Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Leevi Pärssinen

Ambulanssisimulaattorin kommunikointi ohjelmoitavalle logiikalle

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Konetekniikka

Insinöörityö

28.2.2020

Tekijä Otsikko Sivumäärä Aika	Leevi Pärssinen Ambulanssisimulaattorin kommunikointi ohjelmoitavalle logiikalle 25 sivua + 1 liite 28.02.2020
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Konetekniikka
Suuntautumisvaihtoehto	Koneautomaatio
Ohjaaja	Lehtori Antti Liljaniemi

Insinöörityön tarkoituksena oli selvittää, miten saadaan opiskelijoiden Unity-pelimoottorilla luoma ambulanssipeli kommunikoimaan TwinCAT-nimisen ohjelmiston kanssa. Tarkoituksena oli ohjata reaaliajassa fyysistä ajotuolia ambulanssipelin sisällä tapahtuvilla asennon muutoksilla. Ajotuolin ohjausjärjestelmään kuului kuusi pneumaattisesti ohjattua lihasta, jotka paineilmaa syöttäessä pullistuvat, logiikkaterminaali sekä logiikkaohjelma, joka toimii tietokoneella TwinCAT-ohjelmistossa.

Alkuperäinen idea oli käyttää Mevean kinematiikkasimulointiohjelmaa, joka toimii yhteistyössä Unityn kanssa ja jolla on mahdollisuus kommunikoida PLC-ohjelman muuttujien kanssa TwinCAT-ohjelmistossa. Todettiin kuitenkin, että tätä ei ole mahdollista toteuttaa, koska Mevea ei huomioisi kosketuspintoja Unityn puolella, kun pelin maailmaa ei ollut mahdollista siirtää suoraan Mevean puolelle.

Seuraavaksi tutkittiin, olisiko mahdollista kirjoittaa ambulanssin asema-arvot tekstitiedostoon ja lukea ne TwinCAT-ohjelmiston ”FileRead”-komennolla. Komento ei kuitenkaan pystynyt lukemaan muuttuvaa tiedostoa tai tiedosto tulisi avata ja sulkea jatkuvasti, mikä puolestaan ei olisi tehokasta.

Toimivaksi ratkaisuksi ongelmaan osoittautui TwinCAT ADS (Automation Device Specification). Sen avulla saadaan kirjoitettua data Unity:n puolelta suoraan PLC-ohjelman muuttujiin. Unity ei suoraan tukenut TwinCAT ADSia, mutta onneksi saksalainen yritys nimeltään oli luonut Unity-lisäosan, johon oli lisätty TwinCAT ADS -tuki.

Lisäosan avulla saatiin tuotua signaalit TwinCAT-ohjelmistossa toimivasta PLC-ohjelmasta. Myös skriptini, joka kirjoittaa ambulanssin Eulerin kulmat, nopeus- ja kiihtyvyysarvot signaaleihin, onnistui. Sen avulla arvot päivittyvät PLC-ohjelmaan reaaliajassa.

Avainsanat	PLC-ohjelma, ADS, ambulanssisimulaattori, TwinCAT

Author Title	Leevi Pärssinen Communication Solution for an Ambulance Simulator and a Programmable Logic Controller
Number of Pages Date	25 pages + 1 appendix 28 February 2020
Degree	Bachelor of Engineering
Degree Programme	Mechanical Engineering
Specialisation option	Machine Automation
Instructor	Antti Liljaniemi, Senior Lecturer

The objective of the thesis was to examine how to make an ambulance game created by students communicate with TwinCAT. The aim was to control the driving chair in real time with the transformation of the ambulance's rotational values in-game. The control system of the chair consists of 6 pneumatic muscles, which contract when pressure is added, a logic terminal and a logic program running on a computer inside TwinCAT's software.

The original purpose was to use Mevea's kinematic simulation software that already has support for Unity and Input/output communications for TwinCAT. However, this was not possible, because Mevea does not take into account the colliders inside Unity if the map from the game is not imported on Mevea's software, and therefore Mevea's software was not used in the project.

Secondly, we tested if it is possible to write the rotational values of the ambulance into a separate .txt file and then read it with TwinCAT's "FileRead" function, but it was discovered that the "FileRead" function only opens the file but would not take account of the change in the value, or the file would have to be opened again and again constantly, and this would not be efficient enough.

Finally, it was decided to use TwinCAT ADS (Automation Device Specification), so that we were able to write the data from Unity straight on TwinCAT's ADS variables. Unity did not, however, support TwinCAT ADS, but a German company called Game4Automation had made an asset for Unity, which has a TwinCAT ADS built in.

With a Game4Automation asset, we managed to import signals from the PLC-program. With the help of a script, we were able to write the Euler angles, speed and acceleration of the ambulance on the signals. As a result, these signals will be updated into the PLC-program in real-time.

Keywords	PLC-program, ADS, Ambulance simulator, TwinCAT

Sisällys

1 Johdanto

Tämän insinöörityön aiheena on ambulanssisimulaattori. Simulaattoreilla tarkoitetaan yleensä laitteita, joilla luodaan simuloitu kokemus jostain todellisuuden kohteesta. Simulaattoreita on lukuisia, mm. lento-, taistelu-, pimeänajo- ja junasimulaattoreita. Simulaattoreiden mahdollisesti tärkein käyttötarkoitus on kouluttaa ihmisiä tiettyyn tehtävään turvallisesti ja edullisesti.

Insinöörityössä on tarkoituksena käyttää peliä, jossa voi ajella ambulanssilla ympäri Meilahtea, ja jonka ovat kehittäneet aikoinaan Helsinki XR-Centerin opiskelijat. Pelin jatkeeksi on olemassa fyysinen simulaattorialusta, johon on tarkoitus tulevaisuudessa asentaa vanhan ambulanssin runko. Simulaattorin tarkoituksena on liikuttaa pelialustaa pelin mukana realistisesti, mikä luo immersiivisen ympäristön, ikään kuin oikeasti ajaisi ambulanssia ympäri Meilahden aluetta. Valmista tuotetta voisi hyödyntää mm. ensihoitajien ajokoulutuksessa tutustuttaen heitä ajamaan ja navigoimaan sairaalan lähiympäristössä. Peliin ollaan tulevaisuudessa lisäämässä myös muita ominaisuuksia ja toiminnallisuuksia, mm. liikennettä ja liikennesääntöjä. Mikään ei myöskään estä, että pelimaailman maantieteellistä aluetta laajennettaisiin jatkossa.

Insinöörityön tarkoituksena on tutkia, miten saada opiskelijoiden Unity-pelimoottorilla luoma ambulanssipeli kommunikoimaan TwinCAT-ohjelmiston kanssa, jotta ajotuolia voitaisiin ohjata reaaliajassa pelin sisällä tapahtuvilla asennon muutoksilla. Tavoitteena on löytää sopiva kommunikointiratkaisu Unity-pelimoottorin ja TwinCAT-ohjelmiston välille. Jotta tuoli liikkuisi reaaliajassa peliin nähden, kommunikointirajapinnan vaatimuksina on, että pelissä tapahtuvat ambulanssin asennonmuutokset ovat luettavissa TwinCAT-ohjelmistossa ja tietojen päivitysten välissä olisi mahdollisimman pieni vasteaika. Ajotuolin ohjausjärjestelmään kuuluu kuusi pneumaattista lihasta, jotka pullistuvat ja supistuvat painetta lisättäessä. Logiikkaterminaali ja -ohjelma toimivat tietokoneella TwinCAT-ohjelmistossa.

2 Työssä käytetyt ohjelmistot ja laitteet

Tässä luvussa käsitellään insinöörityössä käytettyjä ohjelmistoja ja laitteita sekä niiden käyttötarkoitusta tässä työssä.

2.1 Unity

Projektissa Unity-pelimoottoria käytettiin ambulanssipelin kehitysalustana, koska peli oli alun perin tehty sitä käyttäen. Pelimoottorilla oli mahdollista muokata peliä itsessään, luoda skriptejä *Visual studio* -ohjelmiston avulla sekä hallinnoida lisäosia. Pelimoottori on pelin pohjana toimiva ohjelmisto, joka tarjoaa kehittäjille tarvittavat ominaisuudet, jotta pelien kehittäminen olisi nopeaa ja tehokasta. Pelimoottorit ovat vastuussa näytölle piirtämisestä, kosketuspinnoista, muistin käytöstä ja lukuisista muista asetuksista. (Unity 2020a.)

