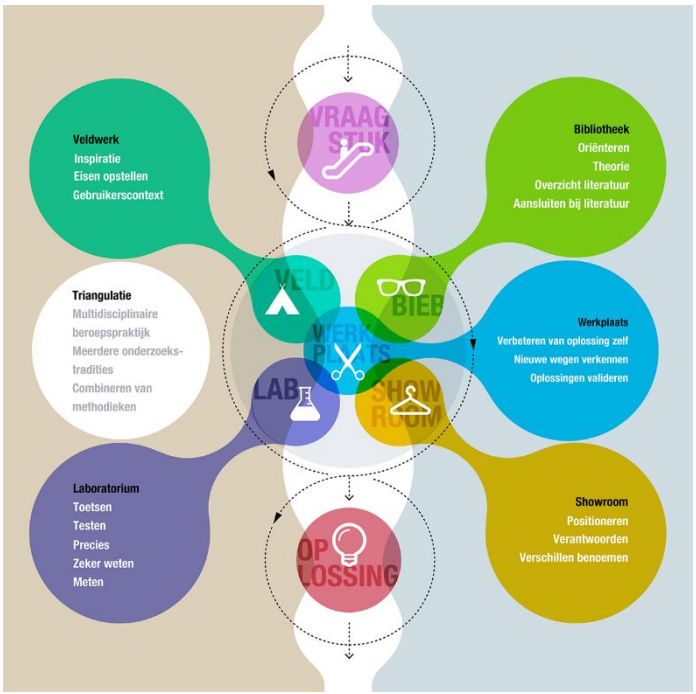
Naast de systeemeisen is er nog een stukje afbakening. Deze zijn al in het plan van aanpak uitgewerkt [2], maar de voor dit document relevante randvoorwaarde is inmiddels verder gedefinieerd.

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.

6 Literatuuronderzoek en oriëntatie

De onderzoeksaanpak is ingedeeld volgende de 'Methodenkaart in de beroepspraktijk van ICT en Media' [3]. Hierbij zijn de verschillende onderdelen van onderzoek opgedeeld in een aantal categorieën (Figuur 7).



Figuur 7: Methodenkaart onderzoek

In de zoektocht naar mogelijke oplossingen zijn een aantal bestaande onderzoeken gevonden die overlap hebben met- of lijken op 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 [4, 5, 6], Virtuele PLC's of HMI's [4, 7] of is er geen gedetailleerde beschrijving van de gebruikte technieken [8].

Vanwege gebrek aan literatuur is het nodig om een eigen onderzoek uit te voeren naar alternatieve mogelijkheden om een PC-applicatie toegang te verlenen tot het geheugen van een PLC. Hieruit zijn een aantal software libraries naar voren gekomen welke kunnen worden geïmplementeerd in eigen code. Deze libraries zijn voor het merendeel ontwikkeld door personen of communities welke tegen dezelfde problemen zijn aangelopen 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.

Van alle resultaten uit dit onderzoek is direct een shortlist opgesteld. Zo zijn libraries die bij voorbaat al aan te weinig eisen voldeden genegeerd, en zijn libraries die niet binnen de afbakening pasten bij voorbaat al niet tegen de eisen afgewogen.

7 Gevonden oplossingen

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.

Nummer	Naam	Opmerking
1	Snap7 [9]	Alleen Siemens PLC's
2	Sharp7 [10]	C# port van Snap7
3	S7.NET+ [11]	Alleen Siemens PLC's
4	DotNetSiemensPLCToolBoxLibrary [12]	Alleen Siemens PLC's
5	CSP Ethernetdriver [13]	Oud Allen Bradley protocol, maar wordt voor backwards compatibility nog steeds ondersteund op nieuwere generaties.
6	AB Ethernet Protocol Driver [14]	Zelfde als 4, maar betaald en recenter geüpdatet, en bevat commerciële support.
7	OPC UA [15]	Server vereist, maar laatste KepServerEx versies (welke aanwezig is bij opdrachtgever) ondersteunen UA. Werkt met nagenoeg elke PLC. [16, 17, 18] Verbetering t.o.v. OPC DA.
8	Experior [19]	

Tabel 3: Gevonden communicatiemogelijkheden

Een aantal van deze oplossingen is door afstudeerder al in een eerder project gebruikt. Hoewel dit geen identieke situatie was, kan het interessant zijn om te refereren naar de resultaten die hieruit zijn voortgevloeid [20]. De hierbij gemeten snelheid is het resultaat van een cyclus van het versturen en ontvangen van 6 REALS. 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	DSC/Kepserver	40Hz
OPC	DSC/DataFEED server	52Hz
OPC UA	KepServer	72Hz
OPC UA	DataFEED server	84Hz
Modbus/TCP		48Hz
TCP/IP		≈200Hz[1]
SNAP7		268Hz

Tabel 4: Snelheidsindicatie per protocol

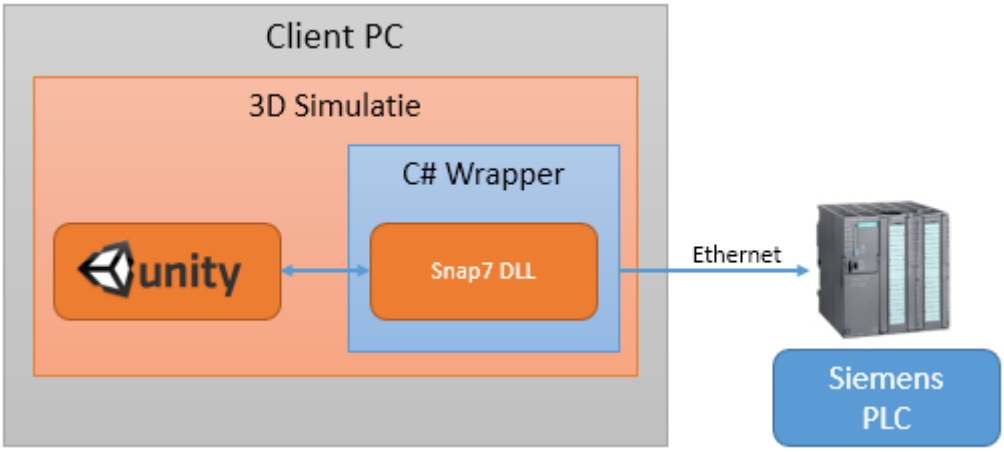
[1] Veel time-outs bij hogere snelheden; Effectief is de bruikbare snelheid niet hoger dan 200Hz

De gevonden oplossingen in Tabel 3 zijn aan de hand van beschikbare literatuur en documentatie geanalyseerd waaruit kan worden afgeleid hoe deze geïmplementeerd kunnen worden in de simulaties.

7.1.1 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.

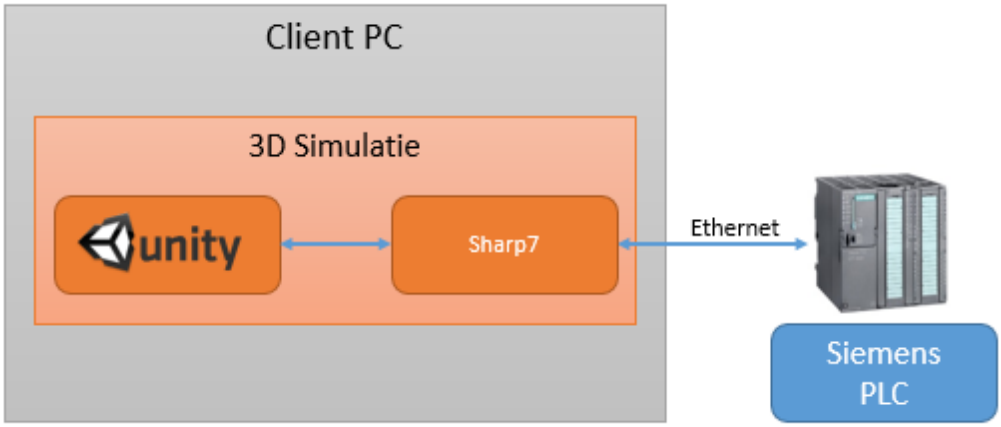
In Figuur 8 is te zien hoe de library wordt geïntegreerd in de applicatie en hoe het communicatiepad tussen simulatie en PLC verloopt.



Figuur 8: Schematische weergave Snap7-communicatie

7.1.2 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.

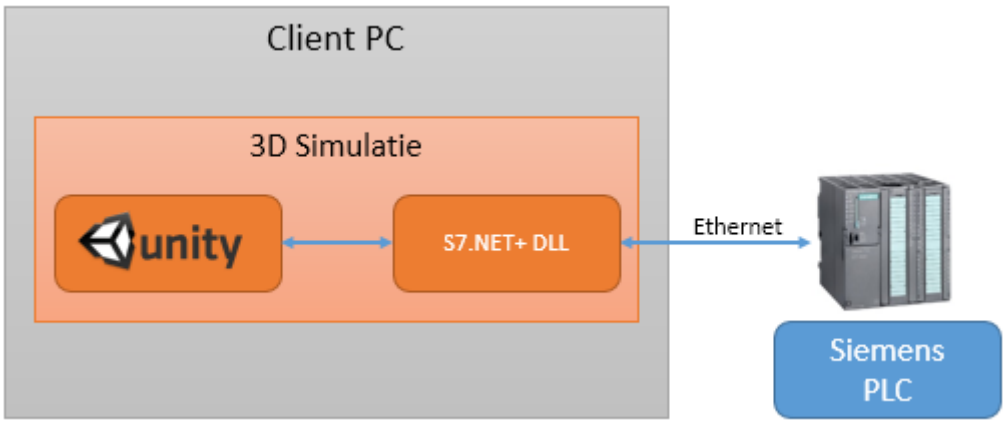


Figuur 9: Schematische weergave Sharp7

7.1.3 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 10 laat zien hoe deze library wordt geïntegreerd in de simulatie.

