jamk

Hardware-In-the-Loop-simulaation laitteistotutkimus

Tuukka Hänninen

Opinnäytetyö, AMK
Helmikuu 2024
Sähkö- ja automaatiotekniikka (AMK)

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Hänninen, Tuukka

Hardware-In-the-Loop-simulaation laitteistotutkimus

Jyväskylä: Jyväskylän ammattikorkeakoulu. Joulukuu 2023, 42 sivua.

Sähkö- ja automaatiotekniikka, insinööri (AMK), monimuoto. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi Valtra Oy Ab. Työ tehtiin Valtran tuotekehitysosastolla, jossa käytetään simulaatioavusteisia tekniikoita tuotekehityksen ja validoinnin apuna. Työn tavoitteena oli tutkia vaihtoehtoisia, kustannustehokkaampia tapoja suorittaa simulaatiota.

Opinnäytetyön tavoitteena oli tutkia voiko nykyisen simulaation siirtää nykyiseltä suoritusalustalta vaihtoehtoiselle Beckhoff-valmistajan toimittamalle alustalle sekä selvittää millä tavoin simulaatiokehitys eroaa vanhan ja vaihtoehtoisen simulaation suoritusalustan välillä. Tietoperusta luotiin pääasiassa simulointiympäristöjen ja simulointialustojen valmistajien vapaasti saatavasta dokumentaatiosta kehittämistutkimuksena, jolloin työn tuloksia analysoitiin tietoperustan perusteella.

Alkuperäinen simulaatiomalli oli hyvin laaja ja tehtäväksi muodostui aiheen rajauksen jälkeen tutkia simulaation traktoria simuloivan osan simulointiratkaisujen toiminta uudella alustalla. Simulaatiota testattiin osissa uudella alustalla ja löydettyjen ongelmien juurisyyt pyrittiin selvittämään ja ratkaisemaan. Samalla raportoitiin simulaation kehityksestä alustojen välillä ja mahdollisuuksien mukaan alustojen välistä suorituskykyä arvioitiin.

Tuloksina todettiin, että simulaation siirtäminen uudelle alustalle oli mahdollista. Osittain simulaatio jouduttaisiin kuitenkin kehittämään alusta uudelle alustalle alustojen välisistä eroista johtuen. Alustojen välistä suorituskykyä ei aikarajoitteista johtuen ehditty vertailemaan.

Työn johdosta muodostui kattava kuva Beckhoff-alustan ominaisuuksista ja mahdollisuuksista ja työn johdosta Valtran tuotekehityksessä otettiin Beckhoff-toimittajan laitteita käyttöön pienemmissä testijärjestelmissä.

Avainsanat (asiasanat)

Simulaatio, Hardware-in-the-loop, Beckhoff

Muut tiedot (salassa pidettävät liitteet)

-

jamk

Description

Hänninen, Tuukka

Platform research of Hardware-In-the-Loop Simulation hardware

Jyväskylä: JAMK University of Applied Sciences, December 2023, 42 pages.

Degree Programme in Electrical and Automation Engineering. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

The work was carried out in Valtra's product development department, where simulation-assisted techniques are used to support product development and validation. The goal of the project was to investigate alternative, cost-effective ways to perform simulation.

The objective of the thesis was to explore whether the current simulation could be transferred from the current execution platform to an alternative platform provided by Beckhoff, and to determine how simulation development differs between the old and alternative simulation execution platforms. The theoretical basis was mainly created through a development study of freely available documentation from simulation environments and simulation platform manufacturers. The results of the work were analyzed based on this theoretical foundation.

The original simulation model was extensive, and after narrowing down the topic, the task was to examine the operation of simulation solutions for the part simulating the tractor on the new platform. The simulation was tested in parts on the new platform, and the root causes of the identified problems were sought and resolved. Simultaneously, the development of the simulation between platforms was reported, and, if possible, the performance between platforms was evaluated.

The results indicated that transferring the simulation to the new platform was possible. However, due to differences between platforms, the simulation would need to be partially developed for the new platform. Due to time constraints, a comparison of performance between platforms was not possible.

As a result of the work, a comprehensive understanding of the features and possibilities of the Beckhoff platform was formed. As a result of the work, Valtra's product development began using Beckhoff supplier devices in smaller test systems.

Keywords/tags (subjects)

Simulation, Hardware-in-the-loop, Beckhoff

Miscellaneous (Confidential information)

-

Sisältö

Kuviot

1 Johdanto

Nykyaikainen tuotekehitys on nopeampaa kuin koskaan kun kehitysiteraatioiden väliin jäävä aika on puserrettu minimiin ja samaan aikaan tuotteiden ominaisuuksien määrä kasvaa. Tämä aiheuttaa haasteita tuotekehitykselle ja ennen kaikkea tuotteen toimivuuden varmistamiselle. Menestyneen tuotteen edellytyksenä on sen ongelmaton toiminta ja ongelmien löytäminen tuotteen kehitysvaiheessa nopean kehityssyklin myötä on haastavaa.

Simulaation avulla tuotekehitystä voidaan nopeuttaa huomattavasti ja etenkin liikkuvien koneiden kehityksessä käytetään simulaatioavusteisia suunnittelutekniikoita, kuten tässä opinnäytetyössä käsiteltävää Hardware-In-the-Loop-simulaatiota, jäljempänä HIL-simulaatiota. HIL-simulaatiossa koneesta luodaan reaaliaikainen simulaatiomalli, joka kytketään I/O-rajapinnan, eli analogisten ja digitaalisten sisään- ja ulostulokanavien kautta, ohjainlaitteisiin.

Reaaliaikainen simulaatio reagoi ohjainlaitteiden signaaliin riittävällä tarkkuudella kuten aito, fyysinen kokoonpano ja mahdollistaa ohjausjärjestelmän testaamisen. Virtuaalinen simulaatio mahdollistaa automatisoidun sovelluspohjaisen robottitestaamisen normaalin toiminnallisen testaamisen lisäksi.

1.1 Työn tausta ja tavoite

Työ toteutettiin Valtran tuotekehitysosastolla, jossa työssä käytetty simulaatio on aktiivisessa kehityksessä. Valtralla HIL-simulaatiota käytetään traktorin sovelluksen validointiin testaamalla traktorin sähköisen ohjauslaitteen (electronic control unit, ECU) sisäistä ohjelmaa simulaatiota vasten.

Valtran järjestelmää käytetään sekä ohjelmalliseen robottitestaukseen, että toiminnalliseen testaukseen. Simulaatiojärjestelmään on integroitu erillinen ajoyksikkö, DSI (driving simulator interface), sekä Unreal Engine-pelimoottorilla toteutettu graafinen käyttöliittymä. DSI:n ja pelimoottoriympäristön avulla simulaatiota on mahdollista ajaa ja testata kuten sen fyysistä vastinetta.