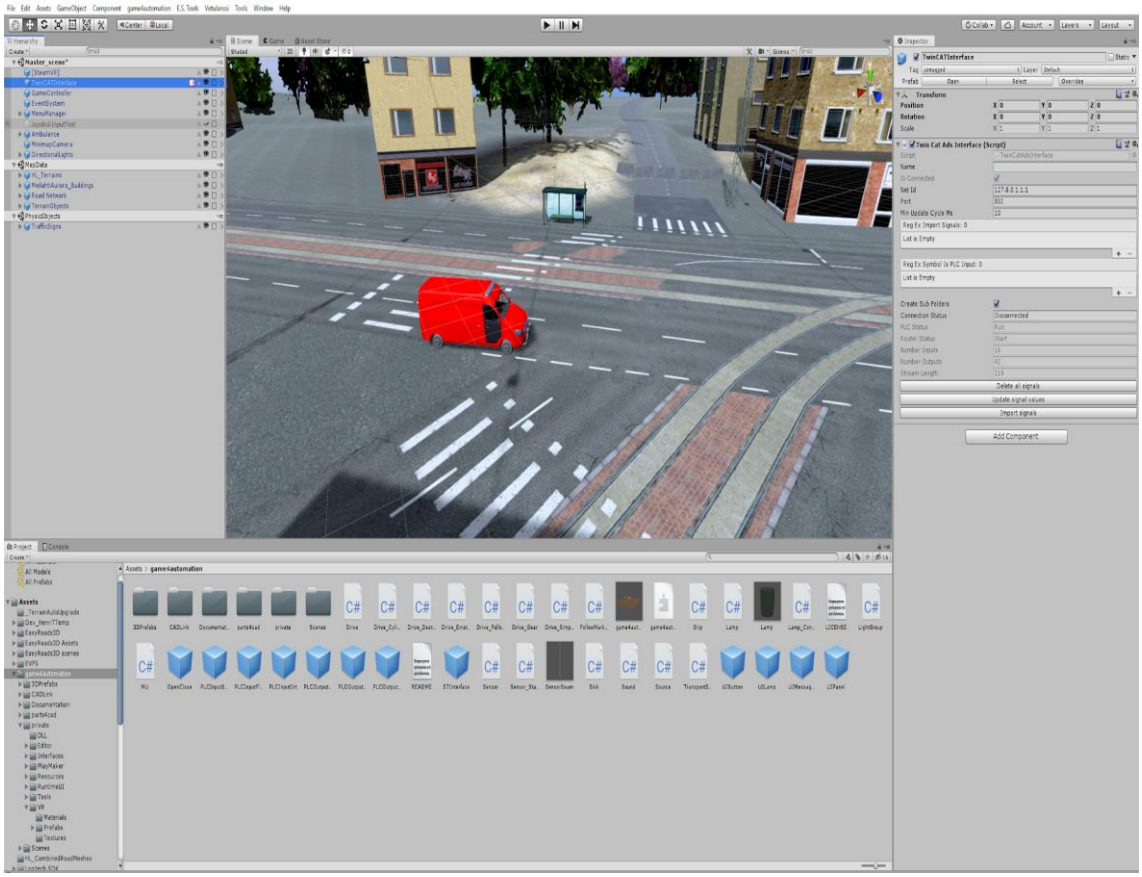
Unity on Unity Technologiesin kehittämä pelimoottori, jonka ensijulkaisu oli 8.6.2005. Unity palkittiin Apple's 2006 Worldwide Developer's Conferencessa, kun se oli esitelty ensimmäisenä täysin itsenäisenä pelimoottorina, kykenevänä grafiikkaan ja fysiikkalaskuihin. Unity sisälsi valmiiksi pelin käyttäytymistapoja, ja oli myös laajennettavissa iphonelle ennen muuta kilpailua kyseisellä alustalla. Vaikka Unity kasvoi iphonen myötä, tänä päivänä sillä tehdyt pelit ovat suosittuja kaikilla alustoilla. (Axon 2016.)

Unity-pelimoottorilla on mahdollista kehittää kaksi- ja kolmiulotteisia pelejä selaimille, mobiililaitteille, konsoleille ja tietokoneille. Sillä voidaan luoda virtuaali- ja lisätyn todellisuuden sovelluksia. Käsite virtuaalitodellisuus (Virtual Reality, lyh. VR) tulee sanoista "la réalité virtuelle", kun vuonna 1938 ranskalainen Antonin Artaud kuvasi sillä teatterin vaikutuskeinoja novellikokoelmassaan. Vuonna 1962 tuli käyttöön ensimmäinen virtuaalitodellisuuslaite *Sensorama*. Laite esitti lyhyitä 3D-kokemuksen tarjoavia laajakuvafilmejä, elämystä tehostettiin stereoäänillä, tuulettimilla ja tuoksuilla samoin kuten monille tutussa *Cinema 4D:ssä.* Nykyisin virtuaalitodellisuus on yleisintä virtuaalilaseissa. Tällaisia laseja tai kypäriä ovat mm. *HTC VIVE*, *Samsung Gear*, *Google Cardboard*, *PlayStation VR* ja *Oculus Rift*. (Arvanaghi & Skytt 2016.)

Virtuaalitodellisuus voidaan luoda yllättävän yksinkertaisella tekniikalla. Molemmille silmille on oma näyttönsä, millä saadaan syvyys- ja 3D-vaikutelma. Laseissa on myös kiihtyvyysanturi, joka lukee pään liikkeet ja mahdollistaa virtuaalimaailmassa ympärille katsomisen. Virtuaalitodellisuutta käytetään muuhunkin kuin pelaamiseen ja videoiden katseluun mm. taistelu- ja astronauttien kuukävelykoulutuksissa. (Arvanaghi & Skytt 2016.)

Pelimoottoriin voidaan ladata sekä ilmaisia että maksullisia lisäosia sen sisäisestä verkkokaupasta (*Asset Store*). Projektissa käytettiin Game4Automation Professional -lisäosaa. Unity-pelimoottorissa C# on ainoa tuettu ohjelmointikieli. Tunnettuja Unity-pelimoottorilla luotuja pelejä ovat mm. *Cuphead* ja *Hearthstone*. (Unity 2020b.)

Projektissa käytetty Unity-versio oli *Unity 2019.2.5f1 Personal* (Kuva 1).

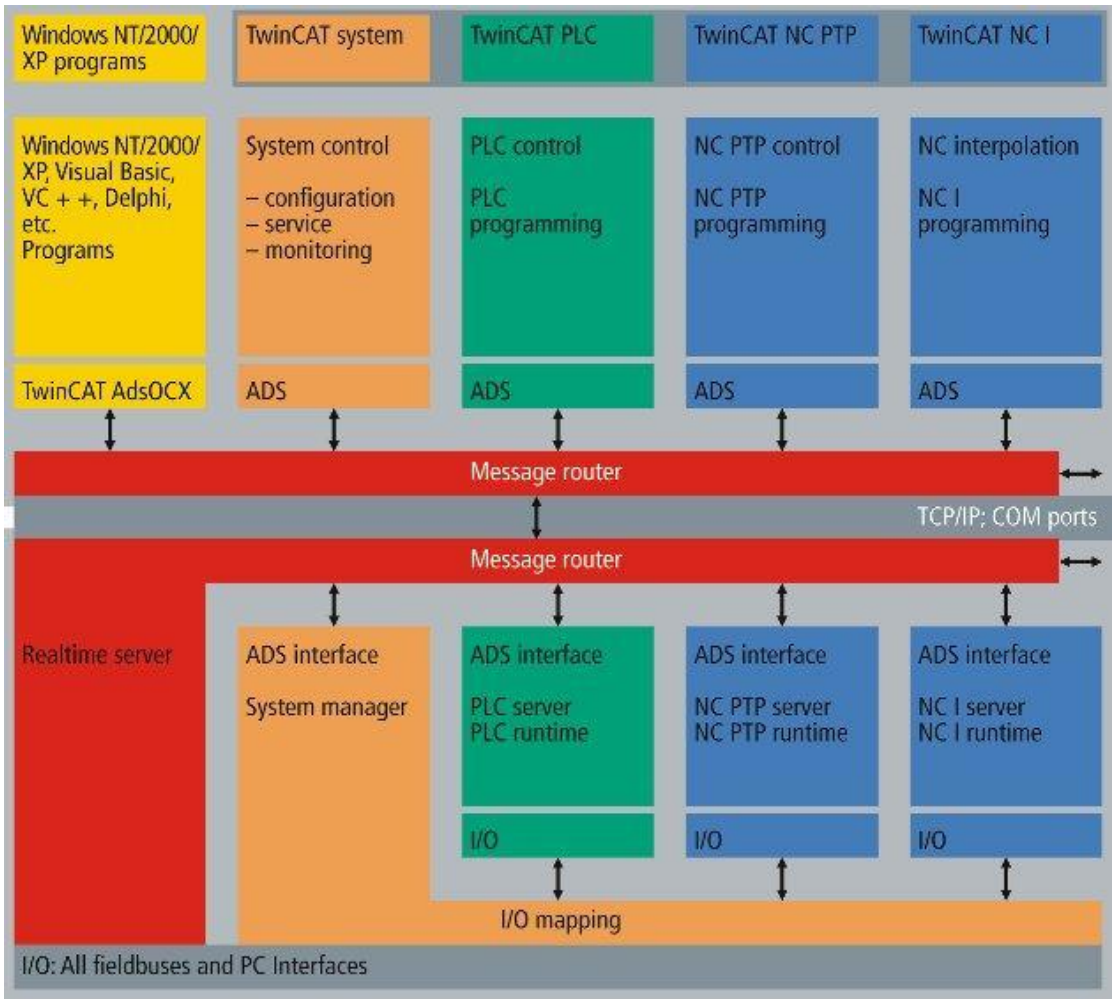


Kuva 1. Projektin perusnäkymä Unity-pelimoottorissa

2.2 Beckhoff TwinCAT

TwinCAT-ohjelmisto tekee melkein mistä tahansa tietokoneesta reaaliaikaisen ohjaimen, jossa on ohjelmoitava logiikka eli PLC (Programmable Logic Controller). TwinCAT-ohjelmistolla voidaan luoda ja testata PLC-projekteja. TwinCAT-järjestelmän arkkitehtuuri antaa kohdella yksittäisten ohjelmistojen moduuleja (esim. TwinCAT PLC, TwinCAT NC) itsenäisten laitteiden tavoin. Jokaiselle tehtävälle on oma ohjelmistomoduulinsa (serveri tai klientti). (Beckhoff 2019.)

Näiden moduulien väliset viestit käytetään yhdenmukaisen Automation Device Specification (ADS) -rajapinnan, tai *message routerin*, kautta (Kuva 2). ADS kuvaa laite- ja kenttäväyläriippumatonta rajapintaa, joka hallitsee pääsyä ADS-laitteisiin. Tämä *message router* hallitsee ja jakaa kaikki järjestelmän viesti- ja TCP/IP-yhteydet. TwinCATin *message routereita* on jokaisessa TwinCAT-tietokoneessa ja jokaisessa Beckhoff BCxxxx Bus Controllerissa. TwinCAT-ohjelmistolla luotiin ja ajettiin PLC-ohjelmaa, se myös yhdistää ja ohjaa terminaalin välityksellä pneumatiikan komponentteja. (Beckhoff 2020.)