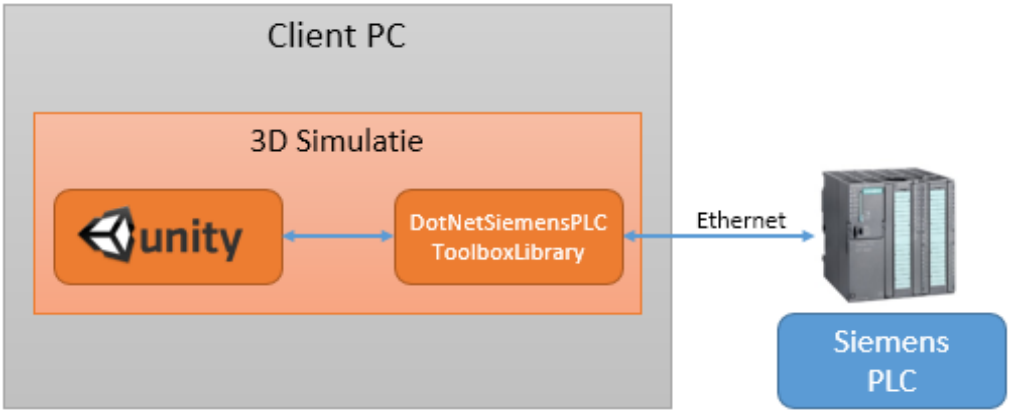
Uit de documentatie blijkt ook dat de mogelijkheden van S7.NET+ wat minder zijn dan die van Snap7, voornamelijk omdat er geen vergelijkbare partner- en serverfuncties aanwezig zijn. [21]



Figuur 10: Schematische weergave S7.NET+ communicatie

7.1.4 DotNetSiemensPLCToolBoxLibrary

De DotNetSiemensPLCToolBoxLibrary is een wat oudere C# library die ondersteuning levert voor Siemens S7-300/400 en S5 PLC's. Het, volgens de documentatie, ontbreken van nieuwere S7 PLC 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 11.



Figuur 11: Schematische weergave DotNetSiemensPLCToolBoxLibrary

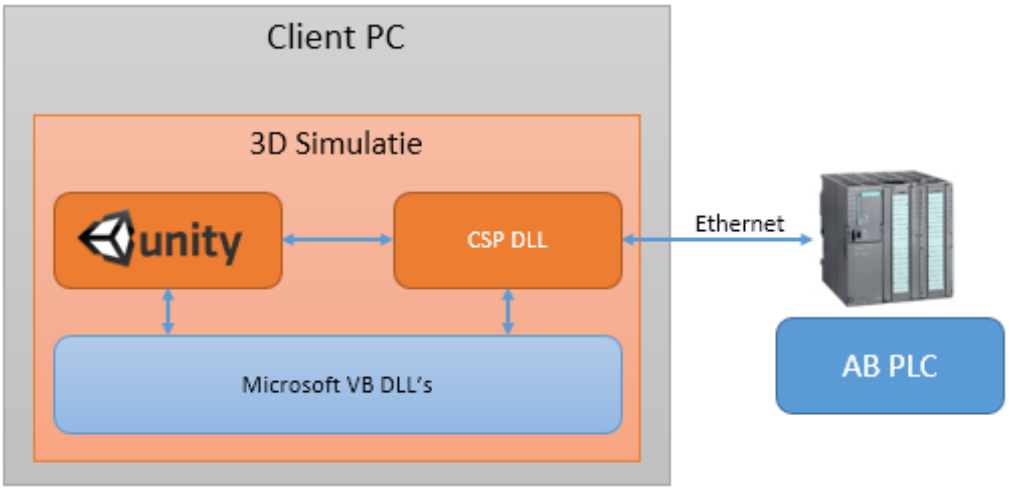
7.1.5 CSP Ethernetdriver

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 12.

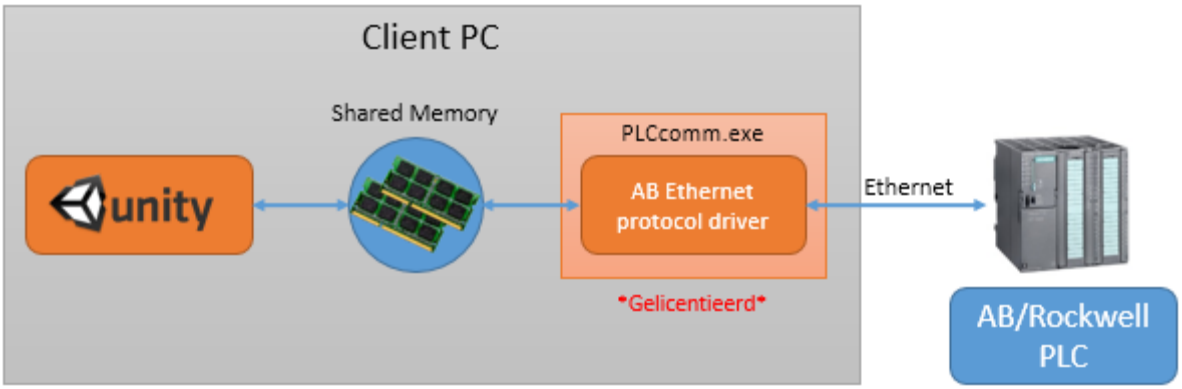


Figuur 12: Schematische weergave CSP Ethernetdriver

7.1.6 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 bijvoorbeeld een shared memory driver kan deze communiceren met de Unity simulatie. Dit zou leiden tot een constructie zoals getoond in Figuur 13.

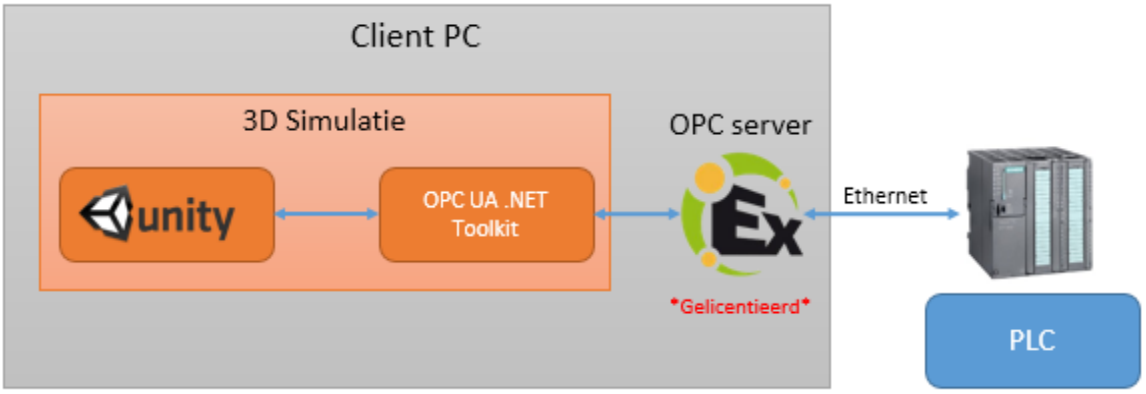


Figuur 13: Schematische weergave AB Ethernet Protocol

Omdat er voor AB/Rockwell nog andere alternatieven beschikbaar zijn (7.1.5, 7.1.7, 7.1.9) is deze library niet getest.

7.1.7 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.



Figuur 14: Schematiscche weergave OPC UA communicatie

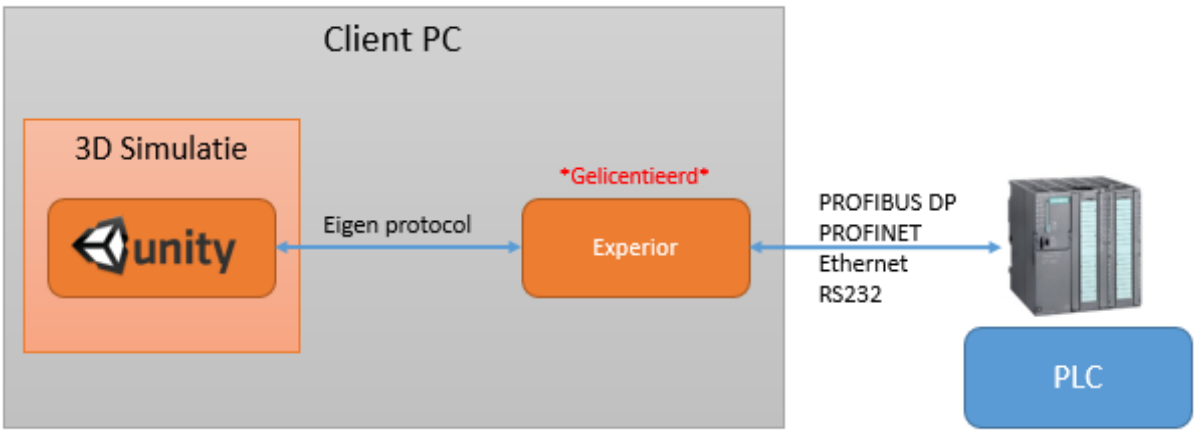
7.1.8 Experior

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 15). 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 eigenlijk direct afvalt.



Figuur 15: Schematische weergave Experior

7.1.9 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.

7.2 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 5. Dit is op basis van een eerste inschatting.

	Eis:	Snap7	S7.NET+	DotNet-Siemens	CSP Ethernet driver	OPC UA	NModbus4	Experior	ASComm.NET
1	*Snelheid*	+	+	O	O	O	O	+	+
2	*Prijs*	+	+	+	+	+	~	-	+
3	*Siemens PLC's*	+	+	~	+	+	~	+	+
4	*Gangbare PLC's*	-	-	-	-	+	~	+	+
5	*Alle PLC's*	-	-	-	-	+	-	+	+
6	*Jitter*	O	O	O	O	O	O	O	O
7	*Latency*	O	O	O	O	O	O	O	O
8	*Stabiliteit*	O	O	O	O	O	O	O	O
9	*Complexiteit*	+	+	+	+	+	+	-	+
10	*Unity3D*	+	+	+	-	-	+	+	
11	*Geen aanpassing PLC code*	+	+	+	O	~	O	+	+
12	*FPS*	+	+	+	+	+	+	+	+

+	Eis voldoet
-	Eis voldoet niet
~	Eis voldoet deels
O	Onbekend

Tabel 5: Mogelijkheden afgezet tegen eisen tegen eisen

De informatie om deze tabel samen te stellen komt voort uit de websites en documentatie van de betreffende libraries die hierboven staan gerefereerd. Voor sommige eisen, zoals 6, 7 en 8 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 4 en 5 zijn voornamelijk ingevuld aan de hand van 'educated guesses'. 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 onbekende zaken zullen in een latere sprint proefondervindelijk moeten worden vastgesteld.