Nykyinen käytössä oleva simulaatioalusta on Speedgoat-valmistajan laitteisto, jonka IO-rajapintaan ECU-ohjaimet kytkeytyvät. Laitteisto on suurikokoinen ja sen esittely esimerkiksi messuilla on verrattain vaivalloista. Vaihtoehtoista rauta-alustaa tutkimalla tavoiteltiin mahdollisuutta

integroida järjestelmä mahdollisesti kokonaan DSI-laitteistoon. Vaihtoehtoiseksi rauta-alustaksi valikoitui jo ennen opinnäytetyön aloittamista Beckhoff-valmistajan laitteisto, ensisijaisesti sen kompaktin koon sekä huomattavasti alemman kustannuksen vuoksi.

1.2 Aiheen rajaus, tutkimuskysymykset ja työn toteutus

Nykyisen järjestelmän ollessa verrattain kompleksinen ja laaja, rajattiin aihe siten, että testattiin järjestelmän ytimessä olevan traktorin simulaatiomallin toiminta uudella simulaatioalustalla ja samalla tutkittiin, kuinka simulaatiokehitys erosi järjestelmillä, minkälaisia ongelmia erilainen alusta aiheutti ja mistä ne johtuivat. Opinnäytetyössä ei tutkittu simulaation ja simulaatioalustan välisiä rautarajapinnan ratkaisuja.

Opinnäytetyön tutkimuskysymykset olivat:

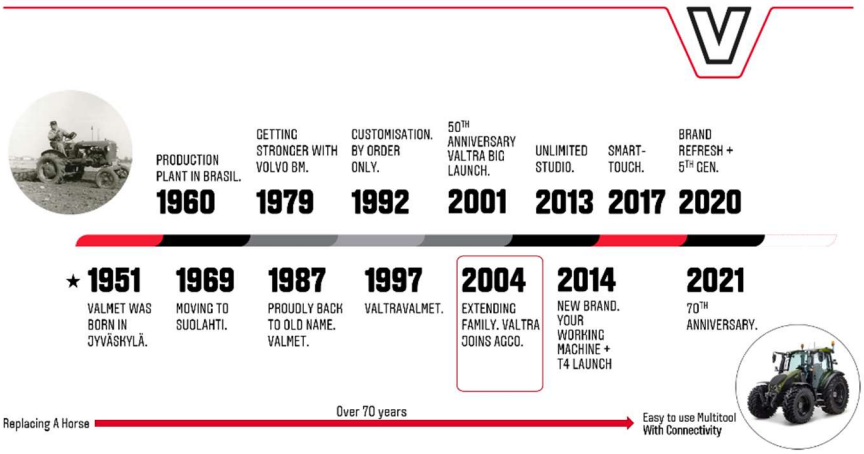
- Onko nykyisen simulaatiomallin siirtäminen uudelle alustalle mahdollista?
- Miten mallin kehitys eroaa alustojen välillä?
- Miten alustojen suorituskyvyt eroavat toisistaan?

Työ toteutettiin kehittämistutkimuksena JAMKin sekä toimeksiantajan eettisiä periaatteita noudattaen kvalitatiivisia, eli laadullisia menetelmiä käyttäen siten, että saavutettaisiin ymmärrys kehitettävästä kohteesta. Koska tutkittava ilmiö on toimeksiantajan sisäisesti tuottama kokonaisuus eikä työssä syvennytty simulaatiototeutuksen yksityiskohtaiseen rakenteeseen, vaan keskityttiin kokeilemaan kokonaisuuden toimintaa eri järjestelmällä, oli tutkimusaineiston pääpaino työssä käytettyjen järjestelmien vapaasti saatavilla olevassa dokumentaatiossa.

1.3 Valtra & AGCO Corporation

Valtra Oy Ab on suomalainen traktorivalmistaja, joka on osa kansainvälistä AGCO-yhtymää. Valtran traktoreita valmistetaan Keski-Suomessa Suolahdessa, Brasiliassa Mogi das Cruzezissa sekä Kiinan Shagnzoussa. Suolahdessa valmistetaan vain traktoreita ja Brasiliassa traktoreiden lisäksi valmistetaan myös muita maatalouskoneita, kuten leikkuupuimureita. Kiinassa valmistetaan pääasiassa pienempiä traktoreita.

Valtran juuret ovat sodanjälkeisessä suomessa valtionyhtiö Valmetissa ja se on perustettu 1945. Ensimmäinen prototyyppi valmistui vuonna 1949 ja traktoreiden tuotanto käynnistyi Jyväskylän Kivääritehtaalla vuonna 1951, josta kuvion 1 aikajana alkaa. 50-luvun loppuun mennessä Valmet maahantoi Etelä-Amerikkaan useita satoja traktoreita vuosittain ja Brasilian hallituksen halusta kansallistaa traktorintuotanto järjestettiin tarjouskilpailu, jonka seurauksena vuonna 1960 Valmet aloitti tuotannon myös Brasiliassa.



Kuvio 1, Valtran historiaa

Vuoteen 1997 asti Valmet oli täysin valtio-omisteinen yhtiö ja erinäisten järjestelyiden ja yritysostojen kautta tuotenimi muuttui Valmetista Valtraksi vuonna 2001. Vuonna 2002 Kone Oyj osti Valmetin silloisen omistajan Partek Oy:n osakkeet ja edelleen vuonna 2004 myi Valtran traktoritoiminnot ja Partekiin kuuluneen SisuDieselin moottoritoiminnot yhdysvaltalaiselle AGCO-yhtymälle 600 miljoonalla eurolla.

AGCO-yhtymä on 1990 perustettu yhdysvaltalainen pörssiyhtiö, joka on maailman suurimpia traktoreiden ja maatalouskoneiden valmistajia. Suomessa Agco omistaa Valtran lisäksi Nokialla Linnavuoressa sijaitsevan AGCO Power Oy:n, jonka juuret ovat myös valtionyhtiö Valmetissa ja se valmistaa dieselmoottoreita. Kuviossa 2 esitellään muut AGCO-yhtymään kuuluvat brändit.

Kuvio 2, AGCO-yhtymän omistamat yritykset

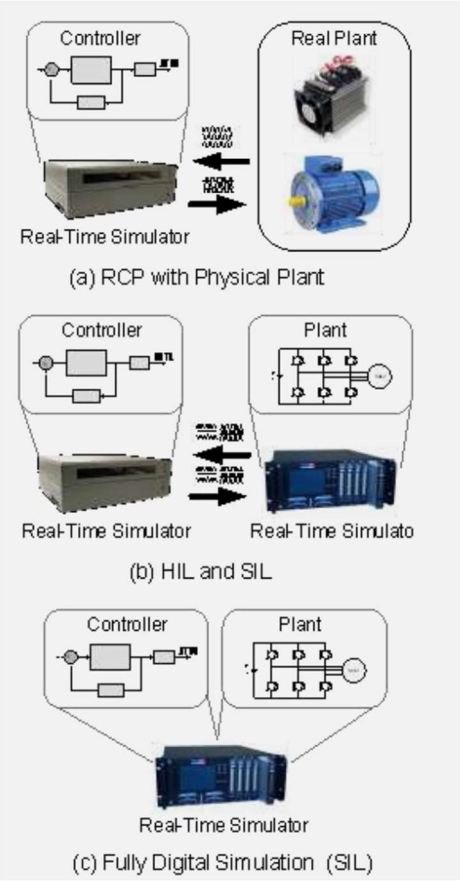
2 Mitä on reaaliaikasimulaatio

Reaaliaikasimulaatio on verrattain nuori simuloinnin muoto, jonka kehityksen on mahdollistanut ennen kaikkea prosessoreiden laskentatehon lisääntyminen. Verrattuna tavanomaiseen simulaatioon, jossa luodaan matemaattinen malli simuloitavasta ilmiöstä, suoritetaan mallin laskenta ja tarkastellaan tuloksia laskennan valmistuttua, reaaliaikasimulaatiossa mallin laskennan tuloksia voidaan tarkastella jokaisella diskreetillä aika-askeleella, askelten ollessa jopa alle millisekunnin mittaisia. Toinen mainittava ero normaaliin simulaatioon verrattuna on siinä, että simulaation muuttujiin voidaan vaikuttaa laskennan aikana. (Bélanger, Venne & Paquin n.d.)