Kuva 2. TwinCAT ADS laitekonsepti (Beckhoff 2020)

2.3 Game4Automation

Game4Automation on runkorakenne, jonka avulla voidaan luoda suorituskykyisiä digitalisia kaksosia. Ne ovat tuotteesta tai systeemistä luotuja identtisiä virtuaalisia malleja. Digitaalisella kaksosella (digital twin) viitataan fyysisen koneen tai kokonaisen tuotannon virtuaaliseen, mahdollisimman yksityiskohtaiseen, malliin. Ne usein yhdistetään 3D-mallinnukseen ja simulaatioon, mutta siitä ei ole kyse vaan ennemminkin tuotteen elinkaaresta ja takaisinkytkennästä. Kaksosten suurimmat hyödyt tulevat esille tuotekehityksessä. Niiden avulla voidaan välttää turhien prototyyppien rakentamista, vähentää virheiden syntymistä ja siten vähentää kustannuksia. (Keränen 2018.)

Digitaalisten kaksosten luonti pelimoottoreilla on yleistymässä. Suuret yksityiskohdat ovat peleissä yleisiä, joten pelimoottorit käsittelevät niitä paremmin kuin teollisen suunnittelun ohjelmistot. Tällöin malleista saadaan realistisemman näköisiä. Pelimoottorien valaistuksen ja varjojen avulla autetaan asiakkaita saamaan realistisempi kuva tuotteesta, kun tuotteet voidaan asettaa myös virtuaaliseen ympäristöön. Suuri osa pelimoottoreista tukee myös virtuaalitodellisuutta ja lisättyä todellisuutta (augmented reality), jotka tuovat suuria etuja tuotteiden ja palvelujen testaamiseen ja esittelemiseen. Digitaalisia kaksosia voidaan käyttää esimerkiksi myös virtuaalitodellisuuden avulla laitekoulutuksessa, tuotantolinjojen ergonomian testauksissa tai työpaikan turvallisuuden kehittämisessä. Unity-pelimoottorilla digitaalisten kaksosten luomisessa muita hyötyjä on mm. sen avoin arkkitehtuuri. Sitä myös päivitetään jatkuvasti pelitrendien mukana, mikä tukee käyttäjäkokemuksen parantamista. Pelimoottorit ovat myös erittäin muokattavissa ja niihin on saatavissa suuret määrät lisäosia ja ominaisuuksia. (Dumaresq 2018.)

Game4Automation-lisäosa parantaa Unity-käyttöliittymää sopivammaksi mekaaniseen ja automaatiosuunnitteluun. Se sisältää useita komponentteja automaatiolaitteille kuten, käyttölaitteita ja sensoreita, lisäksi automaatio-käyttöliittymiä joihin nämä komponentit voidaan yhdistää. Game4Automation sisältää rajapinnat *Beckhhoff TwinCAT, Simit, PLCSim-Advanced* ja *TCP-IP*:n kautta Siemens-laitteistoihin, kuten *S7-300, S7-1500, Sinumerik ja Simotion.*

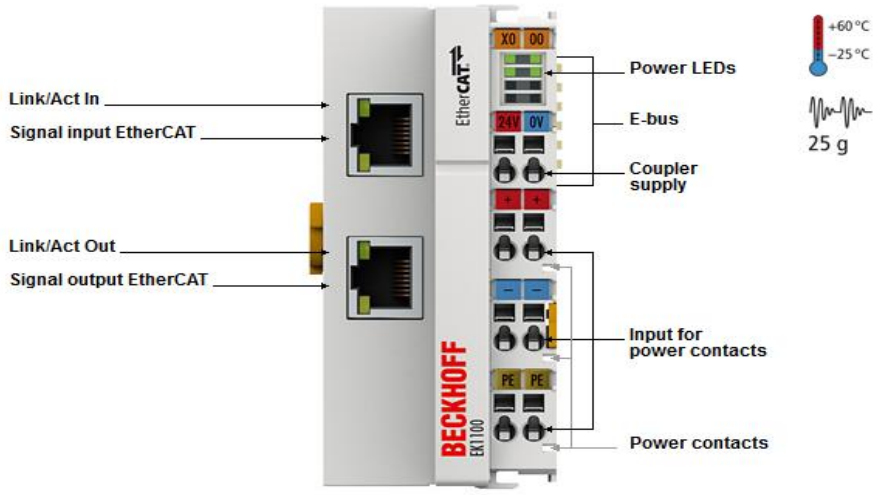
Yhteensopivia CAD-käyttöliittymiä ovat *Solidworks, Solidedge, NX, JT, Parasolid, Catia, ProE-Creo ja Autocad*. Game4Automation hankittiin insinöörityötä varten sen OPC UA- ja TwinCAT ADS -ominaisuuksien takia. TwinCAT ADS Interface toimi, joten OPC UA:ta ei käytetty. TwinCAT ADS Interface toimi rajapintana, joka mahdollisti kommunikoinnin TwinCAT- ja Unity ohjelmistojen välillä. (Game4Automation 2020.)

2.4 Beckhoff EK1100 -kytkinlaite ja terminaalit

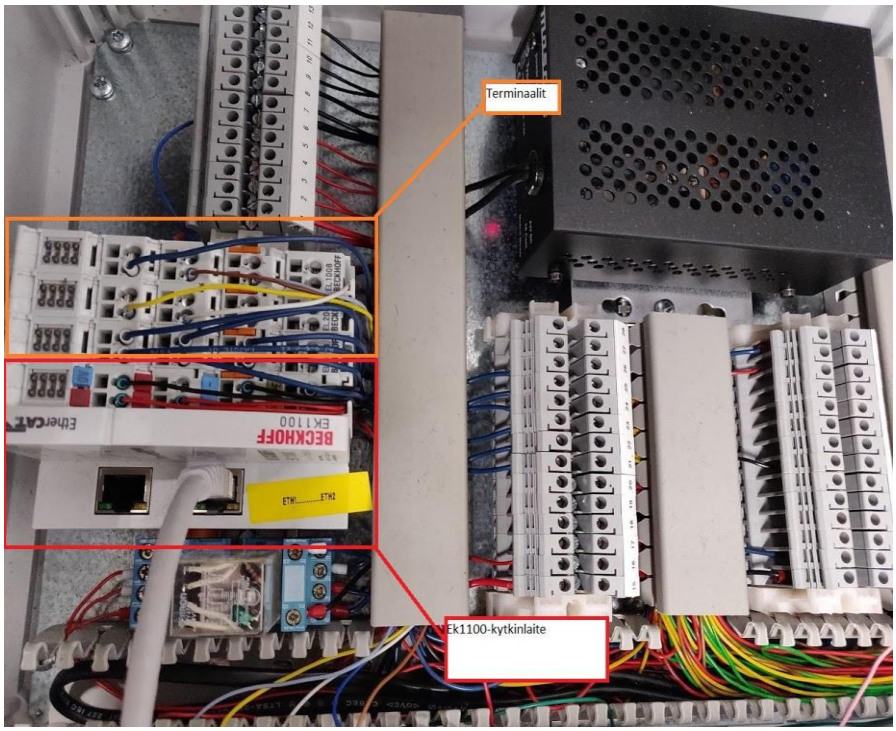
EK1100-kytkin yhdistää EtherCATin EtherCAT-päätteisiin (ELxxxx). EtherCAT on Ethernet-liitäntä automaatio-ohjaukseen. Yksi järjestelmäkokonaisuus koostuu EK1100-kytkimestä, mistä tahansa määrästä EtherCAT-päätelaitteita ja väyläpäätteestä. Kytkin muuntaa läpimenevät viestit Ethernet 100BASE-TX:stä E-väylän signaaliksi. EK1100-kytkinlaite liitetään verkkoon ylemmän Ethernet-liitännän kautta. Alempaa RJ45-liitäntää voidaan käyttää muiden EtherCAT-laitteiden kytkemiseen samaan säikeeseen. EtherCAT-verkossa EK1100-kytkin voidaan asentaa mihin tahansa Ethernet-signaalinsiirto-osioon (100BASE-TX) paitsi suoraan kytkimeen. Kytkimet EK9000 ja EK1000 sopivat asennettaviksi kytkimessä. (Beckhoff 2018.) (Kuvat 3-5.)

Technical data	EK1100
Task within EtherCAT system	coupling of EtherCAT Terminals (ELxxxx) to 100BASE-TX EtherCAT networks
Data transfer medium	Ethernet/EtherCAT cable (min. Cat. 5), shielded
Distance between stations	max. 100 m (100BASE-TX)
Number of EtherCAT Terminals	up to 65,534
Protocol	EtherCAT
Delay	approx. 1 µs
Data transfer rates	100 Mbit/s
Configuration	not required
Bus interface	2 x RJ45
Power supply	24 V DC (-15 %/+20 %)
Current consumption from U_S	70 mA + (∑ E-bus current/4)
Current consumption from U_P	load
Current supply E-bus	2000 mA
Power contacts	max. 24 V DC/max. 10 A
Electrical isolation	500 V (power contact/supply voltage/Ethernet)
Operating/storage temperature	-25…+60 °C/-40…+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protect. class/installation pos.	IP 20/variable
Approvals	CE, UL, Ex, IECEx

Kuva 3. EK1100:n tekniset tiedot (Beckhoff 2018)



Kuva 4. EK1100:n liitännät (Beckhoff 2018)



Kuva 5. EK1100 tuolin ohjausjärjestelmässä

3 Automaatiokommunikaation toimintaperiaate

Tässä luvussa käydään läpi simulaattorin toiminnan vaatimukset, sekä erilaisia Unity-pelimoottorin ja TwinCAT-ohjelmiston välisiä kommunikointiratkaisuja. Tämän lisäksi avataan sitä, kuinka ambulanssin tietoja kirjoitetaan skriptien avulla, sekä esitellään parhaaksi nähdyt ratkaisut ja toiminnallinen kokonaisuus.