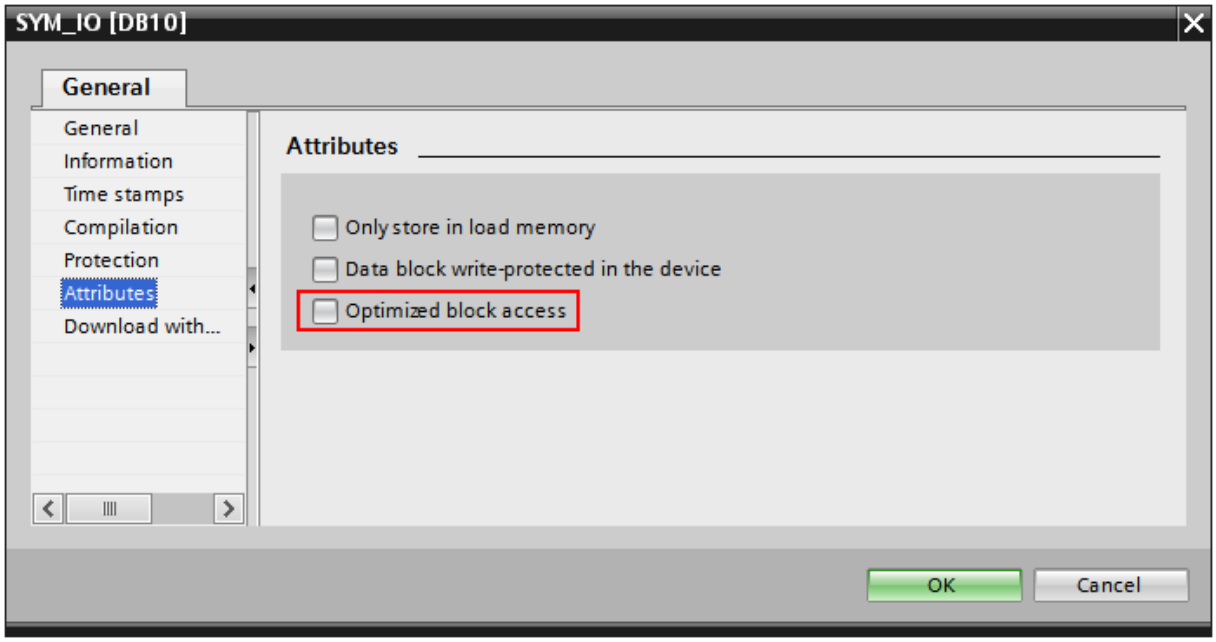
8 Eerste analyse libraries

De uitgekozen libraries zijn elk aan een kleine bestudering onderworpen. Hierbij is gekeken naar hoe de library moet worden geïntegreerd, waar deze uit bestaat en welke functies deze de gebruiker biedt. Ook zijn hier enkele zaken in opgenomen waar rekening mee gehouden dient te worden voordat de libraries gebruikt kunnen gaan worden.

8.1 Toegangsopties Siemens

Allereerst is het voor Siemens nodig om een aantal instellingen op de PLC te wijzigen, afhankelijk van het type PLC. Siemens S7 1200 en 1500 PLC's hebben namelijk uitgebreidere beveiligingen dan voorgaande modellen. Hierdoor is het nodig om bij een datablok in de *properties* de optie *optimized block access* uit te vinken (Figuur 16).

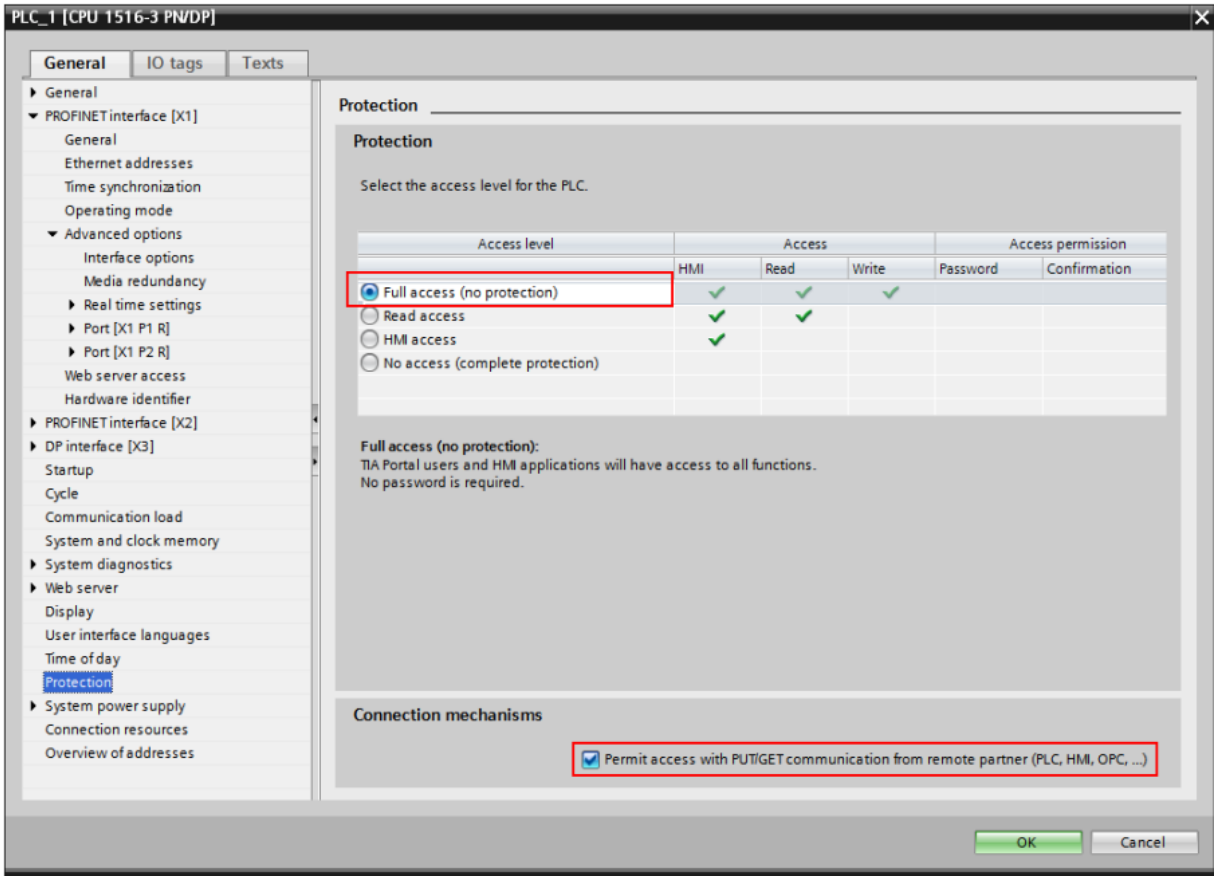
Als dit niet gebeurt is er een kans dat variabelen op een andere volgorde in het geheugen worden geplaatst en dat de standaardadressering niet meer klopt.



Figuur 16: Datablock properties

Daarna moeten de toegangspermissies goed worden gezet in de *properties* van de CPU, onder het kopje *protection*. Deze zal meestal al op *Full access* staan. Zo niet, dan moet deze optie worden aangevinkt. Tevens moet de optie *Permit access with PUT/GET communication from remote partner* worden ingeschakeld.

Met deze opties, te zien in Figuur 17, krijgen externe applicaties lees- en schrijftoegang tot de geheugens in de PLC.



Figuur 17: CPU properties

8.2 Snap7, Sharp7 en S7.NET+

Allereerst zijn de Snap7, Sharp7 en S7.NET+ libraries vergeleken, omdat deze op basis van beschrijving en geclaimde functionaliteit erg overeen lijken te komen. Het grootste verschil zit hem echter in de wijze van integratie: De Snap7 library bestaat uit een in C++ geschreven DLL welke met een C# wrapper kan worden aangesproken vanuit Visual Studio. Unity noemt dit een *native DLL*. S7.NET+ daarentegen is een volledig in C# geschreven library welke gebruik maakt van het .NET framework. De losse .cs bestanden hiervan zijn direct aan een project toe te voegen. Dit maakt troubleshooten en wijzigen van de library iets eenvoudiger. Mocht het echter nodig zijn de Snap7 library aan te passen, dan is het bijgeleverde MSVC project gemakkelijk te openen, en kan de DLL opnieuw gecompileerd worden.

Sharp7 is een port van Snap7 welke is geschreven in C# en wordt aangeleverd als één .cs bestand.

Het integreren van een zogenaamde *native DLL* in Unity is wat omslachtig. Unity heeft moeite met het vinden van de DLL, en moet daar expliciet op worden gewezen, met een extra stukje code, wat maar één keer hoeft te worden aangeroepen. Wanneer de applicatie wordt gebuild, dan lijkt dit niet meer nodig te zijn, echter plaatst Unity de DLL in het verkeerde directory (/test_Data/Plugins, in plaats van /test_Data/Mono).

Het lijkt erop dat in de gebuilde, standalone versie van de simulatie het voorheen genoemde stukje code niet aanwezig hoeft te zijn.

Onderstaande code moet worden toegevoegd aan het begin van de klassedefinitie in het script:

```
public class MyPluginClass
    {
        static MyPluginClass()
        {
            string currentpath = Environment.GetEnvironmentVariable("path",
EnvironmentVariableTarget.Process);
            string dllpath = Environment.CurrentDirectory + Path.DirectorySeparatorChar +
"assets" + Path.DirectorySeparatorChar + "plugins";
            if (currentpath.Contains(dllpath) == false)
            {
                Environment.SetEnvironmentVariable("path", currentpath + Path.PathSeparator
+ dllpath, EnvironmentVariableTarget.Process);
            }
        }
    }
```

Voordat de Snap7 functies worden aangeroepen moet er een object van de klasse worden aangemaakt, bijvoorbeeld in de start-functie:

```
MyPluginClass plug = new MyPluginClass();
```

De code hoeft slechts één keer per windows sessie worden aangeroepen.

9 Bibliografie

[1] J. Scholte, „Definitiestudie," Vlaardingen, 2016.

[2] J. Scholte, „PLAN VAN AANPAK Communicatie-architectuur PLC's en Unity," Vlaardingen, 2016.

[3] 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].

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

[5] 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.

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

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

[8] 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.

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

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

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

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

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

[14] 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].

[15] 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].

[16] 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].

[17] 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].

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

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

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

[21] 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].