Reaaliaikasimulaation on luettava kaikki mallin sisään tulevat muuttujat, ratkaistava ilmiötä kuvaavat funktiot ja tallennettava mallin ulostulomuuttujat yhden annetun aikaikkunan aikana, muutoin simulaation suoritus on virheellinen. Tämä mahdollistaa ilmiön mallintamisen tavalla, joka vastaa sen fyysisen vastineen käyttäytymistä. (Bélanger, Venne & Paquin n.d.)

Reaaliaikasimulaation käyttö voidaan yleisesti jakaa kolmeen eri kategoriaan kuvion 3 mukaisesti:

1. Simuloitu ohjausjärjestelmä yhdistetään fyysisen laitteiston (eng. plant) kanssa, jolloin puhutaan rapid control prototyping (RCP) tekniikasta.
2. Simuloitu plant-malli yhdistetään fyysiseen ohjausjärjestelmään, jolloin puhutaan hardware-in-the-loop (HIL) simulaatiosta
3. Sekä ohjausjärjestelmä, että plant-malli on simuloitu ja yhdistetty, jolloin puhutaan software-in-the-loop (SIL) simulaatiosta. (Bélanger, Venne & Paquin n.d.)

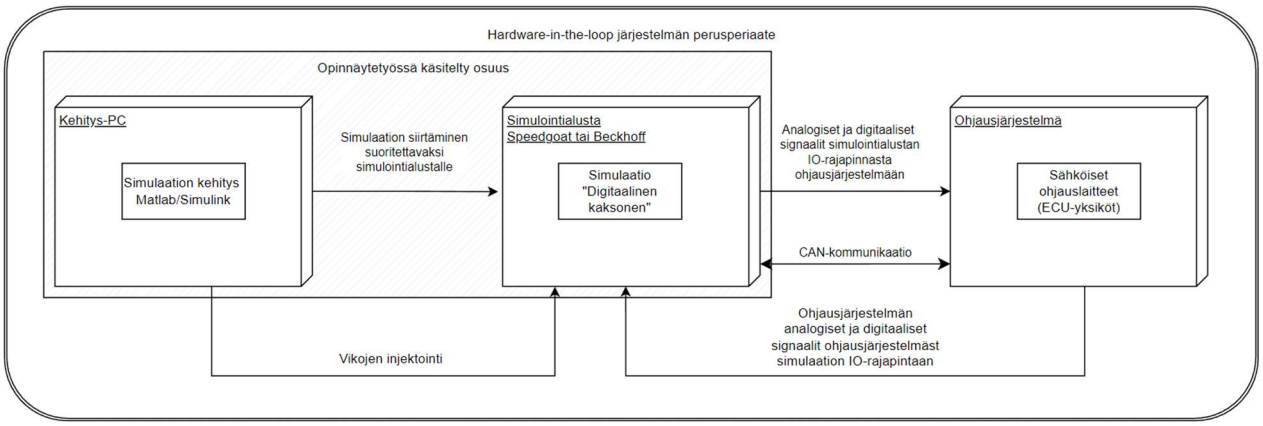


Kuvio 3, Reaaliaikasimulaation variantit

(Bélanger, Venne & Paquin n.d.)

Tässä työssä käsiteltiin HIL-simulaatiota, jota käytetään yleisesti mm. autoteollisuuden tuotekehityksessä, jossa sen merkitys on kasvanut samalla kun liikkuvista koneista, kuten autoista tai tässä tapauksessa traktoreista, on tullut yhä monimutkaisempia. HIL-simulaatio on tehokas ja nopea tapa ECU-ohjelmiston sovelluksen testaamiseen ja validointiin verrattuna testaamiseen, joka tehtäisiin valmiin koneen avulla. (Burnham, Copp, Mouzakitis & Parker 2009)

Hardware-in-the-loop simulaation eduiksi nähdään sen nopeus ja laajat mahdollisuudet testata ohjausjärjestelmää, kun ohjattava laite ja sen kaikki mitta- ja toimilaitteet ovat kontrolloitavissa simulaatiossa. Tällöin voidaan testata erilaisia anturi- ja toimilaitevikoja ja samalla esimerkiksi altistaa ohjausjärjestelmän komponentit kontrolloidusti äärimmäisille olosuhteille (Burnham, Copp, Mouzakitis & Parker 2009). Kuviossa 4 on nähtävillä järjestelmän perusperiaate ja tarkemmin tässä työssä käsitelty osuus siitä.



Kuvio 4, Hardware-in-the-loop järjestelmän perusperiaate ja opinnäytetyössä käsitelty osuus

3 Simulointiympäristöt ja alustat

Tässä työssä käsitellään Mathworks-ohjelmistotuottajan simulointiympäristöä pintapuolisesti sekä esitellään nykyisin Valtran järjestelmässä käytössä olevaa Speedgoat-alustaa. Syvemmin työssä käsitellään Beckhoff-valmistajan alustaratkaisua, jolle simulaation ydinkomponentteja pyrittiin siirtämään.

Seuraavat kappaleet käsittävät edellä mainittujen valmistajien esittelyt. Näiden jälkeen käydään läpi työn edellytyksenä olevien ympäristöjen asennus, konfigurointi ja testaus.

3.1 Mathwork

Mathworks on maailman johtava matemaattiseen laskentaan keskittyvä yksityinen ohjelmistotoimittaja, jonka tuotteita käytetään laajasti tutkimuksessa, kehityksessä ja opetuksessa. Mathworksin tuotteilla on yli 2 miljoonaa käyttäjää yli 180 eri maassa ja Mathworks tukee laajasti akateemista tutkimusta ja opetusta sen tuotteiden ollessa osana opetusta yli 5 tuhannessa yliopistossa.