3.1 Vaatimukset

Työtä varten ambulanssipelistä pitäisi saada toimiva versio ja luoda PLC-ohjelma halutuilla muuttujilla sekä kehittää näiden kahden välille tapa kommunikoida. Kun PLC-ohjelma ja pelin välinen kommunikointi on toteutettu, luodaan skripti, jolla saadaan ambulanssin asento- ja nopeusarvot. Tämän jälkeen tarvitaan tapa kertoa asento- ja nopeusarvot PLC-ohjelmalle kommunikointirajapinnan välityksellä.

3.2 Toteutustavat

Suunnitelmana oli asentaa Unity-pelimoottorista sama versio, jota oli käytetty pelin kehityksessä ja asentaa Mevea-ohjelmisto luomaan digitaalinen kaksonen kinematiikkasimulointia varten Unity-pelimoottorissa. Ensin tarkoitus oli kokeilla ajaa toisten opiskelijoiden jo kehittämää *Unmanned ground vehicle* (UGV) -digitaalista kaksosta, joka toimi Mevea-ohjelmiston ja Unity-pelimoottorin kanssa yhteistyössä. Kun nämä kaksi olisi saatu toimimaan voisi ambulanssipelin asentaa Unity-pelimoottorin kehitysympäristöön ja koettaa saada sitä toimimaan Mevea-ohjelmiston kanssa. Tässä vaiheessa todettiin, ettei olisi mahdollista yhdistää ambulanssipeliä Mevea-ohjelmistoon ilman alkuperäistä pelimaailman mallia ja tämäkin olisi todennäköisesti ollut liian suuri tai yksityiskohtainen ja olisi vaatinut paljon suorituskykyä tietokoneelta. Toinen vaihtoehto oli luoda Unity-pelimoottorissa skripti, joka kirjoittaisi ambulanssin asento- ja nopeusarvoja tekstitiedostoon ja käyttää TwinCat ”FileRead”-funktiota tekstitiedoston lukemiseen. Kolmas vaihtoehto oli ostaa Unity-pelimoottorin kehitysympäristöön Game4Automation Professional -lisäosa, joka avaisi mahdollisuuden luoda skriptin, joka kirjoittaa ambulanssin asento- ja nopeusarvoja suoraan TwinCAT-ohjelmiston muuttujiin, käyttäen TwinCAT ADS- tai OPC UA -protokollaa. Lopulta päädyttiin käyttämään Game4Automation Professional -lisäosaa.

3.3 Asento- ja nopeusarvojen kirjoitus tekstitiedostoon

Insinöörityön alussa asennettiin Unity versio 2019.2.5f1, johon avattiin ambulanssipeli. Ensin luotiin skripti (C# -ohjelmointikielellä), joka kirjoitti ambulanssin transformaatio-arvoja erilliseen tekstitiedostoon (Kuva 6). Skripti kirjoitti onnistuneesti asentoarvoa erilliseen tekstitiedostoon ja päivitti sen joka kuvanvaihdon yhteydessä. Jälkeenpäin todettiin, että tekstitiedosto, jonka sisältö muuttuu jatkuvasti, ei ole sopiva ratkaisu. TwinCAT FileRead-funktio joutuisi avaamaan ja sulkemaan tiedostoa jatkuvasti, mikä lisäisi rasitusta ja viivettä sovellukseen. Testissä FileRead-funktio antoi vain staattisen arvon, jonka se ehti lukea tekstitiedostosta.

FileWrite.cs

Miscellaneous Files

FileWrite

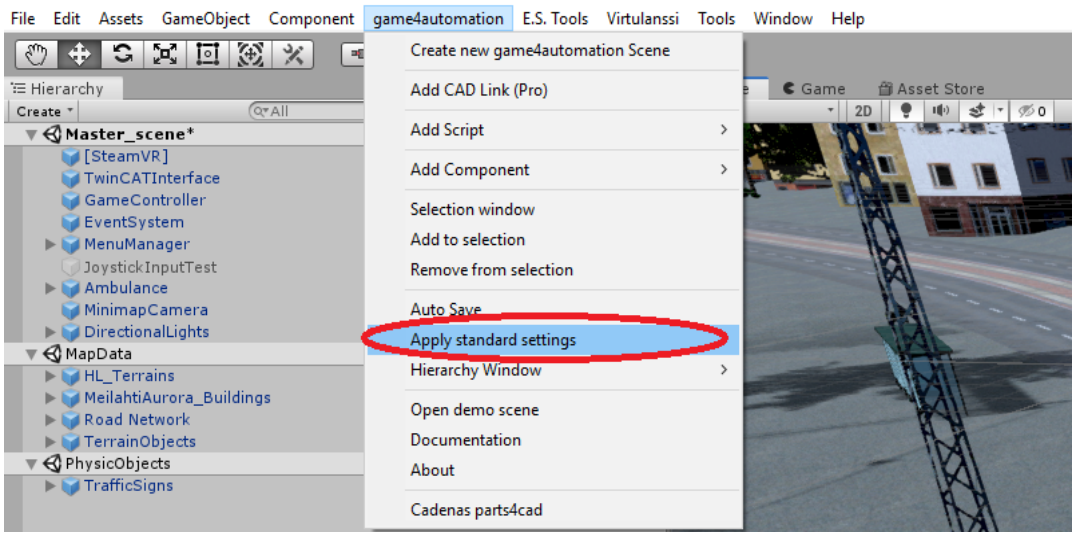
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using System.Text;



public class FileWrite : MonoBehaviour {
    void CreateText() {

            //Path of the file
            string path = Application.dataPath + "/Log.txt";
            //Create File if it doesn't exist
            if (!File.Exists(path))
            {
                File.WriteAllText(path,"");

            }
            //Content of the file
            string content = gameObject.transform.rotation.y + "\n";
        //Add some text to it
            File.WriteAllText(path, content);


    }
    // Start is called before the first frame update
    void Start()
    {
        CreateText();
    }