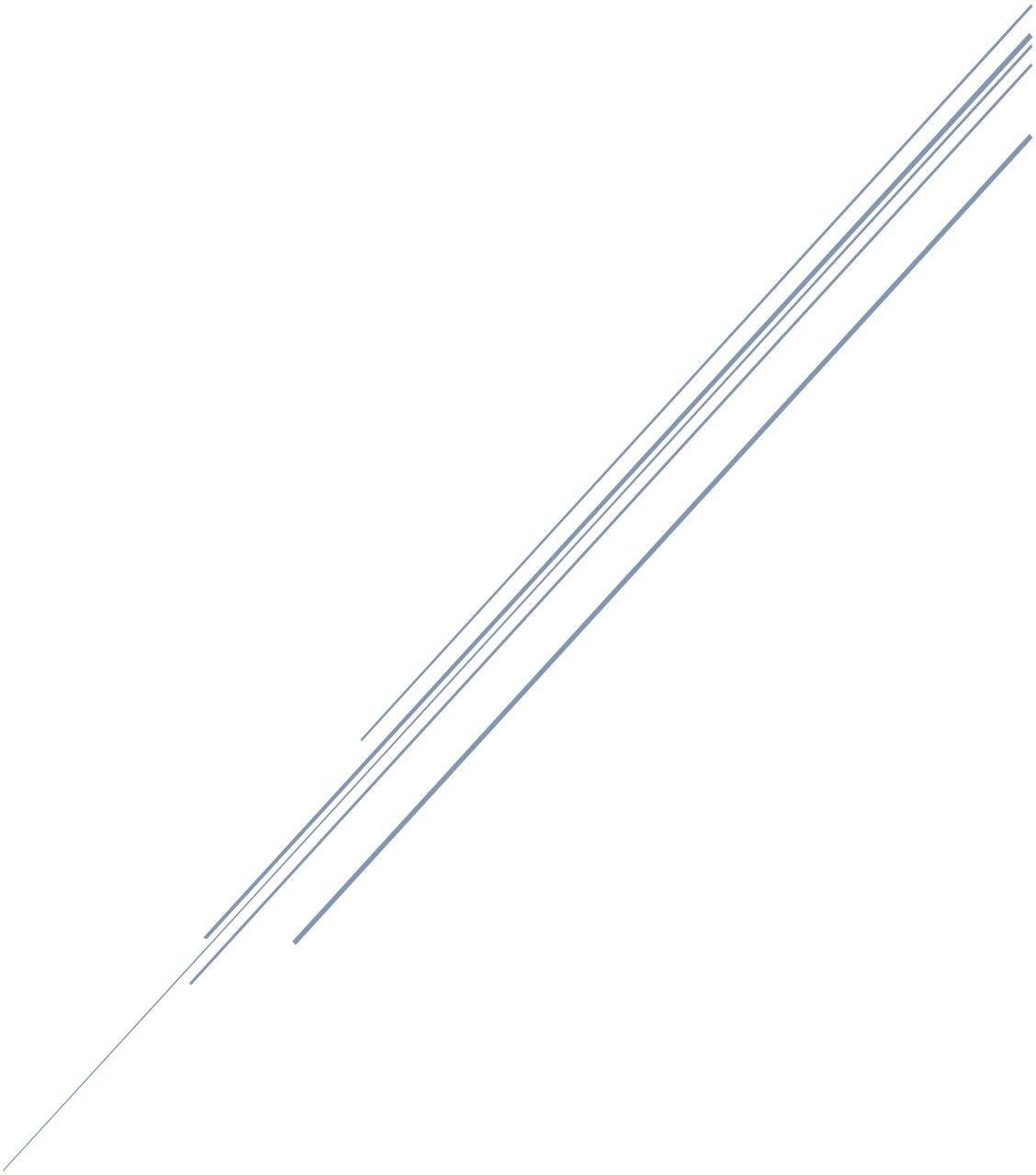
5 Bijlage E: Testrapport

jb systems
industrial automation

TESTRAPPORT

Communicatie-architectuur PLC's en Unity

22-12-2016



Jordy Scholte
Haagse Hogeschool Delft

Inhoudsopgave

1 Hoofdstuk

In dit document wordt beschreven welke tests er zijn uitgevoerd op de gekozen oplossingen uit het ontwerprapport [Bijlage D] [1]. Allereerst betreft dit een omschrijving van de testplannen, met daarin uiteengezet wat er wordt getest en op welke manier dit wordt gedaan. Vervolgens worden de resultaten hiervan behandeld. Als laatste in dit document wordt er een vergelijking gemaakt tussen de oplossingen, en wordt er een advies uitgebracht naar de opdrachtgever.

2 Testplan

Zoals beschreven in het ontwerprapport [1] zijn de eisen in Tabel 6 van toepassing op de gekozen oplossingen.

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 6: Eisen

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.

2.1 Installation/compatibility/functional [2]

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. 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 het ontwerprapport [1] 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.

2.2 Performance/concurrent/reliability [2, 3]

Bij de performancetests zal worden gemeten of de oplossingen voldoen aan de gestelde snelheids- en stabiliteitseisen [2, 3]. 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.

2.3 Eis 2 – Prijs

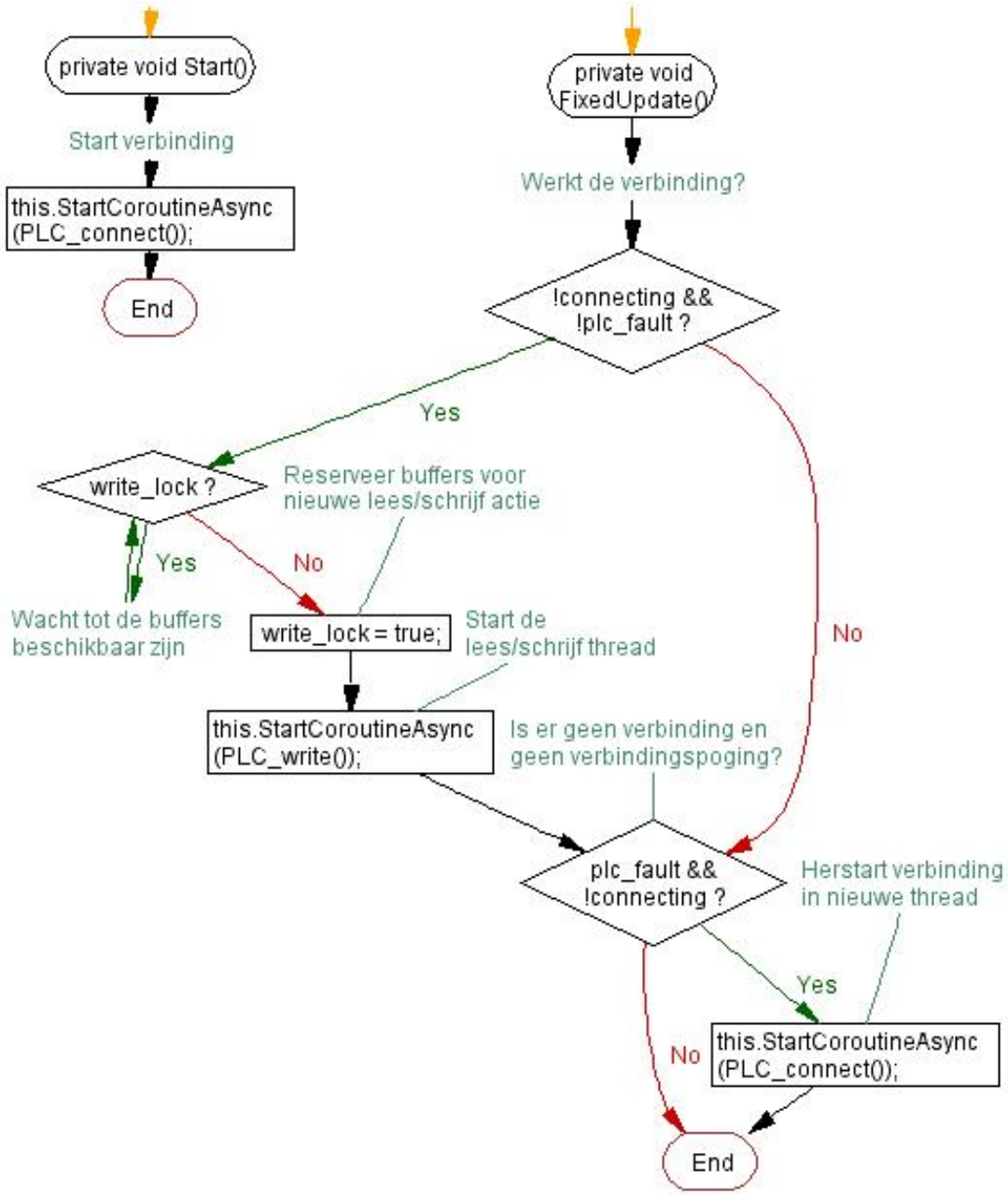
Voor eis 2, waarin wordt gesteld dat een oplossing niet duurder mag zijn dan WinMOD, kan geen test worden uitgevoerd. Dit is de enige uitzondering in de eisenlijst.

3 Voorstel voor applicatie

Figuur 18 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.

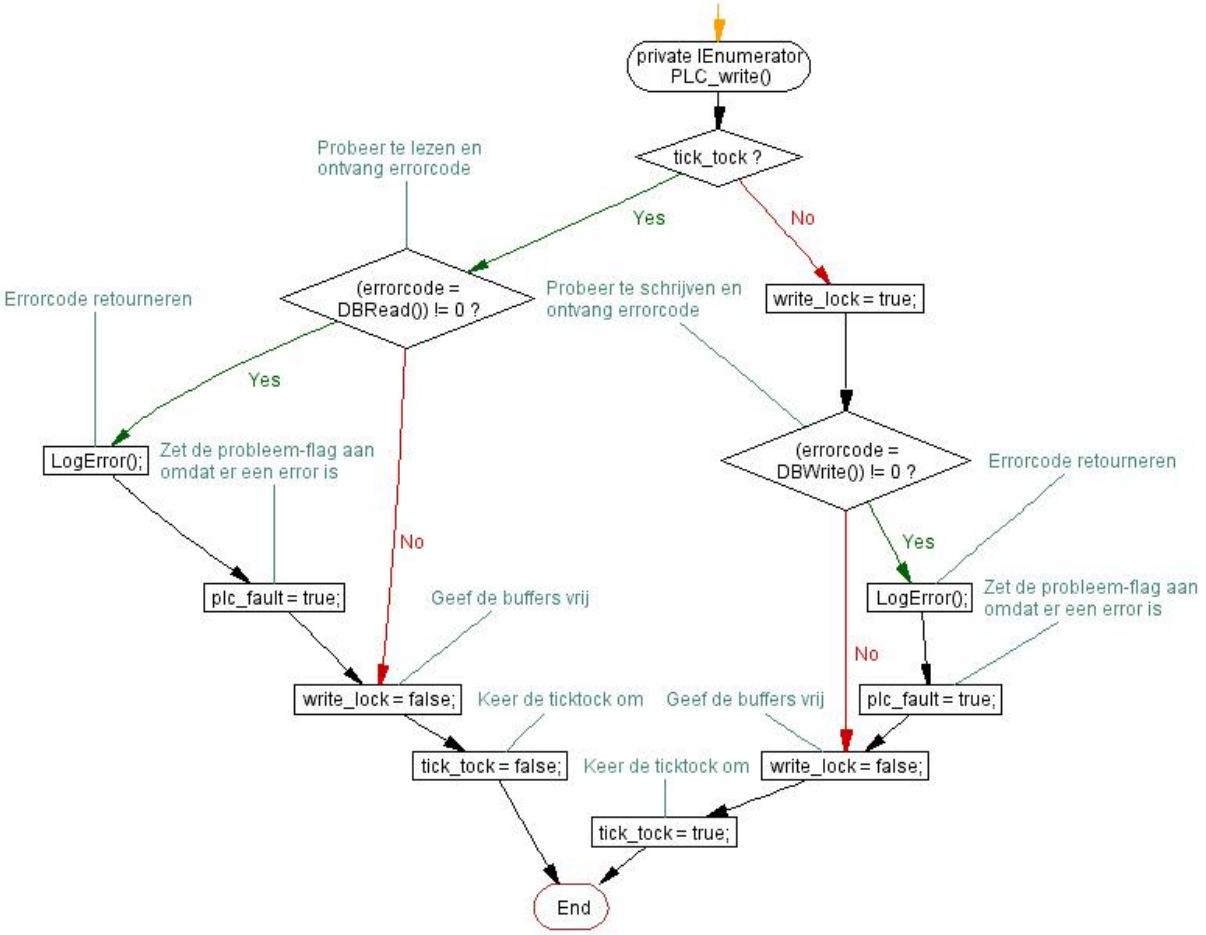
De applicatie zal vervolgens in de FixedUpdate kijken of er op dat moment een connectie wordt opgezet (connecting) en of dat er een probleem is met een bestaande verbinding (plc_fault). Zo niet, dan wordt er gewacht totdat de buffers beschikbaar zijn voor schrijven (write_lock). Zodra deze beschikbaar 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 18: Flowchart communicatie-applicatie