Mainittavia tutkimus- ja teollisuusaloja ovat esimerkiksi avaruus-, kommunikaatio-, elektroniikka-, energia- tai sotilasteknologia. (Simulink Fundamentals 2022)

Mathworksin tärkeimmät tuotteet ovat Matlab ja Simulink, joista Matlab on täsmällisemmin korkean tason ohjelmointikieli, jota käytetään esimerkiksi data-analytiikassa, numeerisessa laskennassa tai algoritmikehityksessä. Simulink on Matlabin päälle kehitetty graafinen, erilaisiin toiminnallisiin lohkoihin ja vuokaavioon perustuva lisäosa. (Simulink Fundamentals 2022)

3.2 Simulink

Simulink on ympäristö, joka on kehitetty dynaamisten systeemien ja ilmiöiden mallintamiseen ja simulointiin. Lohkokaavioon perustuva ohjelmointi mahdollistaa nopean ja joustavan tavan luoda virtuaalisia prototyyppejä, joiden avulla kehitettävää järjestelmää voidaan tutkia ja testata laaja-alaisesti ja yksityiskohtaisesti. Simulink on kiinteä osa mallipohjaista suunnittelua, joka on verrattain uusi tapa ajatella suunnitteluprosessia. Mallipohjaisen suunnittelun perimmäisenä ajatuksena on mallin luominen kehitettävästä kohteesta ja tämän mallin jatkuva kehittäminen sen koko elinkaaren ajan. (Simulink Fundamentals 2022)

Simulinkistä löytyy valmiiksi kattava kirjasto erilaisia lohkoja, joiden avulla voidaan nopeasti luoda malli järjestelmästä, jonka fyysiseen kehitykseen menisi huomattavasti pidempi aika laboratorio-olosuhteissa. Asennettavien lisäosien avulla ympäristöön saadaan myös kolmannen osapuolen lohkoja, joiden avulla Simulinkissä toteutettu malli voidaan yhdistää esimerkiksi Speedgoat-valmistajan laiterajapintaan ja ohjata siten erilaisia IO-kanavia, kuten analogisia jännite- ja virtakanavia tai digitaalisia sisään- tai ulostulokanavia. (Simulink Fundamentals 2022)

3.3 Speedgoat

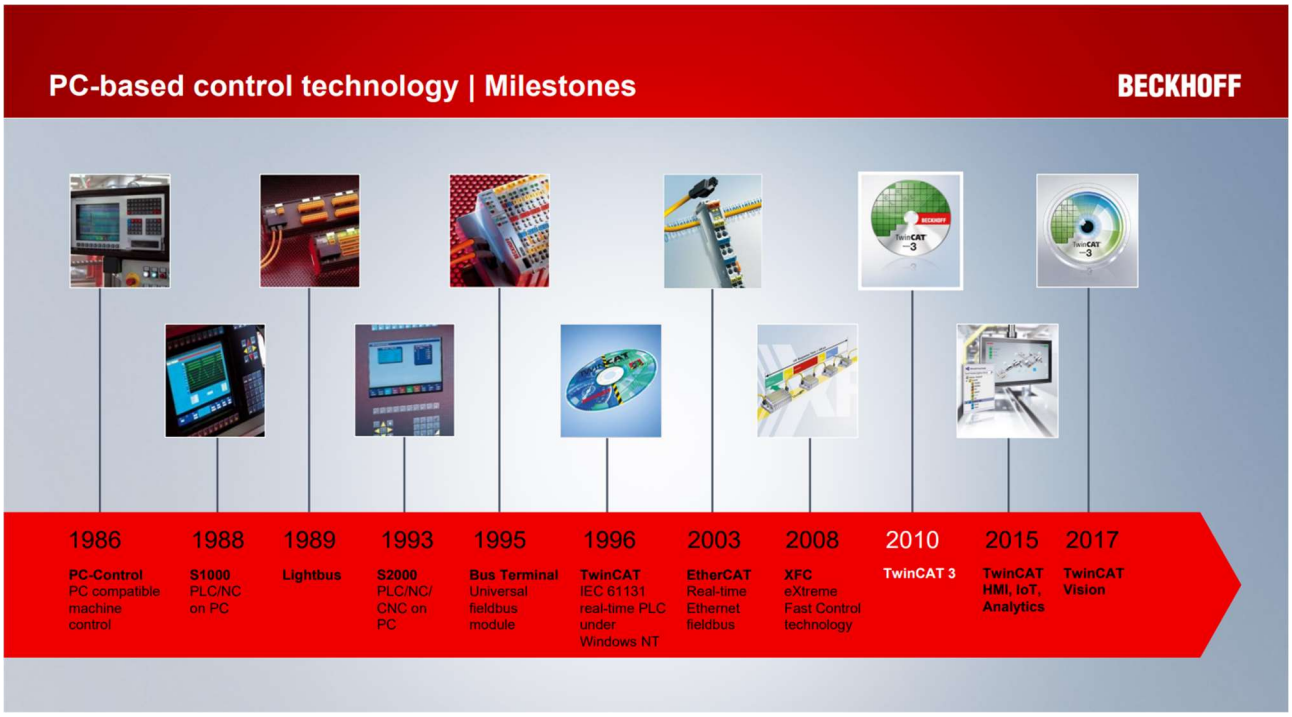
Speedgoat on sveitsiläinen reaaliaikatietokoneiden kokonaistoimittaja, jonka järjestelmissä käytetään Mathworksin reaaliaikaista käyttöjärjestelmää. Speedgoat perustettiin 2007 entisen Mathworksin työntekijän toimesta ja yhteistyö näiden kahden toimijan välillä on tiivistä. (About Speedgoat n.d.)

Speedgoat tekee yhteistyötä myös muiden simulaatiotoimijoiden kanssa ja järjestelmiä voidaan suoraan käyttää esimerkiksi CarSim dynamiikkasimulaation tai VI-Grade reaaliaikasimulaation kanssa. Vielä laajempi yhteensopivuus eri simulaatiojärjestelmien välillä saavutetaan Matlab lisäosien ja integraatioiden kanssa. (Third-Party Connections n.d.)

3.4 Beckhoff & Twincat

Beckhoff on saksalainen 1950-luvulla alkunsa saanut perheomisteinen sähköalan konserni, jonka ydintoimintaa on teollisuusautomaation valmistus ja kehitys. Alun perin Beckhoffin ydintoimintaa oli sähkötavaran vähittäismyynti ja sähköasennuspalvelut, kunnes 1980-luvun alussa perustajaperheen jälkeläinen ja konsernin nykyinen toimitusjohtaja Hans Beckhoff alkoi tutkimaan PC-pohjaista ohjauslogiikkaa paikallisten teollisuusyritysten tarpeeseen. (Beckhoff konserni n.d.)