    // Update is called once per frame
    void Update()
    {
        CreateText();
    }
}
```

Kuva 6. Muuttujien kirjoitus erilliseen tekstitiedostoon (C#)

3.4 Game4Automationin asennus ja käyttöönotto

Tietoon tuli, että saksalainen yritys oli luonut Unity-pelimoottoriin lisäosan, joka voisi sopia työn sovellukseen. Päädyttiin ottamaan yhteyttä Game4Automationiin ja tultiin lopputulokseen, että heidän Professional-pakettinsa sopisi insinöörityön toteutukseen ja sitä voisi käyttää myös tulevaisuudessa muihin opetustarkoituksiin.

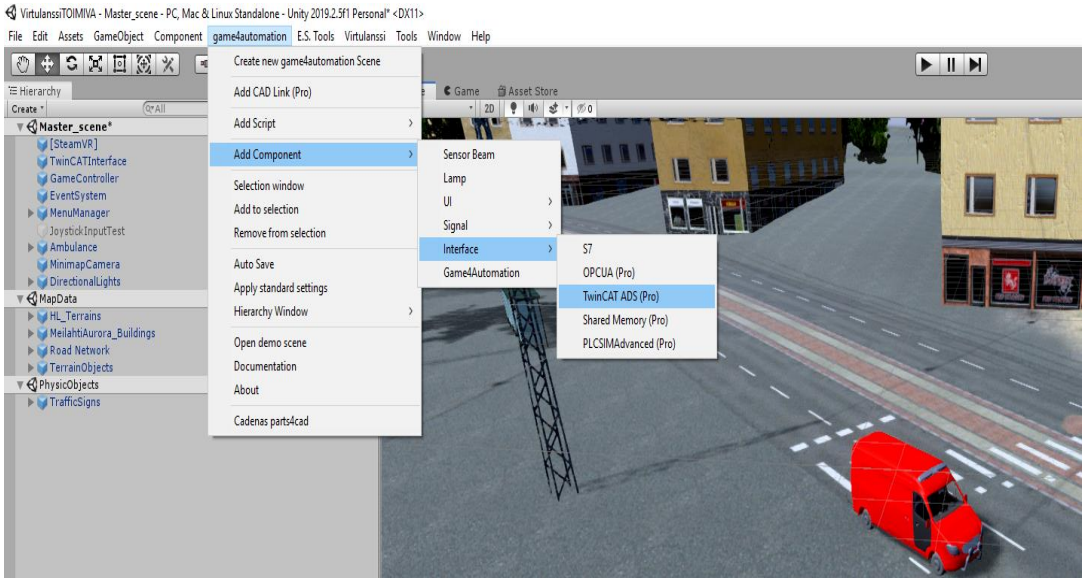
Game4Automation Professionalin lataamisen jälkeen asennuspaketti lisättiin projektiin avaamalla projekti Unity-pelimoottorissa, siirtymällä ”Asset”-pudotusvalikkoon ja klikkaamalla ”Import Custom Asset”. Lopulta valittiin tiedosto, joka oli ladattu, ja valittiin koko paketin sisältö painamalla ”All” -painiketta, minkä jälkeen tuotiin paketti projektiin painamalla ”Import Asset” -painiketta. Asentamisen jälkeen voitiin lisätä Game4Automationin oletusasetukset hakemalla ”Apply standard settings” game4automation-pudotusvalikosta. (Kuva 7.)



Kuva 7. Game4Automation oletusasetusten lisäys

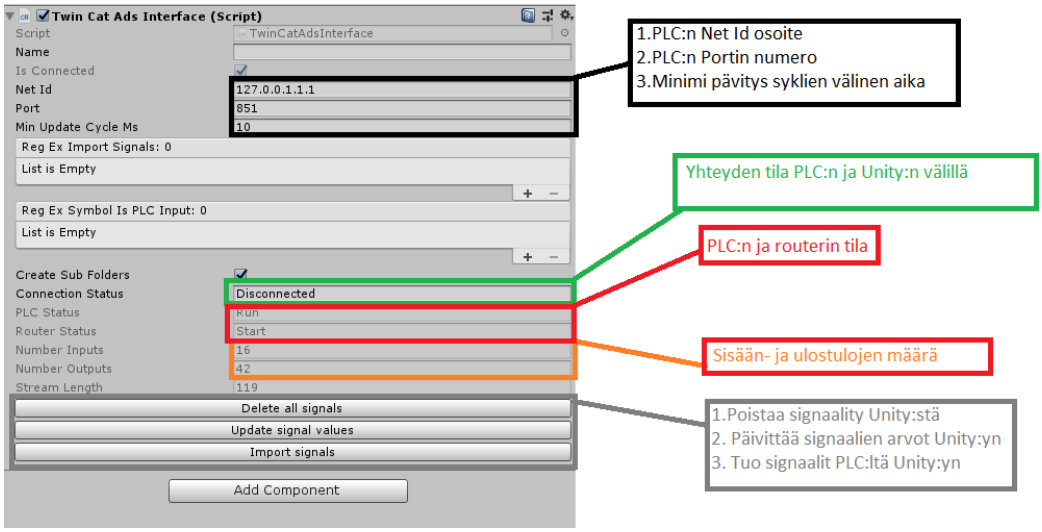
3.5 Kommunikointirajapinnan luominen ja yhdistäminen

Ohjelmistot oli asennettu, mutta ei ongelmitta: Game4Automationia ei onnistuttu lisäämään projektiin halutulla tavalla, joten internetin välityksellä pidettiin videoneuvottelu lisäosan kehittäjän Thomasin kanssa. Lisäosan lisäyksen jälkeen ”Compile” -asetukset eivät olleet oikein, kollision-tasot projektissa olivat vaihtuneet vääriksi ja ”Postprocessing” ei ollut sisällytetty ”Assembly Definitions” -asetusten alle. Nämä kuitenkin saatiin korjattua ja nykyisin kyseisiin ongelmiin löytyy myös ohjeet Game4Automation kotisivuilta dokumentointi-välilehdestä (https://game4automation.com/). (Kuva 8.)



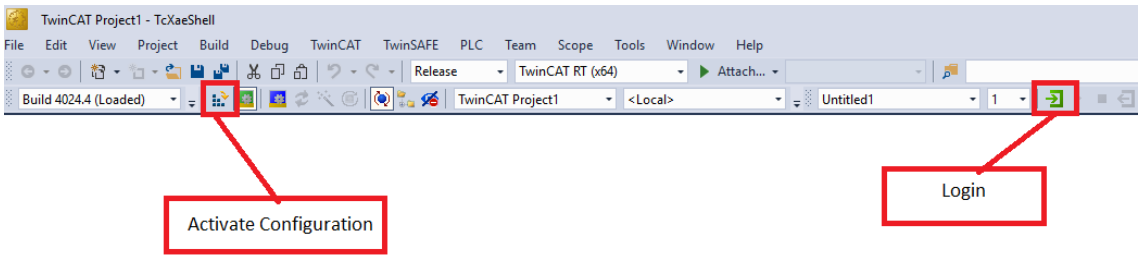
Kuva 8. TwinCAT ADS-käyttöliittymän lisäys työhön

Kun lisäosa toimi Unity-pelimoottorin kehitysympäristössä ongelmitta, pystyi projektiin lisäämään TwinCAT ADS -käyttöliittymäkomponentin Game4Automation-pudotusvalikosta: “Add Component” → “Interface” → “TwinCAT ADS (Pro)”. Lisätty komponentti näkyy projektin rakennepuussa ja sitä klikkaamalla, aukeaa sen sisältö tarkkailtavaksi ”Inspector”-ikkunaan (Kuva 9).



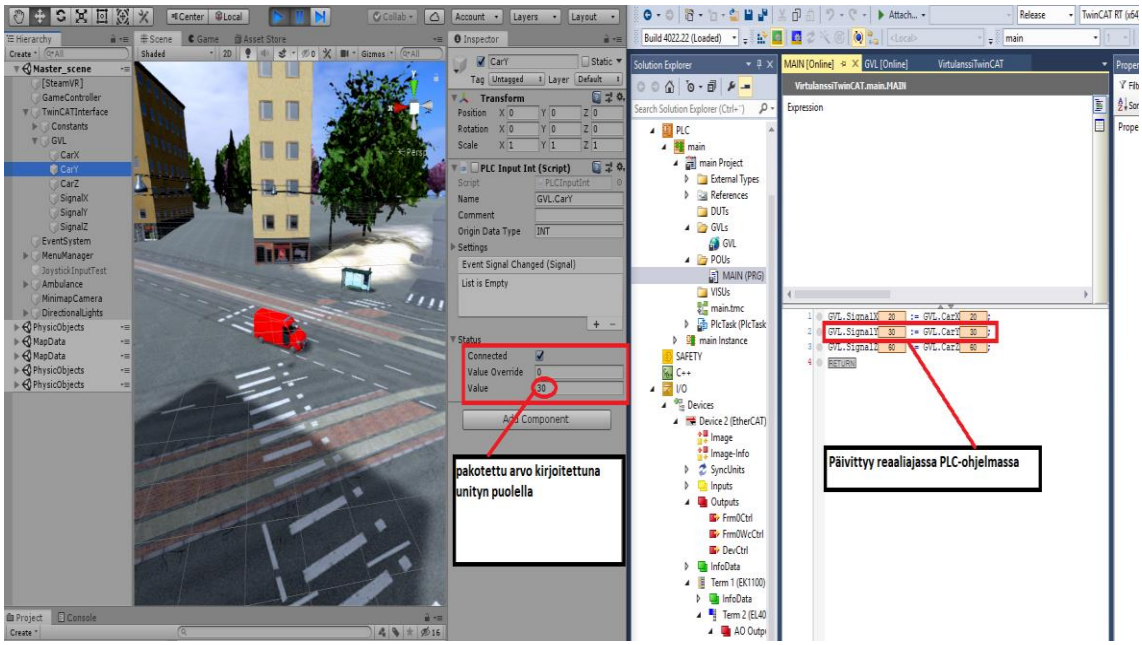
Kuva 9. TwinCAT ADS -käyttöliittymä ”Inspector”-ikkunassa

PLC-ohjelman käynnistys tapahtuu TwinCAT-ohjelmistosta ”Activate Configuration” -painikkeella, minkä jälkeen ohjelman saa ladattua PLC:n muistiin painamalla ”Login”-painiketta (Kuva 10).



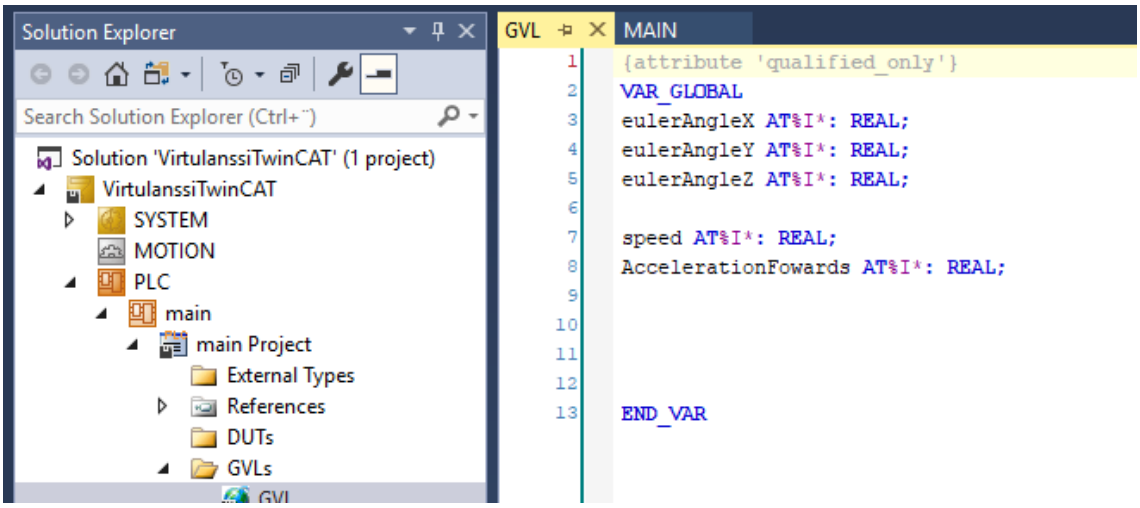
Kuva 10. TwinCAT-ohjelmiston painikkeet

Tämän jälkeen on mahdollista yhdistää TwinCAT Unity-pelimoottoriin, kirjoittamalla ohjelmiston ”Net Id” Unityn TwinCAT-käyttöliittymään ”Inspector”-ikkunassa kohtaan ”Net Id” ja porttinumero kohtaan ”Port”. Tämän jälkeen ”Connection Status” (eli yhteyden tila) muuttuu ”Connected” (eli yhdistetyksi). Kun ohjelmistot on yhdistetty, voi signaalit (eli muuttujat) tuoda TwinCAT-ohjelmiston puolelta Unity-pelimoottoriin painamalla ”Import signals”. Tämän seurauksena muuttujat tulevat näkyviin TwinCAT-käyttöliittymäkomponentin alle projektin rakennepuuhun. Signaalien toimintaa voi testata kirjoittamalla signaalien ”value”-kohtaan pakotettuja arvoja ja tarkistamalla päivittyvätkö ne PLC-ohjelman puolella. (Kuva 11.)



Kuva 11. Pakotettujen arvojen kirjoitus Unitystä PLC-ohjelmaan

PLC-ohjelmaan luotiin Unity-pelimoottorista tuleville sisään- ja ulostuloille omat muuttujat (eulerAngle- X, Y, Z, speed ja AccelerationFowards). Unity-pelimoottoriin linkitettyjen muuttujien täytyy olla sisääntuloja (ohjelmassa “AT%I*”), jotta Unity pystyy kirjoittamaan niihin. Silloin, kun halutaan kirjoittaa Unity-pelimoottorista tuodut arvot fyysisille laitteille, ne tulee määrittää ulostuloiksi. Ulostuloilla on myös mahdollista kirjoittaa Unity-pelimoottorin sisälle. Muuttujat määritettiin reaaliluvuiksi “REAL”, jotta saataisiin arvot desimaalien tarkkuudella, toisin sanoen mahdollisimman tarkat lukemat. (Kuva 12.)



Kuva 12. PLC-ohjelman muuttuja kommunikoinnin testaukseen

3.6 Asento- ja nopeusarvojen päivittäminen signaaleihin

Ambulanssin peliobjektille luotiin signaalien sisääntuloja varten ”Behavior model”-skripti eli käyttäytymisen malli. Skriptissä käytettiin Game4Automation-lisäosan kirjastoa. Skriptille täytyi ilmoittaa, että kirjastoa halutaan käyttää, ja se tapahtui kirjoittamalla koodin eteen: ”using game4automation;”. Pelistä haluttiin tuoda ulos ambulanssin Eulerin kulmat: kallistuminen (roll), kääntyminen (yaw) ja nyökkääminen (pitch). Eulerin kulmat saa Unityn peliobjektista rigidbodystä komennolla: ”OmaKappale.transform.eulerAngles.x/y/z”. (Kuva 13.)