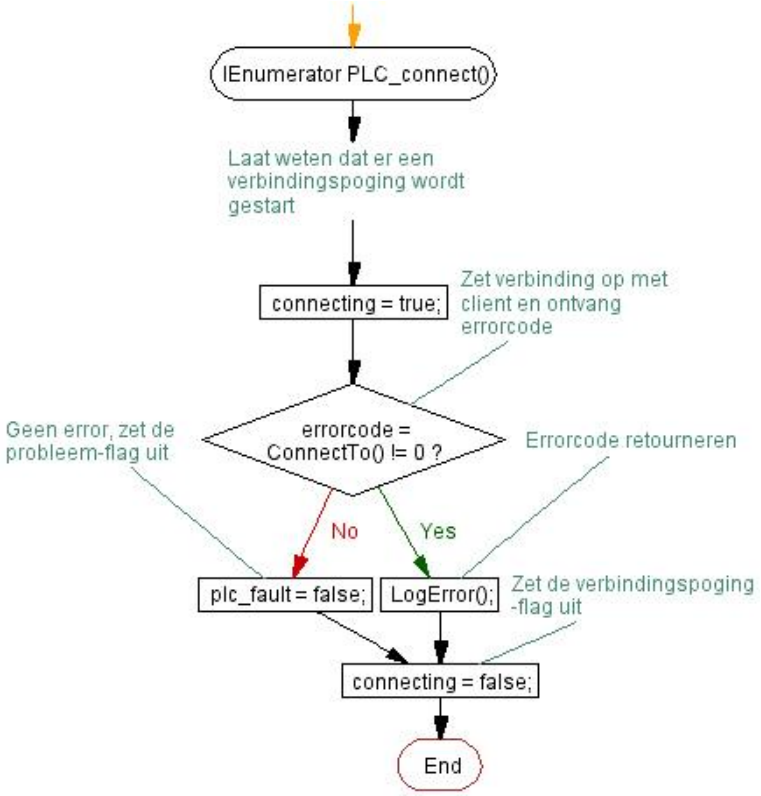
Als FixedUpdate is afgelopen blijft de communicatiethread (Figuur 19) in de achtergrond doorgaan, als deze is aangemaakt. Deze thread zal data naar de PLC schrijven, of deze uitlezen, afhankelijk van hetgeen in de voorgaande thread is uitgevoerd. Dit is een tick-tock systeem, om te voorkomen dat er tegelijkertijd 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-condities 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 19: Flowchart communicatiethread

De connectiethread (Figuur 20) 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, en wordt alleen de connecting flag uitgezet. Hierdoor kan er bij de volgende FixedUpdate weer een nieuwe verbindingspoging worden gestart.



Figuur 20: Flowchart connectiethread

4 Testplanning

Testweek 5 t/m 15 December

Dag	Locatie	Tests	Hardware
Maandag	Vlaardingen	Voorbereiding S7.NET+	
Dinsdag	Vlaardingen	S7.NET+	Siemens S7-1500
		Voorbereiding Sharp7/Snap7	
Woensdag	Vlaardingen	Sharp7/Snap7	Siemens S7-1500
Donderdag	Delft	Voorbereiding ASComm.NET op S7	
Vrijdag	Vlaardingen	ASComm.NET S7	Siemens S7-1500
Maandag	Vlaardingen	Voorbereiding ASComm.NET op AB en Modbus	
Dinsdag	Vlaardingen	ASComm.NET op AB	AB CL5561
Woensdag		Uitloopdag	
Donderdag	Vlaardingen	ASComm.NET Modbus	Schneider M340

Vanwege de voorbereidingsdagen in delft kunnen de simulaties 48 uur draaien voordat de nieuwe worden getest. De simulatie op de vrijdag kan zelfs 72 uur draaien (over het weekend heen).

5 Resultaten

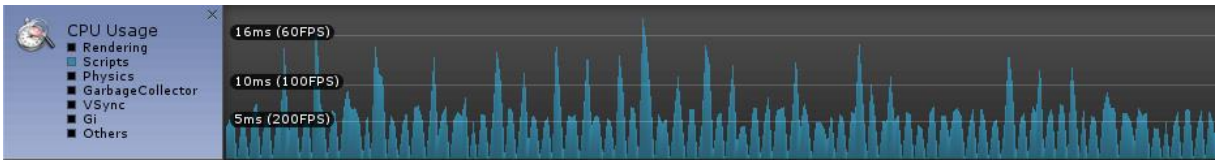
5.1 S7.NET+

Vanwege drukte in het lab bij JB Systems, is de S7.NET+ library helaas maar 24 uur in plaats van 48 uur getest. Hieruit zijn de onderstaande 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.

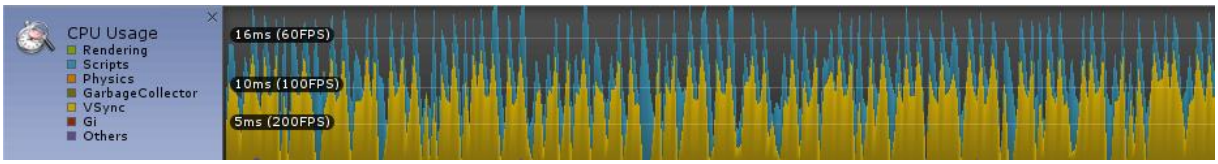
Na de testperiode was de waarde van de Reals op de PLC 4.271.651 en Unity's totale runtime eindigde op 85.868,08 seconde. Het delen van deze twee waarden levert een frequentie van 49,75Hz op. Omdat in deze test elke cycle wordt gelezen én geschreven resulteert dit voor een éénrichtingstransmissie in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

De gemiddelde scripttijd ligt onder de 10ms, zoals te zien in Figuur 21. 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 21: 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 22: S7.NET+ Frame timing

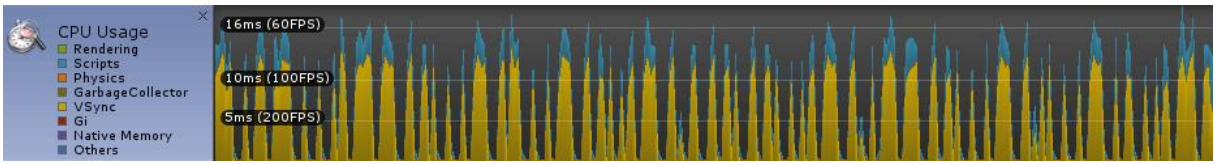
5.2 Snap7

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 23 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 23: 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.



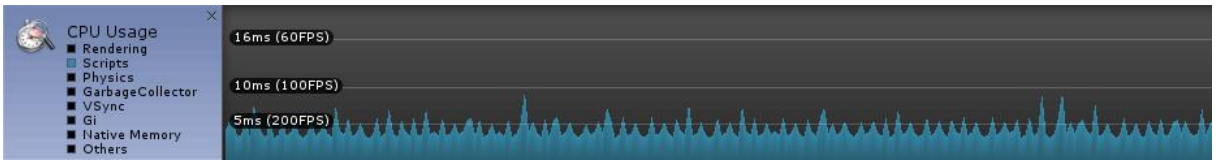
Figuur 24: Snap7 Frame timing

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 **Fout! Verwijzingsbron niet gevonden.**. 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.

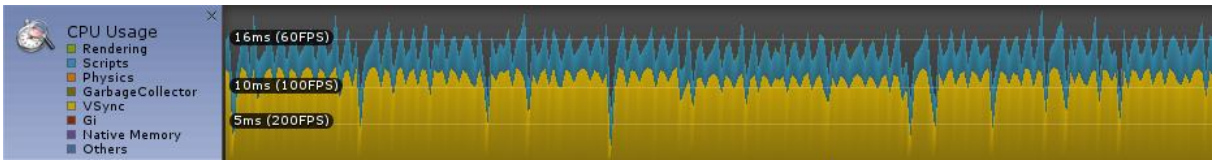
5.3 Sharp7

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 25 is te zien dat de gemiddelde scripttijd onder de 5ms ligt. Figuur 26 toont aan dat de gemiddelde framesnelheid rond de 70FPS ligt. Aan eis 6, 7 en 12 is hiermee voldaan.



Figuur 25: Sharp7 Script timing



Figuur 26: 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.