Vuonna 1986 Beckhoff julkaisi ensimmäisen PC-pohjaisen ohjauslogiikan nimeltä PC-Control, joka loi pohjan Beckhoffin nykyiselle kuvion 5 mukaiselle tuoteportfoliolle. Vuonna 1989 Beckhoff julkaisi Lightbus-nimisen ohjausväylän, jonka avulla järjestelmä saatiin hajautettua laajemmalle alueelle. Vuonna 1996 julkaistiin ensimmäinen versio Twincat-nimisestä ohjelmointiympäristöstä, joka on edelleen Beckhoff-tuoteportfolion ytimessä, nykyisen version ollessa Twincat 3. Vuonna 2003 julkaistiin EtherCAT-väylätyyppi, jonka siirtyi käyttämään ethernet-protokollaa ja on laajasti käytetty väylätyyppi myös Beckhoff-laitteiden ulkopuolella. (PC-based Control 2019)

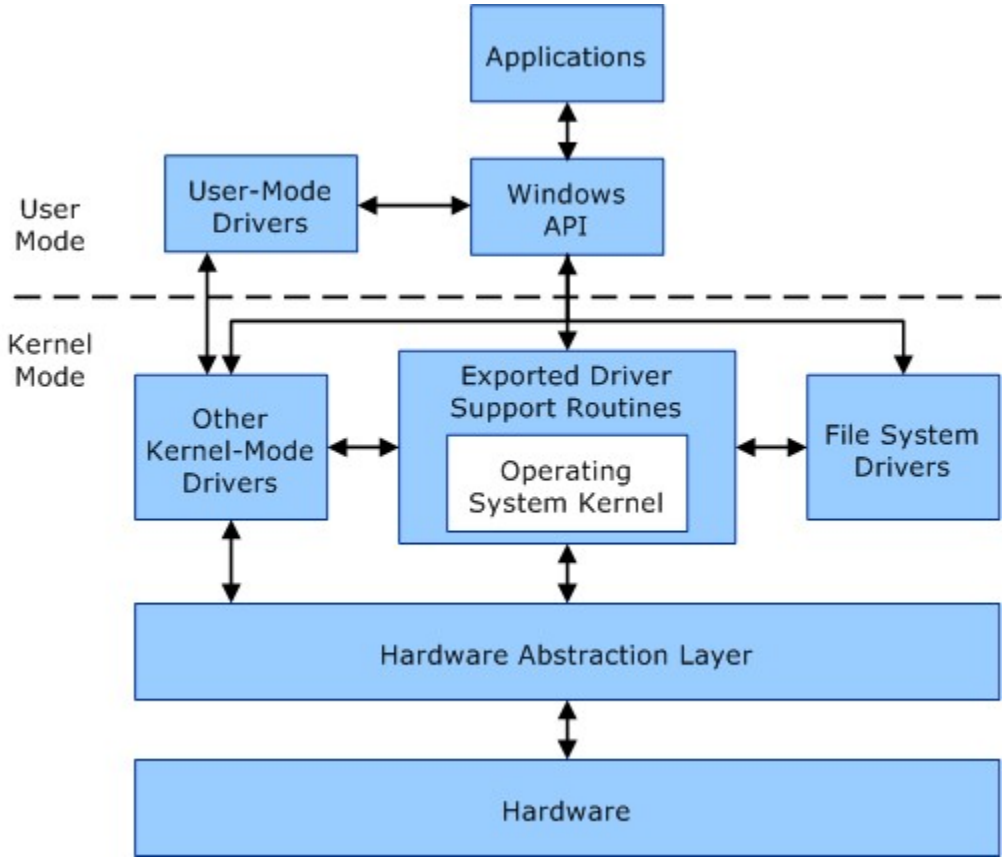


Kuvio 5, Beckhoff tuotehistoriaa (PC-based Control 2019)

4 Ympäristöjen asennus ja konfigurointi

Työtä varten asennettiin tarvittavat sovellusympäristöt ja niille tehtiin vaadittavat perusmääritte-lyt ja testi, jotta varmistuttiin ympäristöjen toiminnasta. Ympäristöjen asennus oli verrattain suora-viivainen prosessi ja asennusvaiheessa tärkein huomioon otettava asia oli asennuksen aloittami-nen Visual Studio 2019-ympäristöstä, johon myöhemmin asennettu TwinCAT 3.1 integroitui. Kehityskoneen ja Beckhoff-kohdekoneen käyttöjärjestelmä oli Windows 10.

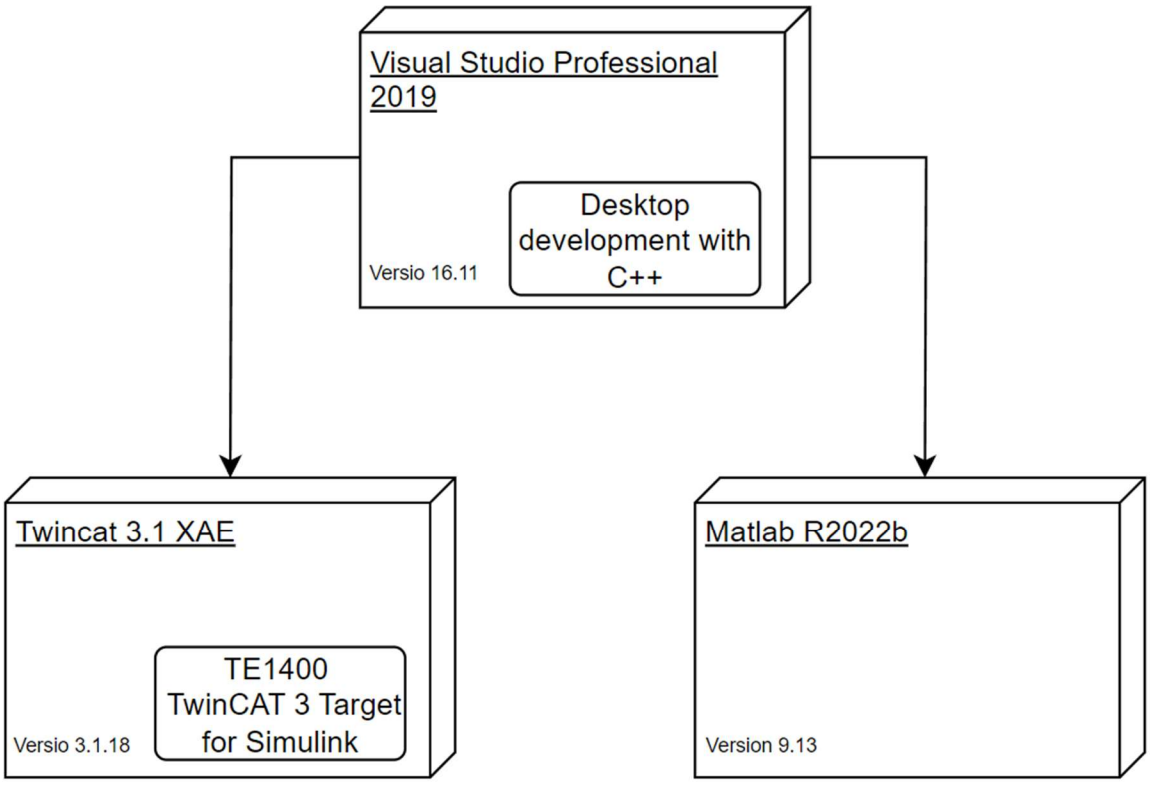
Asennusten lisäksi piti kehityskoneella ja Beckhoff-kohdekoneella (myöh. IPC, industrial personal computer) tehdä käyttöjärjestelmän turvallisuusvaatimuksiin liittyen toimenpiteitä, jotka johtuivat koodin suorittamisesta normaalia syvemmällä järjestelmätasolla. Windows 10-käyttöjärjestel-mässä oli kaksi tasoa kuvion 6 mukaisesti, joilla ohjelmaa suoritettiin: käyttäjätaso ja kerneltaso (User mode and kernel mode n.d.).