```
X = rb.transform.eulerAngles.x; //Pitch
Y = rb.transform.eulerAngles.y; //Yaw
Z = rb.transform.eulerAngles.z; //Roll
```

Kuva 13. Koodi, jolla Eulerin kulmat saatiin

Ambulanssista haluttiin myös tuoda nopeus ja kiihtyvyys. Kappaleen kulkusuunnan nopeus haetaan pelistä komennolla: ”OmaKappale.velocity.magnitude” ja keskikiihtyvyys saadaan laskemalla: $a_k = \frac{\Delta v}{\Delta t} = \frac{v_2 - v_1}{t_2 - t_1}$ (Δv = nopeuden muutos, Δt = ajan muutos) (Kuvat 14–15). Koodissa ajan muutoksena käytettiin: Time.fixedDeltaTime eli väli sekunteina, jolloin fysiikka ja muut kiinteät kuvataajuuden päivitykset suoritetaan (Unity 2019). Nopeuden muutos saatiin ottamalla kappaleen hetkellinen nopeus ja erotettiin siitä aikaisemman kuvan päivityksen nopeuteen.

```
Speed.Value = rb.velocity.magnitude;
```

Kuva 14. Koodi, jolla kappaleen kulkusuunnan nopeus saatiin

```
Acceleration.Value = (sp - lastX) / Time.fixedDeltaTime;
```

Kuva 15. Koodi, jolla kiihtyvyys laskettiin (sp = nopeus, lastX = viimeisin tallennettu nopeus)

Signaalien yhdistämistä varten luotiin julkiset (public) muuttujat käyttäen Game4Automation-lisäosan kirjastosta löytyvää komentoa: ”PLCInputFloat”, niihin voitiin sitten raahata signaalit Unity-pelimoottorin puolella. (Game4Automation 2019.) (Kuva 16.)

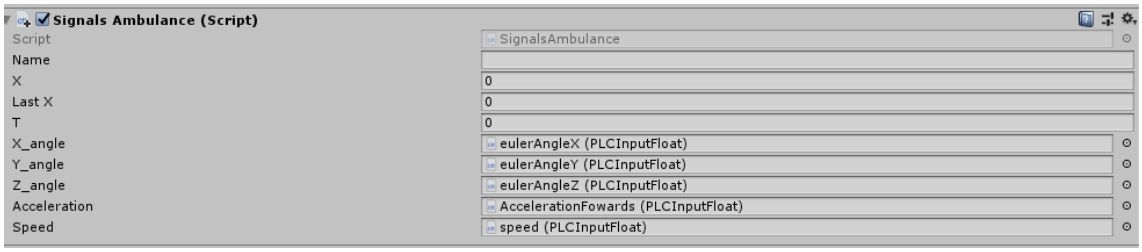
```
//PLC Inputs
public PLCInputFloat x_angle;
public PLCInputFloat y_angle;
public PLCInputFloat z_angle;
public PLCInputFloat Acceleration;
public PLCInputFloat Speed;
```

Kuva 16. Julkiset ”PLCInputFloat”-muuttujat

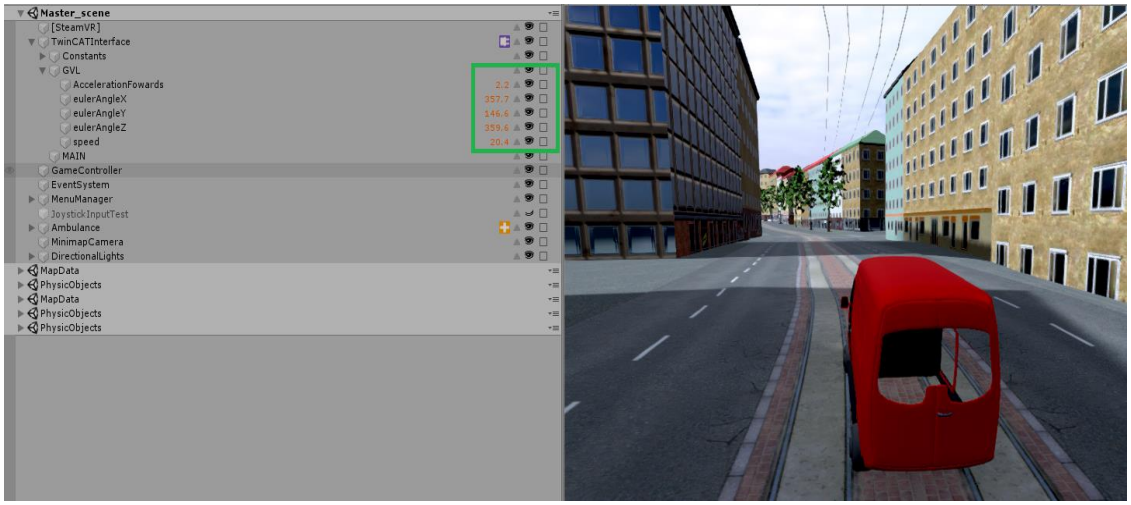
```
using UnityEngine;
using game4automation;
using System;


//[RequireComponent(typeof(PLCInputFloat))]
0 references
public class SignalsAmbulance : BehaviorInterface
{

    //Variables
    public float X = 0;
    float Y = 0;
    float Z = 0;
    float sp;
    public float lastX;
    public float T;

    //PLC Inputs
    public PLCInputFloat x_angle;
    public PLCInputFloat y_angle;
    public PLCInputFloat z_angle;
    public PLCInputFloat Acceleration;
    public PLCInputFloat Speed;

    2 references
    public void xval()
    {

        Time.timeScale = 1f;


        // acceleration = (rigidbody.velocity - lastVelocity) / Time.fixedDeltaTime;

        //Get eulerAngles from Rigidbody (Ambulance)
        Rigidbody rb = GetComponent<Rigidbody>();
        X = rb.transform.eulerAngles.x; //Pitch
        Y = rb.transform.eulerAngles.y; //Yaw
        Z = rb.transform.eulerAngles.z; //Roll
        Speed.Value = rb.velocity.magnitude;
        sp = rb.velocity.magnitude;


        //Assigning PLC Output "Value"
        x_angle.Value = X;
        y_angle.Value = Y;
        z_angle.Value = Z;
        Acceleration.Value = (sp - lastX) / Time.fixedDeltaTime;



    }




    // Start is called before the first frame update
    0 references
    void Start()
    {
        Time.timeScale = 1f;

        xval();