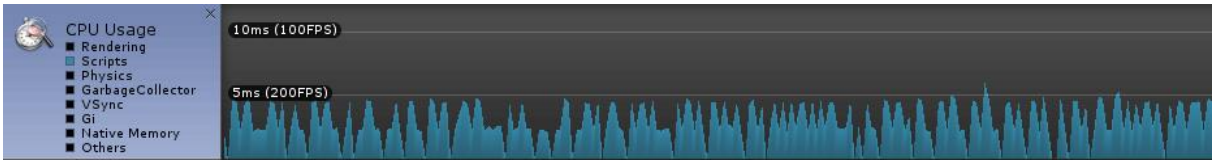
Het is lastig te verklaren waarom de frametijd hoger ligt dan die van Snap7, en waarom de uiteindelijke gemiddelde snelheid een paar Hertz lager ligt. Waarschijnlijk zit er in de .NET-gebaseerde Sharp7 library meer overhead dan in de Snap7 library welke is geschreven in C++. Allebei de libraries voldoen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

5.4 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.

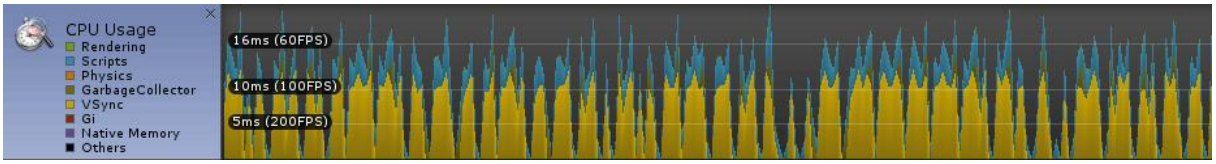
De uiteindelijke waarde van de Reals was 12.563.599, na een door Unity gemeten looptijd van 251.275 seconden. Hiermee is de frequentie vast te stellen op 49,999Hz. Hiermee is ASComm meteen al de library met de laagste afwijking. De éénrichtingssnelheid komt neer op 99,999Hz. Aan eis 1 is dus voldaan.

Figuur 27 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 27: ASComm Siemens Script timing

Figuur 28 laat zien hoe de gemiddelde framesnelheid tussen de 80 en de 100FPS ligt. Aan eis 12 is hiermee voldaan.

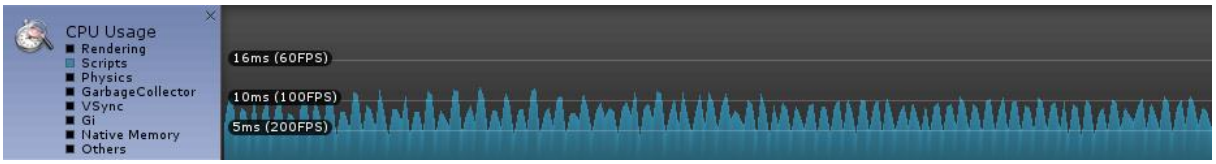


Figuur 28: ASComm Siemens Frame timing

5.5 ASComm.NET Allen-Bradley Logix driver

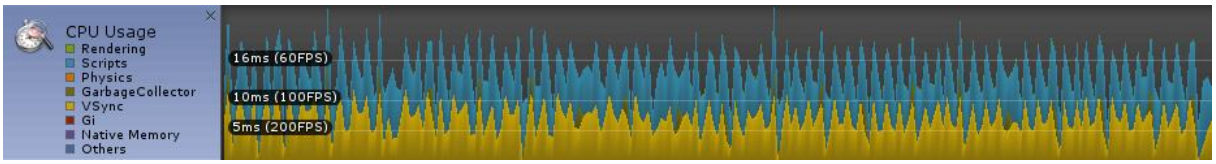
De Allen-Bradley Logix driver van de ASComm.NET library is getest gedurende 45 uur. Hierbij zijn geen fouten aangetroffen in de logs. De waarde van de Reals na deze periode was opgelopen tot 4.458.240 na een door Unity gemeten runtime van 161.948 seconden. Het ontbreken van verschillen tussen de Reals 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 29: ASComm A-B Script timing

In Figuur 30 is te zien hoe de gemiddelde frametijd rond de 70FPS ligt, waarmee aan eis 12 is voldaan.



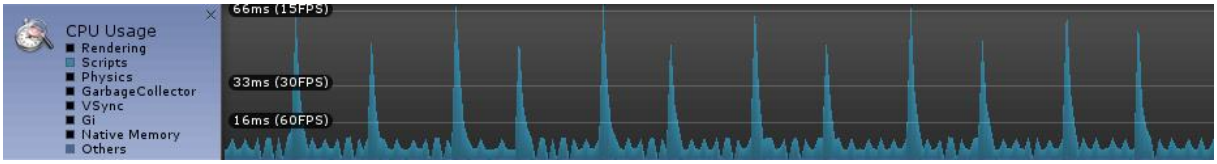
Figuur 30: ASComm A-B Frame timing

5.6 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 [4] 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 [5]. 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.

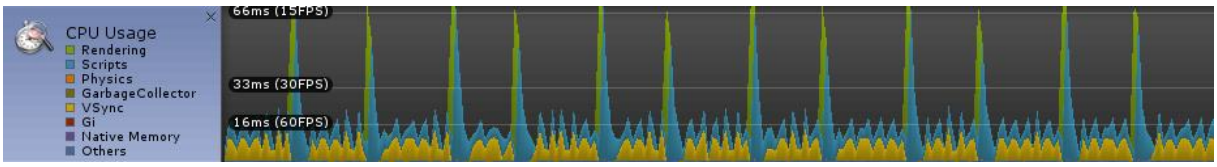
Vervolgens is er met Unity de test gedraaid. Hieruit bleek na bijna 10 minuten runtime dat Modbus bij lange na niet aan eis 1 voldoet. Na 555,0818 seconden dat de simulatie liep waren de Floats opgelopen tot 6142. Dit komt neer op een snelheid van 11,07Hz. Door dit om te rekenen naar éénrichtingstransmissie 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 31 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 31: ASComm Modbus Script timing

De gemiddelde framesnelheid ligt rond de 60FPS, waarmee dus wel voldaan is aan eis 12 (Figuur 32).



Figuur 32: ASComm Modbus Frame timing

5.7 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	99,999Hz
ASComm.NET Modbus driver	22,13Hz	Getest met virtuele PLC
ASComm.NET A-B driver	55,06	

In Tabel 7 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 7: Vergelijkingstabel met resultaten

6 Conclusie en aanbevelingen

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

6.1 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.

6.2 Aanbevelingen

Kijkend naar de resultaten in dit document 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.

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 grootste minpunt de prijs.

7 Bibliografie

[1] J. Scholte, „Ontwerprapport [Bijlage D],” Vlaardingen, 2016.

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

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

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

[5] 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].