Kuvio 6, Windows 10 järjestelmän tasot (User mode and kernel mode n.d.)

IPC:llä TwinCAT-ympäristöstä yksittäiselle prosessoriytimelle allokoitu koodi suoritettiin kernel-tasolla, jonka vuoksi koodi täytyi allekirjoittaa Microsoftin vaatimusten mukaisesti ennen sen suorittamista (Kernel-mode code signing requirements n.d.). Tätä varten piti TwinCAT-ympäristössä luoda sertifikaatti, jolla suoritettava koodi allekirjoitetaan. Tämän lisäksi piti IPC asettaa testitilaan, koska koodin virallinen suorittaminen vaati koodin allekirjoittamisen virallisen valmistajan julkaisemalla sertifikaatilla, mutta kehitysvaiheessa voidaan edetä testitilassa ja kehittäjän generoimalla sertifikaatilla (Enable loading of test signed drivers, N.d.).

Asennusjärjestys ja sovellusympäristöjen versiot olivat kuvion 7 mukaisia. Sovellusympäristöjen lisäksi asennettiin tarvittavat lisäosat.



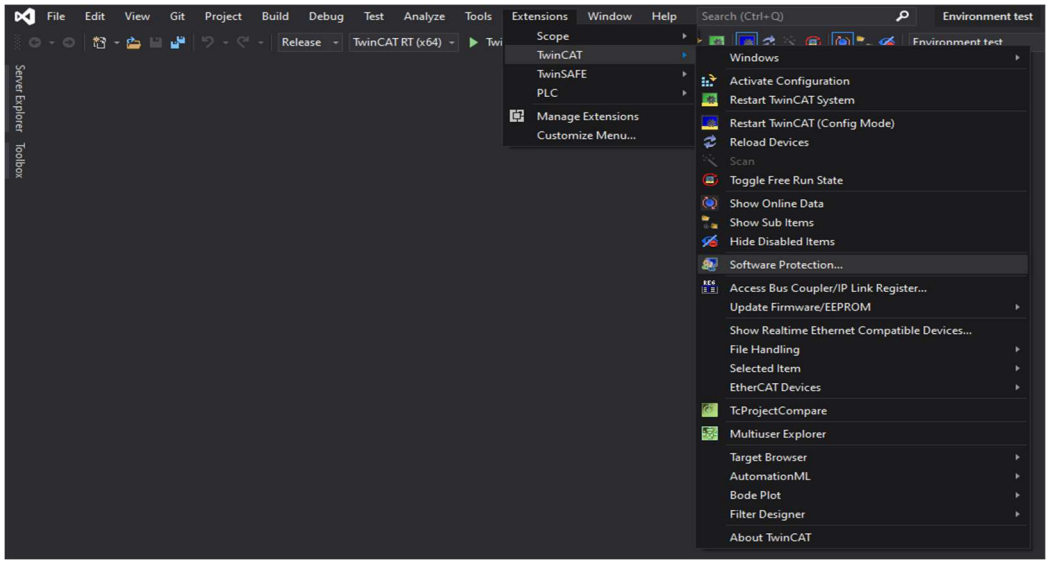
Kuvio 7, Asennusjärjestys, sovellusversiot ja asennettavat lisäosat

Visual Studio Professionalin ja Matlabin käyttö vaati lisenssin ja tässä työssä käytin Valtralle myönnettyjä lisenssejä. TwinCAT 3.1 XAE sovellus oli vapaasti ladattavissa Beckhoffin verkkosivuilta.

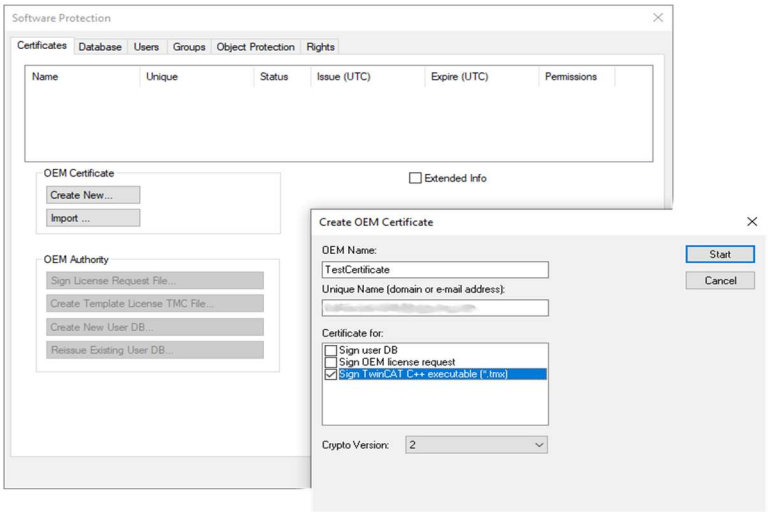
TE1400-laajennus mahdollisti Simulink-mallin suorittamisen TwinCAT-ympäristössä ja se on lisenssivapaa, jos Simulink-mallissa on enintään 100 lohkoa ja 5 sisään tulevaa signaalia ja 5 ulostulevaa signaalia. Tässä työssä vaadittiin lisenssi TE1400-laajennuksen käyttöön ja se saatiin ilmaiseksi työn ajaksi käyttöön Beckhoffilta.

4.1 Sertifikaatin luonti ja IPC:n testitila

Vaadittava sertifikaatti sovelluksen suorittamista varten IPC:llä luotiin TwinCAT-ympäristössä (kuvio 8 ja 9) ja se otettiin käyttöön järjestelmässä. Käyttöönottamiseen on 4 eri tapaa, joista tässä työssä esiteltiin yksi (Setting up driver signing n.d.).