    }


    // Update is called once per frame
    0 references
    void Update()
    {

        xval();
        T = Time.fixedDeltaTime; //making sure that timescale is normal when code runs


        lastX = GetComponent<Rigidbody>().velocity.magnitude;
    }

    0 references
    private void LateUpdate()
    {



    }
}
```

Kuva 17. Skripti kokonaisuudessaan asento- ja nopeusarvojen signaaleihin päivittämiseen

Kun skriptin lisää Unityssä johonkin peliobjektiin, tulevat sen julkiset muuttujat näkyviin ”Inspector”-ikkunaan (Kuva 18). Skriptin ulkoasuun päivittyvät myös muuttujan arvot. Game4Automation-lisäosan ”PLCInput/output”-tyyppisiin muuttujiin on mahdollista yhdistää tuodut signaalit, ja näin insinöörityössä signaalien yhdistäminen toteutettiin. Kun skripti ja signaalit oli yhdistetty, päivittyivät niihin arvot myös Unityn puolella tarkkailtaviksi (Kuva 19).



Kuva 18. Skriptin ulkoasu Unityssä

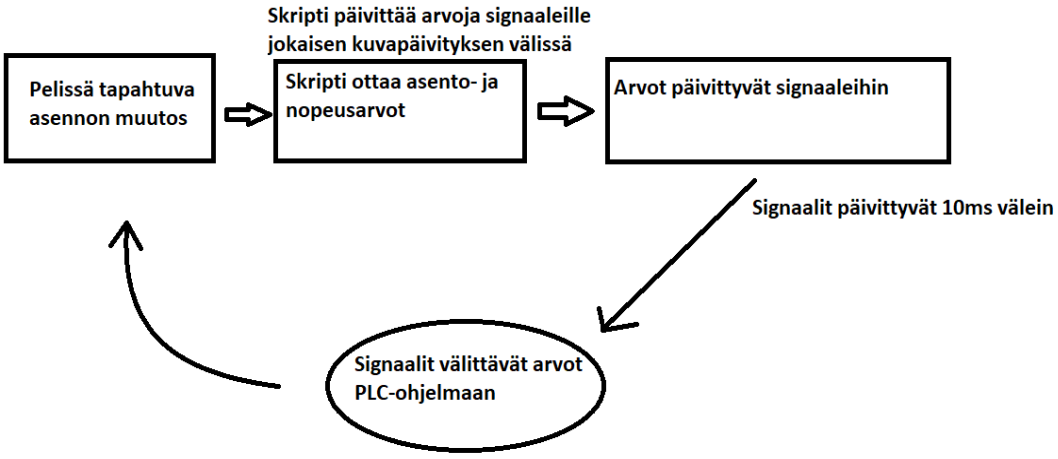


Kuva 19. Skripti yhdistettynä signaaleihin toiminnassa

3.7 Toiminnan kokonaisuus

TwinCAT-ohjelmistolla luotiin PLC-ohjelma. Unity-pelimoottoriin asennettiin ambulanssipeli ja siihen Game4automation-lisäosa. Lisäosan avulla pystyttiin yhdistämään TwinCAT-ohjelmistossa toimivan PLC-ohjelman muuttujat Unity-pelimoottorilla pyörivään ambulanssipeliin ja luomaan signaalit käyttäen TwinCAT ADS-väylää ja Game4Automation-lisäosan tarjoamaa TwinCAT ADS-käyttöliittymää.

Ohjelmien kommunikointi toimii siten, että pelissä tapahtuvat nopeuden, kiihtyvyyden ja asentojen muutokset otetaan ambulanssin pelikappaleesta (Rigid body) C#-ohjelmointikielellä kirjoitetulla skriptillä. Nopeus liikkumissuuntaan ja Eulerin kulmat saadaan suoraan olemassa olevilla komennoilla, kun taas kiihtyvyys lasketaan kiihtyvyyden kaavalla (löytyy luvusta 3.4). Skripti kirjoittaa arvot Game4Automation-lisäosalle tyypillisiin ”PLCInputFloat” -muuttujiin, joihin voidaan kiinnittää tuodut signaalit raahaamalla ne muuttujien päälle. Kiinnittämisen jälkeen signaalit päivittävät muuttujien arvot reaaliajassa PLC-ohjelmaan.



Kuva 20. Toiminnan havainnollistaminen

4 Yhteenveto

Insinöörityön tavoitteena oli keksiä sopiva kommunikointiratkaisu TwinCAT-ohjelmiston ja pelin välille, jotta olisi mahdollista liikuttaa fyysistä tuolia mahdollisimman pienellä viiveellä peliin verrattuna.

Aloittaessa kului paljon aikaa tarvittavan tietokoneohjelmistojen hankkimiseen, tutkimiseen, lisensointiin ja opetteluun. Aluksi ambulanssipelistä ei löytynyt toimivaa versiota, kunnes se saatiin yhdeltä peliä tekevistä opiskelijoista. Tämän jälkeen yritettiin yhdistää Mevean ohjelmistoa ja peliä, mikä loppujen lopuksi ei onnistunut, ja suunnitelmaa päädyttiin muuttamaan. Unity ei suoraan tukenut omien skriptien luomista PLC-ohjelman muuttujiin, joten päädyttiin käyttämään Game4Automation -lisäosaa. Tämäkään ei tapahtunut ongelmitta, mutta se oli ainoa toimiva ratkaisu.

Projektin tavoitteisiin päästiin hyvin, sillä Game4Automation -lisäosan avulla saatiin luotua kommunikointirajapinta pelin ja automaatio-ohjelmiston välille. TwinCAT-ohjelmisto saatiin yhdistettyä terminaaliin, joka puolestaan ohjaa pneumaattisia lihaksia. Lopputuloksena saatiin luotua skripti, joka kirjoittaa ambulanssin Eulerin kulmat, kiihtyvyyden ja kulkusuunnan nopeuden. Skriptin laskemat arvot taas päivittyvät PLC-ohjelman muuttujiin reaaliajassa.

Toiset opiskelijat olivat tehneet valmiiksi PLC-projektin, jonka pitäisi ohjata tuolia halutulla tavalla Unityn kirjoittaessa pelistä tulevat ambulanssin X-, Y- ja Z-arvot TwinCATiin. Tosin vaikka muuttujat saatiin pelistä PLC-projektiin, ei silti saatu PLC-projektia toimimaan muuttujien kanssa. Tämä vaatisi enemmän aikaa, muutoksia tulojen skaalauksiin ja mahdollisesti kokonaan uuden PLC-ohjelman laatimisen.

5 Kehitysehdotukset

Projektia voisi parantaa muuttamalla arvoja, skriptiä ja PLC-ohjelmaa yhteen sopiviksi, jotta tuoli liikkuisi pelin ambulanssin mukana realistisesti ja peli tuntuisi siltä, että ajaisi oikeaa ambulanssia. Tuolin lisäksi ratti ja polkimet tulisi vaihtaa tuolin alustaan. Peli tukee valmiiksi VR-laitteita, joten projektiin voisi myös lisätä mitkä tahansa SteamVR:ää tukevat VR-lasit, esim. HTC Vive Pro tai Valve Index. VR-teknologiaa käyttäen voitaisiin koettaa käyttää täysin virtuaalista rattia, jota käytetään VR-ohjaimilla, tai fyysistä pelirattia, jolla taas olisi mahdollista antaa pelistä ratille vastusta, joka saisi pelistä entistä realistisemman tuntuisen.

Myös pelin malleja voitaisiin muokata yksityiskohtaisemmaksi, lisäämällä ambulanssiin jousituksia, vaihteet ja muita ajamiseen vaikuttavia mekanismeja, jotta ambulanssi liikkuisi pelissä sulavammin.