6 Bijlage F: IO List Bravenes

shared memory Unity

	location	U3D Offset	array [item]	U3D Count	U3D Read/Write	name		type	unit	min	max	opmerkingen
0	0	1	1		Read		Write by Winmod	real	Graden	-10	100	MANIPULATOR AFT: Arm rotatie
1	4		2		Read		Write by Winmod	real	m	0	10	MANIPULATOR AFT: Lift beweging
2	8		3		Read		Write by Winmod	real	%	0	100	MANIPULATOR AFT: Clamp position 0% = open 100% = dicht
3	12		4		Read		Write by Winmod	real	%	0	100	MANIPULATOR AFT: Clamp Lock 0% = unlock 100% = Lock
4	16		5		Read		Write by Winmod	real	-	0	1	MANIPULATOR AFT: Bucket detection 0 = geen 1= bucket aanwezig
5	20		6		Read		Write by Winmod	real	Graden	-10	100	MANIPULATOR FWD: Arm rotatie
6	24		7		Read		Write by Winmod	real	m	0	10	MANIPULATOR FWD: Lift beweging
7	28		8		Read		Write by Winmod	real	%	0	100	MANIPULATOR FWD: Clamp position 0% = open 100% = dicht
8	32		9		Read		Write by Winmod	real	%	0	100	MANIPULATOR FWD: Clamp Lock 0% = unlock 100% = Lock
9	36		10		Read		Write by Winmod	real	-	0	1	MANIPULATOR FWD: Bucket detection 0 = geen 1= bucket aanwezig
10	40		11		Read		Write by Winmod	real	m	0	10	BUCKETLOADER AFT: Lift beweging
11	44		12		Read		Write by Winmod	real	%	0	100	BUCKETLOADER AFT: Fork position 0% = Fork in, 100% = Fork out
12	48		13		Read		Write by Winmod	real	%	0	100	BUCKETLOADER AFT: Seperator 0% = In , 100% = out
13	52		14		Read		Write by Winmod	real	-	0	1	BUCKETLOADER AFT: Bucket detectie 0 = geen 1 = bucket aanwezig
14	56		15		Read		Write by Winmod	real	-	0	1	BUCKETLOADER AFT: Bucket type 0 = plastic 1 = metaal
15	60		16		Read		Write by Winmod	real	m	0	10	BUCKETLOADER FWD: Lift beweging
16	64		17		Read		Write by Winmod	real	%	0	100	BUCKETLOADER FWD: Fork position 0% = Fork in, 100% = Fork out
17	68		18		Read		Write by Winmod	real	%	0	100	BUCKETLOADER FWD: Seperator 0% = In , 100% = out
18	72		19		Read		Write by Winmod	real	-	0	1	BUCKETLOADER FWD: Bucket detectie 0 = geen 1 = bucket aanwezig
19	76		20		Read		Write by Winmod	real	-	0	1	BUCKETLOADER FWD: Bucket type 0 = plastic 1 = metaal
20	80		21		Read		Write by Winmod	real	m	0	3	CARROUSEL AFT: Upper cylinder
21	84		22		Read		Write by Winmod	real	m	0	3	CARROUSEL AFT: Lower cylinder
22	88		23		Read		Write by Winmod	real	%	0	100	CARROUSEL AFT: Drive lock 0% = unlock, 100% = locked
23	92		24		Read		Write by Winmod	real	%	0	100	CARROUSEL AFT: Loader lock 0% = unlocked, 100% = locked
24	96		25		Read		Write by Winmod	real	-	1	16	CARROUSEL AFT: CassetteNR @ loader
25	100		26		Read		Write by Winmod	real	m	0	3	CARROUSEL FWD: Upper cylinder
26	104		27		Read		Write by Winmod	real	m	0	3	CARROUSEL FWD: Lower cylinder
27	108		28		Read		Write by Winmod	real	%	0	100	CARROUSEL FWD: Drive lock 0% = unlock, 100% = locked
28	112		29		Read		Write by Winmod	real	%	0	100	CARROUSEL FWD: Loader lock 0% = unlocked, 100% = locked
29	116		30		Read		Write by Winmod	real	-	1	16	CARROUSEL FWD: CassetteNR @ loader
30	120		31		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 1
31	124		32		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 2
32	128		33		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 3
33	132		34		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 4
34	136		35		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 5
35	140		36		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 6
36	144		37		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 7
37	148		38		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 8
38	152		39		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 9
39	156		40		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 10
40	160		41		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 11
41	164		42		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 12
42	168		43		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 13
43	172		44		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 14
44	176		45		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 15
45	180		46		Read		Write by Winmod	real	-	1	18	CARROUSEL AFT: Aantal buckets in cassette 16
46	184		47		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 1
47	188		48		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 2
48	192		49		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 3
49	196		50		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 4
50	200		51		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 5
51	204		52		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 6
52	208		53		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 7
53	212		54		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 8
54	216		55		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 9
55	220		56		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 10
56	224		57		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 11
57	228		58		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 12
58	232		59		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 13
59	236		60		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 14
60	240		61		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 15
61	244		62		Read		Write by Winmod	real	-	1	18	CARROUSEL FWD: Aantal buckets in cassette 16
62	248		63		Read		Write by Winmod	real	m	-100	1000	ROV AFT: ROV kabel lengte
63	252		64		Read		Write by Winmod	real	m	-100	1000	ROV FWD: ROV kabel lengte
64	256		65		Read		Write by Winmod	real	m	-100	1000	ROV SB: ROV kabel lengte
65	260		66		Read		Write by Winmod	real	m	-100	1000	CHAIN FALL PIPE: lengte afgerolde ketting
67	264		67		Read		Write by Winmod	real	Graden	-100	100	ROV: Roll
68	268		68		Read		Write by Winmod	real	Graden	-100	100	ROV: Pitch
69	272		69		Read		Write by Winmod	real	%	-100	100	CHAIN STORAGE AFT: Spoolerdrive
70	276		70		Read		Write by Winmod	real	m	-10	30	CHAIN STORAGE AFT: Gantry position
71	280		71		Read		Write by Winmod	real	m	-10	30	CHAIN STORAGE AFT: Trolley position
72	284		72		Read		Write by Winmod	real	m	-2	15	CHAIN STORAGE AFT: Buffer position
73	288		73		Read		Write by Winmod	real	%	-100	100	CHAIN STORAGE FWD: Spoolerdrive
74	292		74		Read		Write by Winmod	real	m	-10	30	CHAIN STORAGE FWD: Gantry position
75	296		75		Read		Write by Winmod	real	m	-10	30	CHAIN STORAGE FWD: Trolley position
76	300		76		Read		Write by Winmod	real	m	-2	15	CHAIN STORAGE FWD: Buffer position
77	304		77		Read		Write by Winmod	real	-	0	1	SYNC UNIT: Bucket positie 1
78	308		78		Read		Write by Winmod	real	-	0	1	SYNC UNIT: Bucket positie 0
79	312		79		Read		Write by Winmod	real	-	0	1	SYNC UNIT: Bucket positie -1
80	316		80		Read		Write by Winmod	real	-	0	1	SYNC UNIT: Running
81	320		81		Read		Write by Winmod	real	graden	0	360	SYNC UNIT: Positie slag
82	324		82		Read		Write by Winmod	real	m	-10	30	CATCHER FRAME AFT PS: Position
83	328		83		Read		Write by Winmod	real	m	-10	30	CATCHER FRAME AFT SB: Position
84	332		84		Read		Write by Winmod	real	m	-10	30	CATCHER FRAME FWD PS: Position
85	336		85		Read		Write by Winmod	real	m	-10	30	CATCHER FRAME FWD SB: Position
86	340		86		Read		Write by Winmod	real	m	-10	30	CHAIN GUIDANCE FRAME AFT: Position
87	344		87		Read		Write by Winmod	real	m	-10	30	CHAIN GUIDANCE FRAME FWD: Position
88	348		88		Read		Write by Winmod	real	%	0	100	HATCHDOOR ROV: Position. 100% =closed
89	352		89		Read		Write by Winmod	real	%	0	100	HATCHDOOR BUCKET SB: Position. 100% =closed
90	356		90		Read		Write by Winmod	real	%	0	100	HATCHDOOR BUCKET PS: Position. 100% = closed
91	360		91		Read		Write by Winmod	real	-	0	3	BUCKETLOADER AFT: Aantal buckets in loader
92	364		92		Read		Write by Winmod	real	-	0	3	BUCKETLOADER FWD: Aantal buckets in loader
93	368		93		Read		Write by Winmod	real	-	0	1	MANIPULATOR AFT: Bucket type 0 = plastic 1 = metaal
94	372		94		Read		Write by Winmod	real	-	0	1	MANIPULATOR FWD: Bucket type 0 = plastic 1 = metaal
95	376		95		Read		Write by Winmod	real	-	0	3	MANIPULATOR AFT: Aantal buckets in manipulator
96	380		96		Read		Write by Winmod	real	-	0	3	MANIPULATOR FWD: Aantal buckets in manipulator

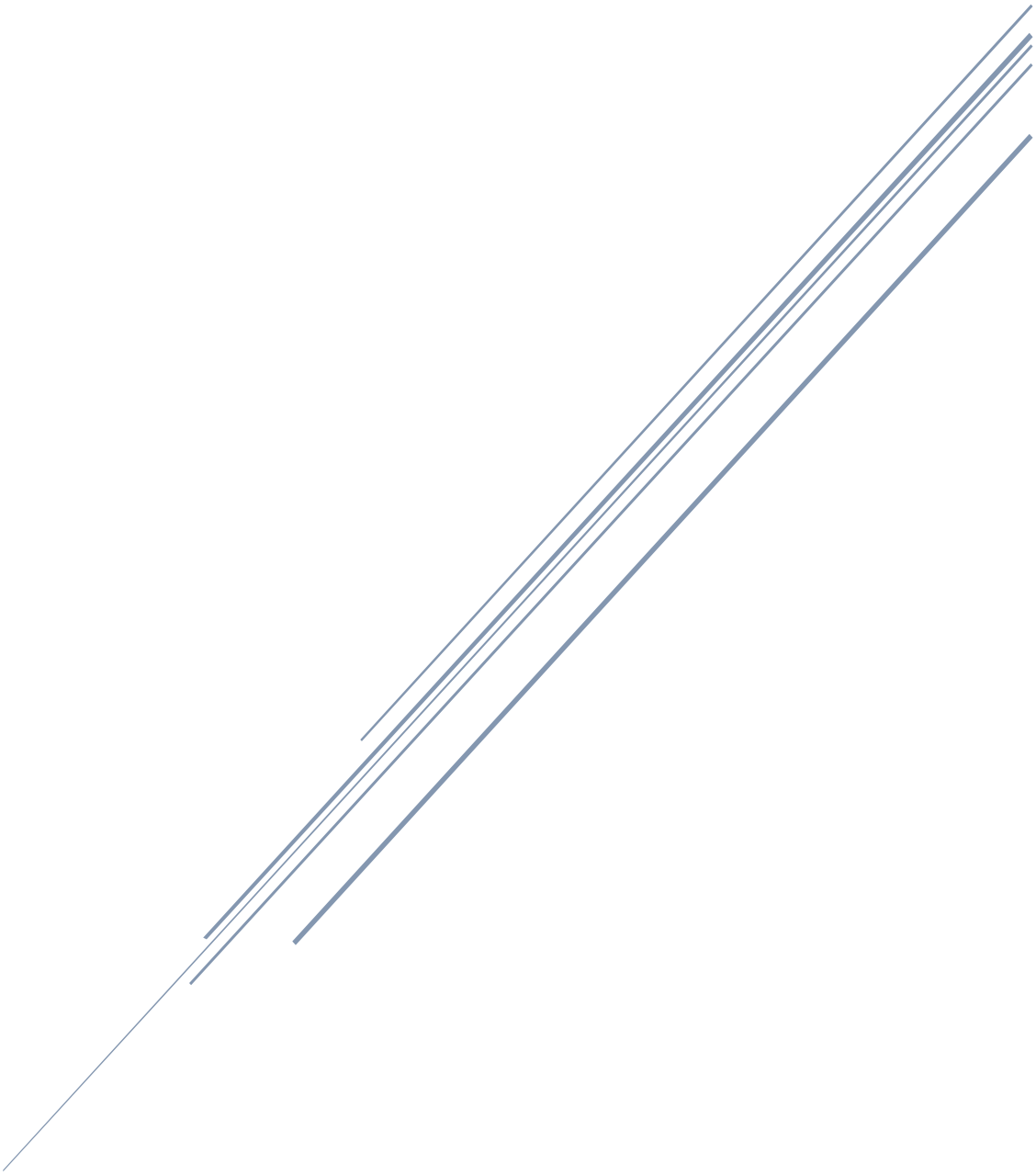
7 Bijlage G: Codedocument

jb systems
industrial automation

CODEDOCUMENT

Communicatie-architectuur PLC's en Unity

22-12-2016



Jordy Scholte
Haagse Hogeschool Delft

Inhoudsopgave

1 Inleiding

Dit document laat de code zien die tijdens de tests van de libraries is gebuikt. Voor libraries of drivers welke zeer veel overeen komen is maar één voorbeeld getoond. Eventuele bijzonderheden voor een bepaald stuk code staan vermeld aan het begin van het hoofdstuk of in de code-comments. Als laatste is het algemene 'skelet' te zien dat kan worden gebruikt om andere communicatielibraries te parallelliseren.