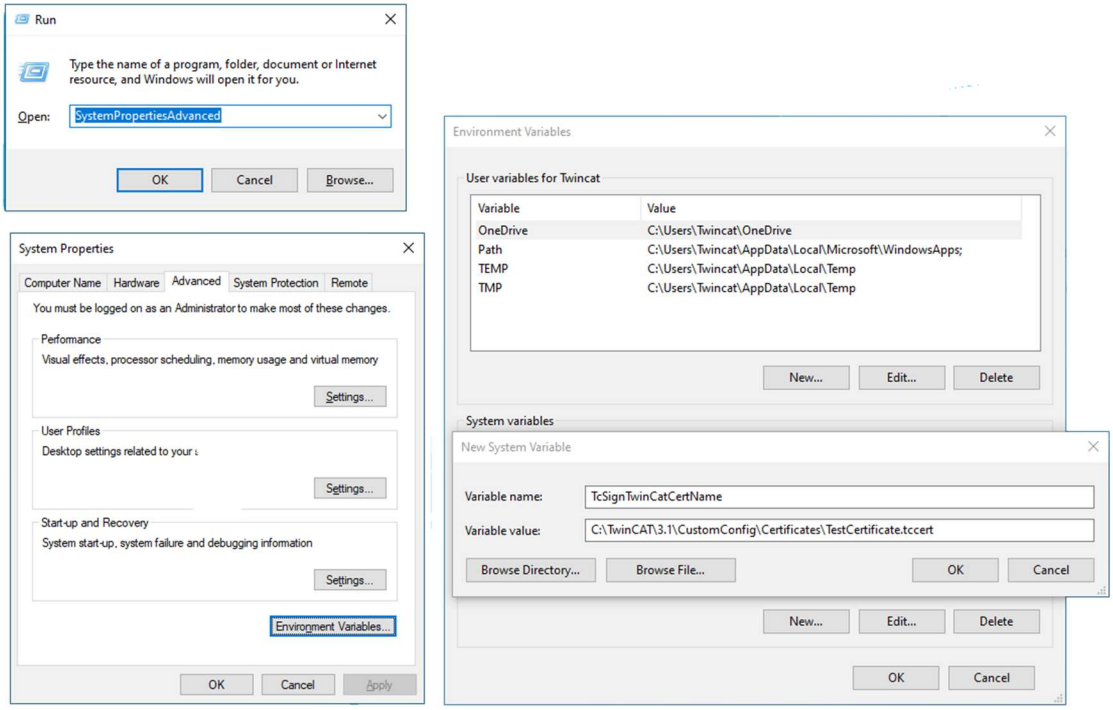


Kuvio 8, TwinCAT ympäristön ylävalikosta valitaan Extensions - TwinCAT - Software Protection (Setting up driver signing n.d.).



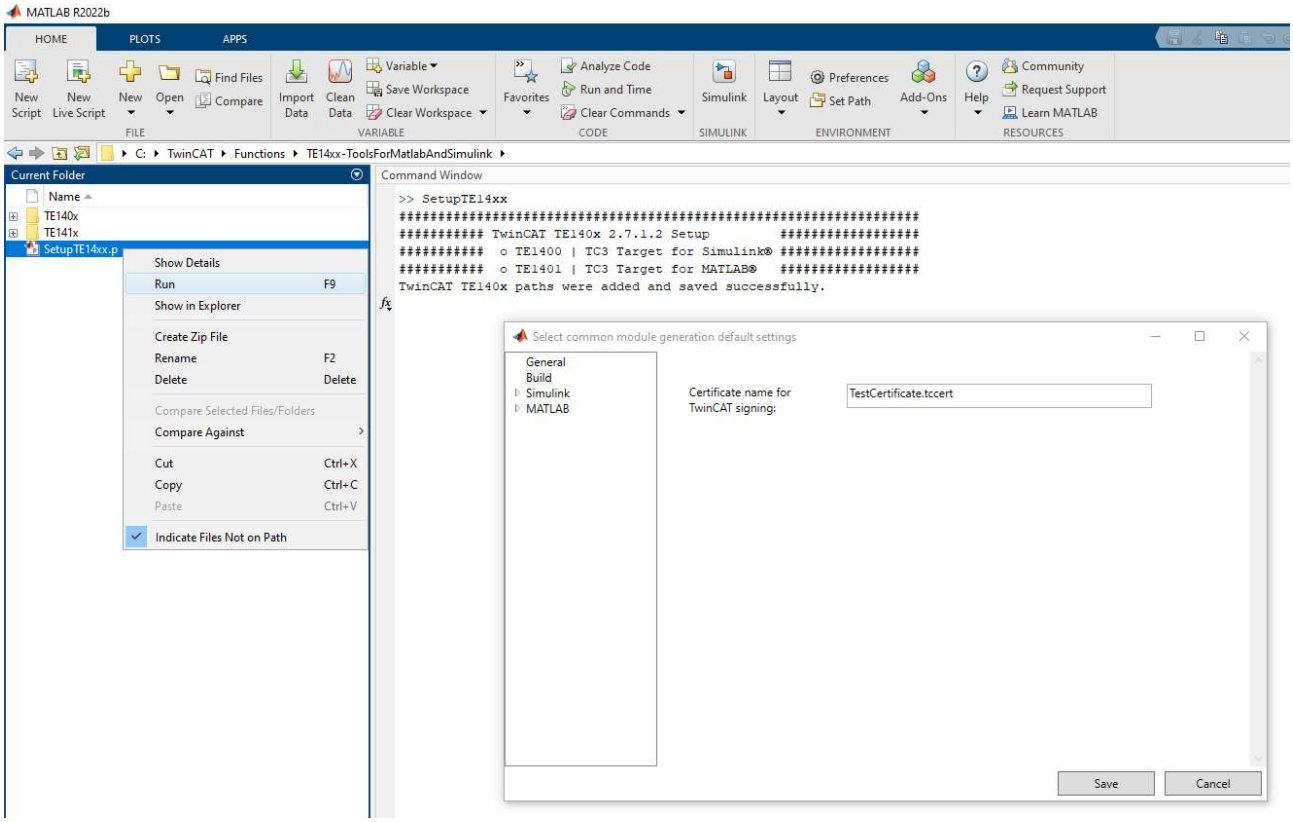
Kuvio 9, Valittiin Sign TwinCAT C++ executable (*.tmx) ja valittiin Crypto Version 2 (Setting up driver signing n.d.).

Sertifikaatille annettiin salasana ja se tallennettiin tietokoneelle oletuspolkuun, jonka jälkeen se lisättiin Windows-järjestelmän polkumuuttujaan. Polkumuuttuja luotiin, jotta sertifikaatti oli järjestelmän laajuisesti käytössä ja siten Matlabin löydettävissä. Painamalla näppäimistöltä näppäimiä win+r ja syöttämällä SystemPropertiesAdvanced aukesi järjestelmän asetusvalikko, jonne polkumäärittely tehtiin kuvion 10 mukaisesti (Setting up driver signing n.d.).