6 Lähteet

Axon, Samuel. 2016. Unity at 10. Verkkoaineisto. <https://arstechnica.com/gaming/2016/09/unity-at-10-for-better-or-worse-game-development-has-never-been-easier/>. Luettu 01.02.2020.

Arvanaghi, Babak & Skytt, Lasse. 2016. Virtuaalitodellisuus – tulevaisuus on täällä tänään. Verkkoaineisto. <https://tieku.fi/teknologia/vempaimet/virtuaalitodellisuus>. Luettu 01.02.2020.

Baker, Maverick. 2016. How Do Game Engines Work. Verkkoaineisto. <https://interestingengineering.com/how-game-engines-work>. Luettu 01.02.2020.

Beckhoff. 2018. EK1100 | EtherCAT Coupler. Verkkoaineisto. <https://www.beckhoff.com/EK1100/>. Luettu 19.01.2020.

Beckhoff. 2019. TwinCAT – PLC and Motion Control on the PC. Verkkoaineisto. <https://www.beckhoff.com/twincat/>. Luettu 28.01.2020.

Beckhoff. 2020. TwinCAT ADS. Verkkoaineisto. <https://infosys.beckhoff.com/english.php?content=../content/1033/tcadscommon/html/tcadscommon_intro.htm&id=>. Luettu 28.01.2020.

Dumaresq, Scott. 2018. Digital Twins - Using gaming technology. Verkkoaineisto. <https://www.linkedin.com/pulse/digital-twins-using-gaming-technology-scott-dumaresq/>. Luettu 28.01.2020.

Game4automation. 2020. Kotisivu. Verkkoaineisto. <https://game4automation.com/en/>. Luettu 28.01.2020.

Game4Automation. 2019. Automation Interfaces. Verkkoaineisto. <https://game4automation.com/documentation/current/interface.html#Check-Signal-status-and-connections-to-Behavior-models>. Luettu 28.01.2020.

Keränen, Matti. 2018. Digitalisaatio. Verkkoaineisto. <https://www.tekniikkatalous.fi/uutiset/digitaalinen-kaksonen-on-iotn-seuraava-vaihe-tuottopotentiaalia-on-viela-vaikea-nayttaa/b628a399-2c0b-3091-bb32-f3da804486b>. Luettu 28.01.2020.

Landefeld. MPPES-3-1/8-6-420. 2020. Verkkoaineisto. <https://www.landefeld.com/artikel/en/mppes-3-18-6-420-187353-prop-press-reg-/OT-FESTO012915>. Luettu 28.01.2020

Unity. 2019. Documentation. Verkkoaineisto. <https://docs.unity3d.com/ScriptReference/index.html>. Luettu 12.01.2020.

Unity. 2020a. Game engines—how do they work. Verkkoaineisto. <https://unity3d.com/what-is-a-game-engine>. Luettu 01.02.2020.

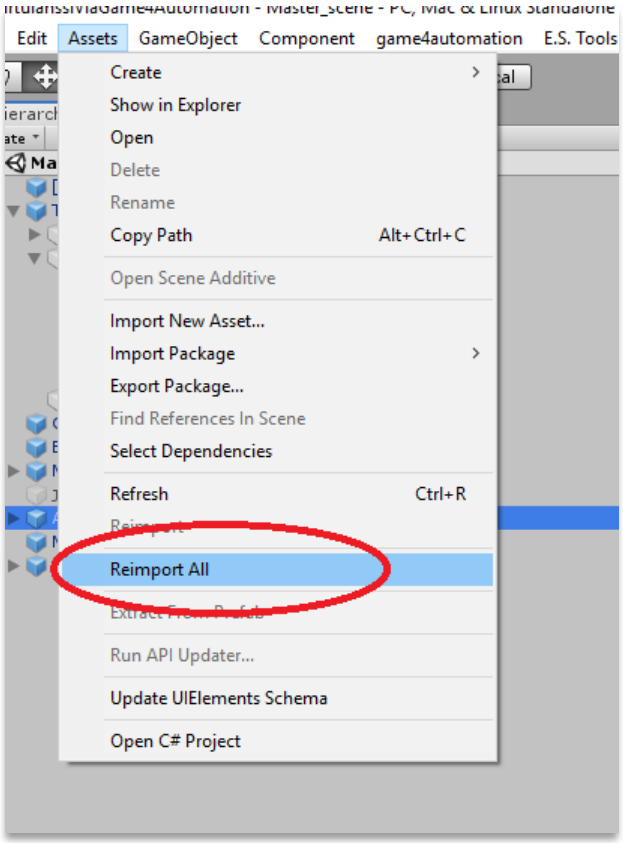
Unity. 2020b. Games made in Unity. Verkkoaineisto. <https://unity3d.com/games-made-with-unity>. Luettu 20.12.2019.

Asennus- ja käyttöohje

Lataa ja asenna Unity Hub, Unity:n omilta verkkosivuilta: https://unity3d.com/get-unity/download

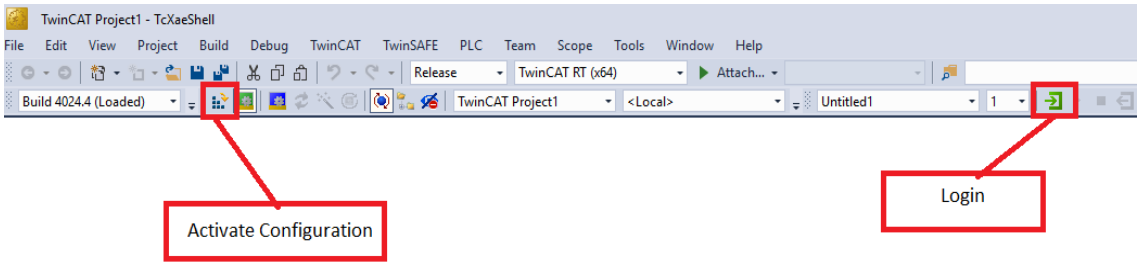
Lataa ja asenna myös TwinCAT 3, Beckhoff:n verkkosivuilta: https://www.beckhoff.com/twincat/

Kun Unity Hub on asennettu, asenna sen avulla Unity versio 2019.2.5f1 tai uudempi. **HUOM! Jos Unity versio on uudempi, jotkin pelin ominaisuudet eivät välttämättä toimi…"Assets → Reimport all" voi korjata. Mahdollisesti täytyy asentaa myös uusi versio Game4Automatio-lisäosasta.**



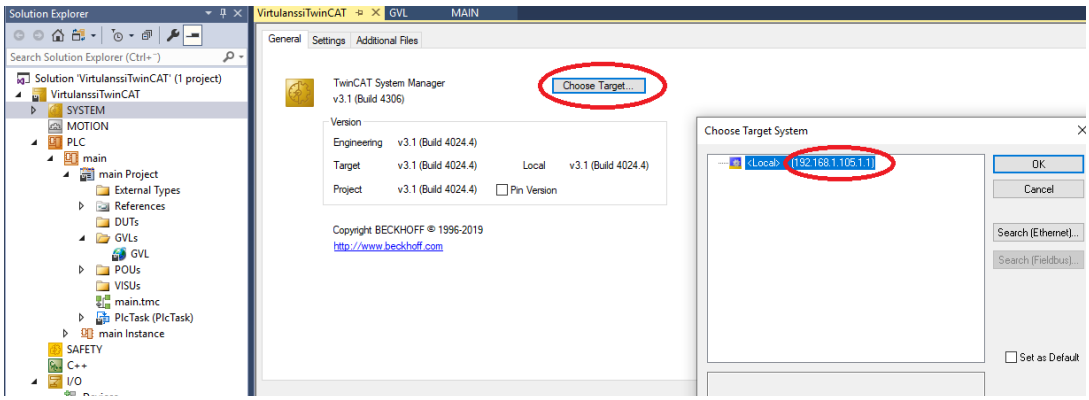
Kuva 1. Reimport all

Unityn asennuksen jälkeen voidaan avata projekti sen kansiosta Projektin kansio → Assets → Scenes → käynnistä MasterScene. Kun projekti on asentunut, voidaan tuoda muutkin skenet projektiin raahaamalla ne hiirellä projekti näkymästä projekti puuhun. Avaa PLC projekti TwinCATissä → käynnistä ”Activate Configuration” → lataa ohjelma PLC-laitteelle ”Login”.



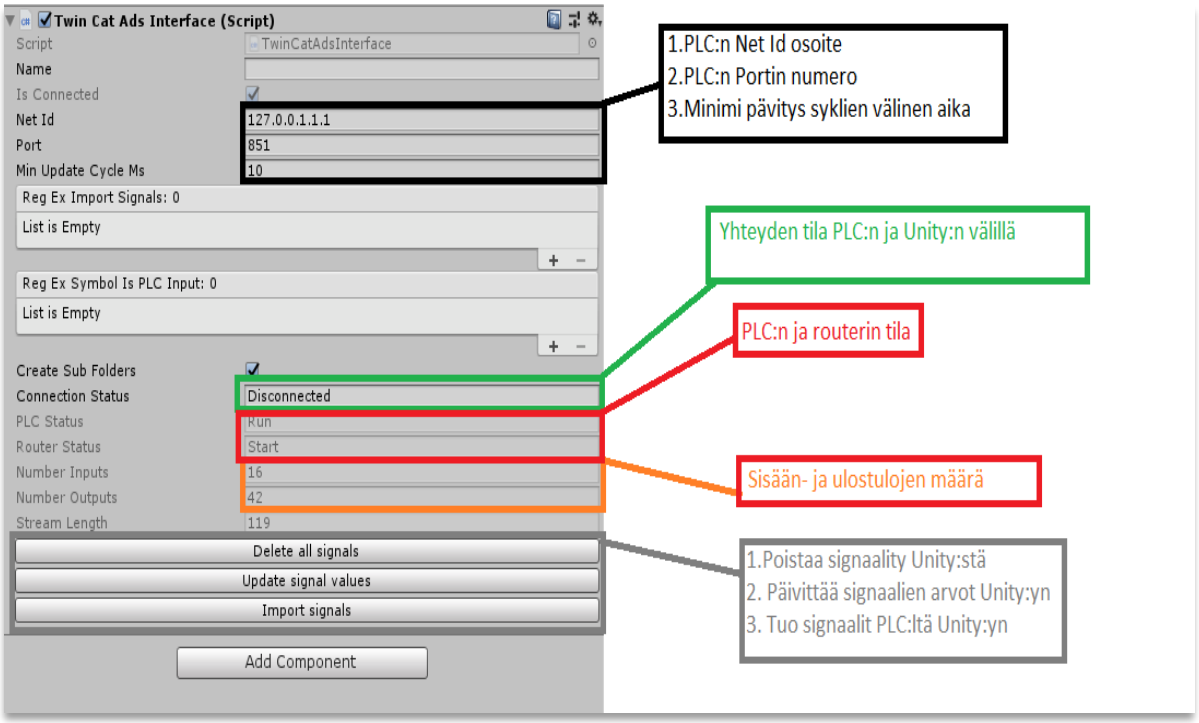
Kuva 2. Painikkeet TwinCATissä

Unity:n puolella avaa ”Inspector”- ikkunaan ”TwinCATInterface”-komponentti → Kirjoita kohtaan ”Net Id” TwinCATin ”Net Id” (Id:n löytää TwinCAT-projektista klikkaamalla projektipuusta ”System” → ”Choose Target”).



Kuva 3. Net Id

Tämän jälkeen syötä ”Port” (”Port” on paikallisessa systeemissä vakiona ”851”). Kun kohdelaitteen tiedot ovat asetettu voidaan tuoda signaalit PLC-projektista klikkaamalla ”Import signals”, jonka jälkeen ne tulevat näkyviin ”TwinCATInterface”-komponentin alle projektipuuhun.



Kuva 4. TwinCAT ADS-käyttöliittymä

Signaalien tuonnin jälkeen voidaan ne yhdistää ambulanssin muuttujiin. Ensin avataan projektipuusta ”Ambulance”-peliobjekti, jolloin sen sisältämät tiedot näkyvät ”Inspector”-ikkunassa. Listalla näkyy skripti nimeltä: ”SignalsAmbulance” → Tuodut signaalit voidaan nyt raahata skriptin kohtiin, joissa lukee ”PLCInputFloat”. Skriptin tiedot: X-, Y-, ja Z_angle → syöttävät ambulanssin Eulerin kulmat, Acceleration → kulkusuunnan kiihtyvyyttä, Speed → kulkusuunnan nopeutta.

Kuva 5. Signaalien raahaus

Jos tarvetta saada lisää informaatiota ulos ambulanssista voi skriptiin lisätä ”PLCInputFloat” tai ”PLCInputInt” muuttujia. Jos haluaa muuttujat, joihin voi raahata signaaleja, täytyy muuttujista tehdä julkisia (Public). Skripti löytyy sekä ”Ambulance”-ikkunasta että ”Asset”-kansiosta.

Nyt kun pelin käynnistää, tulisi sen toimia!