2 S7.NET+

Onderstaande code is gebruikt voor de S&.NET+ implementatie. Opvallend aan deze library is dat er per transmissie maar 200 bytes kunnen worden verzonden. Voor 100 Reals zijn 400 bytes nodig, waardoor er 2 lees- en 2 schrijfacties worden gedaan, met ieder twee aparte buffers.

```
using System.Collections;
using S7.Net;
using Threader;
using UnityEngine;

public class PLC_S7_NET : MonoBehaviour
{
    private static readonly string plc_ip = "192.168.78.128";
    public double[] DB1converted = new double[50];
    private readonly double[] DB1converted2 = new double[50];
    private int iterator;

    private byte output;
    private readonly Plc plc = new Plc(CpuType.S71500, plc_ip, 0, 1);
    private bool plc_fault = true, connecting = true, write_lock, tick_tock = true;
    private double[] reals;
    private double[] reals2;
    private ErrorCode start_error;

    private void Start()
    {
        this.StartCoroutineAsync(PLC_connect());
    }

    private void FixedUpdate()
    {
        if (!connecting && !plc_fault)
        {
            while (write_lock)
            {
            }

            write_lock = true;

            this.StartCoroutineAsync(PLC_write());
        }
        if (plc_fault && !connecting)
        {
            Debug.Log("Trying to reestablish connection to target PLC");
            this.StartCoroutineAsync(PLC_connect());
        }
    }

    private void OnApplicationQuit()
    {
        Debug.Log("Application ending after " + Time.realtimeSinceStartup + " seconds");
        plc.Close();
    }

    private IEnumerator PLC_write()
    {
        if (tick_tock)
        {
            try
            {
                reals = (double[]) plc.Read(DataType.DataBlock, 1, 0, VarType.Real, 50);
                reals2 = (double[]) plc.Read(DataType.DataBlock, 1, 200, VarType.Real, 50);
                // twee leesacties van 50 Reals (200 bytes) elk
            }
            catch
            {
                plc_fault = true;
                Debug.Log("Read exception");
            }
            tick_tock = false;
            write_lock = false;
```

```
        }
        else
        {
            write_lock = true;
            iterator = 0;
            foreach (var real in reals)
            {
                DB1converted[iterator] = real + 1;
                DB1converted2[iterator] = reals2[iterator] + 1;
                iterator++;
            }
            try
            {
                start_error = plc.Write(DataType.DataBlock, 1, 0, DB1converted);
                start_error = plc.Write(DataType.DataBlock, 1, 200, DB1converted2);
                //Debug.Log(start_error);
            }
            catch
            {
                plc_fault = true;
                Debug.Log("Write exception");
            }
            write_lock = false;
            tick_tock = true;
        }
        yield break;
    }

    private IEnumerator PLC_connect()
    {
        connecting = true;
        if ((start_error = plc.Open()) != 0) // 2) CONNECTEN MET CLIENT (PLC)
        {
            Debug.LogError("Error connecting to target: " + start_error); //(EVT. ERROR CODE RETURNEN)
        }

        else
        {
            plc_fault = false;
            Debug.Log("Connected to target");
        }
        connecting = false;
        yield break;
    }
}
```

3 Sharp7/Snap7

De code hieronder is geschikt voor zowel de Sharp7 als de Snap7 library. In onderstaande code is de referentie naar Sharp7 gemaakt. Voor Snap7 moet deze referentie uiteraard worden veranderd.

```
using System.Collections;
using Sharp7;
using Threader;
using UnityEngine;

public class Script_Sharp7 : MonoBehaviour
{
    private readonly byte[] DB1 = new byte[400], DB1_write = new byte[400];
    private readonly float[] DB1converted = new float[100];
    private readonly int DBSize = 400;

    private readonly S7Client MyClient = new S7Client(); // 1) NIEUWE S7 CLIENT AANMAKEN
    private bool plc_fault = true, connecting, write_lock, tick_tock = true;
    private int returnvalConnect;
    private int returnvalRead;
    private int tempReturn;

    // Use this for initialization
    private void Start()
    {
        this.StartCoroutineAsync(PLC_connect());
    }

    private void FixedUpdate()
    {
        if (!connecting && !plc_fault)
        {
            while (write_lock)
            {
            }

            write_lock = true;

            this.StartCoroutineAsync(PLC_write());

        }
        if (plc_fault && !connecting)
        {
            this.StartCoroutineAsync(PLC_connect());
        }
    }

    private void OnApplicationQuit()
    {
        Debug.Log("Application ending after " + Time.realtimeSinceStartup + " seconds");
        MyClient.Disconnect();
    }

    private IEnumerator PLC_write()
    {
        if (tick_tock)
        {
            write_lock = true;

            if ((returnvalRead = MyClient.DBRead(1, 0, DBSize, DB1)) != 0)
                // 3) LEZEN VAN DB1, BEGINNEND BIJ VALUE 0, LENGTE = DBSIZE, SCHRIJF NAAR BUFFER 'DB1'
                Debug.LogError("Error reading from target: " + returnvalRead); //(EVT. ERROR CODE
RETURNEN)
            tick_tock = false;
            write_lock = false;
        }
        else
        {
            write_lock = true;
            for (var i = 0; i < DBSize; i += 4)
            {
                DB1converted[i/4] = S7.GetRealAt(DB1, i) + (float) 1.0; //waarde uit db1 verhogen met 1
om te testen
```

```
                S7.SetRealAt(DB1_write, i, DB1converted[i/4]);
            }
            MyClient.DBWrite(1, 0, DBSize, DB1_write);
            // 4) SCHRIJVEN NAAR DB1, BEGINNEND BIJ VALUE 0, LENGTE = DBSIZE, LEES VANUIT BUFFER 'DB1'
            tick_tock = true;
            write_lock = false;

        }
        yield break;
    }

    private IEnumerator PLC_connect()
    {
        connecting = true;
        if ((returnvalConnect = MyClient.ConnectTo("192.168.78.128", 0, 1)) != 0)
        {
            Debug.LogError("Error connecting to target: " + returnvalConnect); //Return error code if
any
        }

        else
        {
            plc_fault = false;
            Debug.Log("Connected to target");
        }
        connecting = false;
        yield break;
    }
}
```

4 ASComm.NET

Onderstaande code toont de ASComm.NET implementatie voor Siemens. Voor de andere drivers is de code nagenoeg gelijk; Alleen het aanmaken van de connectie verschilt, zoals het initialiseren van tags, datablokken en datatypes. Uiteraard moeten ook de referenties in de declaratie worden aangepast om de betreffende driver te reflecteren.

Deze code is de enige zonder aparte *PLC_connect()* functie. Dit is niet nodig omdat de ASComm.NET drivers automatisch op de achtergrond de verbinding in stand houden.

```
using System;
using System.Collections;
using System.ComponentModel;
using AutomatedSolutions.Win.Comm.SI.S7.Net;
using Threader;
using UnityEngine;
using SIS7 = AutomatedSolutions.Win.Comm.SI.S7;

public class ASComm : MonoBehaviour
{
    private readonly float[] DB1converted = new float[100];
    private readonly Channel myChan = new Channel();

    private SIS7.Device myDevice;
    private SIS7.Group myGroup;
    private SIS7.Item myItem;
    private bool plc_fault = true;
    private bool connecting = true;
    private bool write_lock, tick_tock = true;


    // Use this for initialization
    private void Start()
    {
        try
        {
            Debug.Log("channel gemaakt");
            myDevice = new SIS7.Device("192.168.78.128,0,1", SIS7.Model.S7_1200, 1000, 50); // Merk-
            specifiek
            Debug.Log("device gemaakt");
            myGroup = new SIS7.Group(false, 500);
            Debug.Log("group gemaakt");
            myItem = new SIS7.Item("DB1.DBD0");// Merk-specifiek
            Debug.Log("item gemaakt");

            myChan.Devices.Add(myDevice);
            myDevice.Groups.Add(myGroup);
            myGroup.Items.Add(myItem);
            Debug.Log("dingen aan elkaar toegevoegd");

            myItem.HWDataType = SIS7.DataType.Single; // Merk-specifiek
            myItem.Elements = 100;
        }
        catch (LicenseException lex)
        {
            if (lex.Message.ToLower().Contains("expired"))
                Debug.Log("Trial Expired");
            else
                Debug.Log(lex.Message + " ASComm.NET License Exception Thrown");
            throw lex;
        }
        catch (Exception ex)
        {
            throw ex;
        }
                connecting = false;
                plc_fault = false;
    }

    private void FixedUpdate()
    {
```

```
                if (!write_lock)
        {
            write_lock = true;
            this.StartCoroutineAsync(PLC_write());
        }
    }


    private void OnApplicationQuit()
    {
        Debug.Log("Application ending after " + Time.realtimeSinceStartup + " seconds");
        myDevice.Dispose();
    }

    private IEnumerator PLC_write()
    {
        if (tick_tock)
        {
                        write_lock = true;
            try
            {
                myItem.Read();
            }
            catch
            {
                Debug.Log("Read exception");
                tick_tock = false;
                write_lock = false;
                yield break;
            }
            for (var i = 0; i < myItem.Elements; i++)
                DB1converted[i] = (float) myItem.Values[i] + (float) 1.0;
            tick_tock = false;
            write_lock = false;
        }
        else
        {
            write_lock = true;
            try
            {
                myItem.Write(DB1converted);
            }
            catch
            {
                Debug.Log("Write exception");
            }
            write_lock = false;
            tick_tock = true;
        }
    }
}
```

5 Algemene parallellisatie

Deze code kan worden gebruikt om een eigen library of driver mee te parallelliseren. De connectie wordt automatisch hersteld bij problemen. Dit herstel alsmede het lezen en schrijven worden parallel aan de game-logica uitgevoerd.

```
using System.Collections;
using Threader;
using UnityEngine;

public class PLC_S7_NET : MonoBehaviour
{
    private static readonly string plc_ip = /*IP address of PLC*/;
    public double[] DB1converted = new double[100];
    private int DBSize;

        //Declare PLC if necessary

    private byte output;
    private bool plc_fault = true, connecting = true, write_lock, tick_tock = true;
    private double[] reals;
    private ErrorCode start_error;

    private void Start()
    {
        this.StartCoroutineAsync(PLC_connect());
    }

    // Update is called once per frame
    private void Update()
    {
    }

    private void FixedUpdate()
    {
        if (!connecting && !plc_fault)
        {
            while (write_lock)
            {
            }

            write_lock = true;
            this.StartCoroutineAsync(PLC_write());
        }
        if (plc_fault && !connecting)
        {
            Debug.Log("Trying to reestablish connection to target PLC");
            this.StartCoroutineAsync(PLC_connect());
        }
    }

    private void OnApplicationQuit()
    {
        Debug.Log("Application ending after " + Time.realtimeSinceStartup + " seconds");
        plc.Close();
    }

    private IEnumerator PLC_write()
    {
        if (tick_tock)
        {
            try
            {
                reals = //Read DB from PLC function
            }
            catch
            {
                plc_fault = true;
                Debug.Log("Read exception");
            }
            tick_tock = false;
            write_lock = false;
```

```
            yield break;
        }
                else
                {
                        write_lock = true;
                        for (var i = 0; i < DBSize; i += 4)
                        {
                                //Manipulate values
                        }
                        try
                        {
                                start_error = //Write DB to PLC function
                                //Debug.Log(start_error);
                        }
                        catch
                        {
                                plc_fault = true;
                                Debug.Log("Write exception");
                        }
                        write_lock = false;
                        tick_tock = true;
                }
    }

    private IEnumerator PLC_connect()
    {
        connecting = true;
        if ((start_error = /*Connect with PLC function*/) != 0)
        {
            Debug.LogError("Error connecting to target: " + start_error); //Return error code if any
        }

        else
        {
            plc_fault = false;
            Debug.Log("Connected to target");
        }
        connecting = false;
        yield break;
    }
}
```