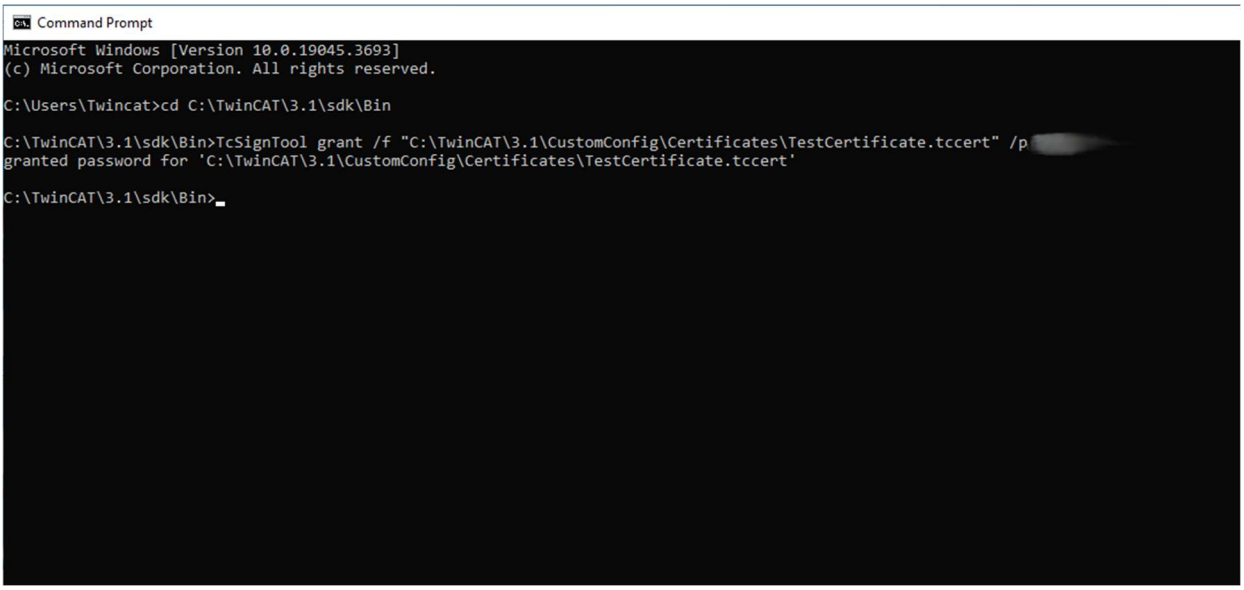
Kuvio 10, Luodun sertifikaatin polkumäärittely (Setting up driver signing n.d.)

Seuraavaksi sertifikaatin nimi osoitettiin TE1400-sovellukselle määrittämällä Matlab-ympäristössä TE1400-sovelluksen asetukset. Avattiin Matlab 2022b sovellus, navigoitiin kansioon C:\TwinCAT\Functions\TE14xx-ToolsForMatlabAndSimulink ja suoritettiin SetupTE14xx.p tiedosto. Tämä avasi asetusikkunan, josta valittiin Build-valikko. Tänne lisättiin aikaisemmin luodun sertifikaatin nimi (kuvio 11).



Kuvio 11, Sertifikaatin lisääminen TE1400-sovellukselle Matlab ympäristöstä

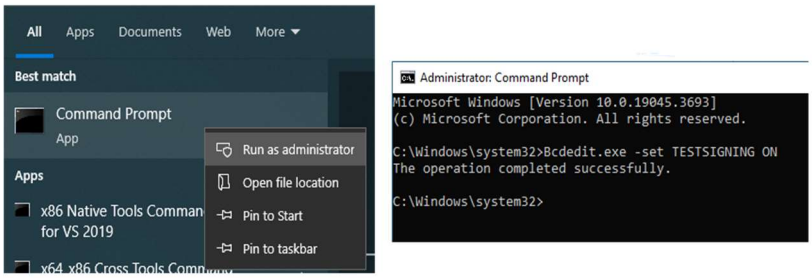
Sertifikaatin luonnin yhteydessä sille määritelty salasana täytyi vielä tallentaa Windows-rekisteriin käyttämällä Beckhoffin komentorivityökalua. Tämä tehtiin turvallisuussyistä, jotta vältyttiin syöttämästä sertifikaatin salasanaa suojaamattomana Simulink-malliin (Setting up driver signing n.d.). Avattiin Windowsin komentolinjatyökalu ja navigoitiin kansioon, jossa työkalu sijaitsi, syöttämällä komento cd C:\TwinCAT\3.1\sdk\Bin. Tämän jälkeen syötettiin komento, kuten kuviossa 12: “TcSignTool grant /f "C:\TwinCAT\3.1\CustomConfig\Certificates\TestCertificate.tccert" /p salasana”.



Kuvio 12, Sertifikaatin salasanan tallentaminen Windows-rekisteriin (Setting up driver signing n.d)

Viimeiseksi täytyi Beckhoff-kohdekone (IPC) asettaa testitilaan, jotta puutteellisella sertifikaatilla allekirjoitettua koodia voitiin suorittaa IPC:llä (Enable Loading of Test Signed Drivers n.d.). Testitila muodosti tietoturvariskin ja IPC:n pääsy julkiseen verkkoon estettiin eristämällä se fyysisesti verkosta.

Avattiin IPC:n komentorivityökalu pääkäyttäjäoikeuksin ja syötettiin komento Bcdedit.exe -set TESTSIGNING ON (kuvio 13) ja käynnistettiin IPC uudestaan. Käynnistyksen jälkeen työpöydän oikeassa alakulmassa oli ilmoitus testitilan käytöstä.



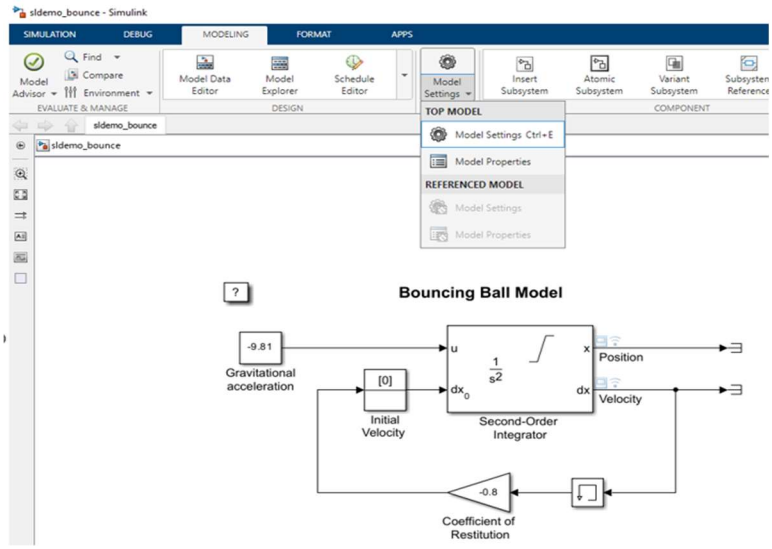
Kuvio 13, Testitilan aktivointi IPC:llä

4.2 Ympäristöjen testaus

Asennettujen ympäristöjen ja tehtyjen määrittelyjen testaus suoritettiin generoimalla valmiista Simulink-esimerkistä koodi TwinCAT-ympäristölle, jossa se käännettiin suoritettavaksi sovellukseksi ja suoritettiin IPC:llä. Jotta Simulink-mallista voitiin generoida koodi TwinCAT-ympäristön käännettäväksi, täytyi Simulink-mallin asetukset muuttaa sopiviksi Beckhoff-alustaa ajatellen.

Simulink-ympäristössä oli laaja valikoima erilaisia asetuksia, jotka vaikuttivat koodin generoimiseen ja ohjelman suoritukseen. Simulink-ympäristöstä pystyi luomaan myös valmiita sertifikaatilla allekirjoitettuja sovelluksia IPC:lle, mutta kehitysvaiheessa nähtiin järkevämmäksi aluksi vain generoida koodi TwinCAT-ympäristölle, jossa se käännettiin. TwinCAT-ympäristössä suoritettavalle koodille voitiin tehdä analyysiä ja diagnostiikkaa, mikäli sovelluksen suoritus keskeytyi virheeseen IPC:llä, mutta tämä vaati sen, että generoitu koodi oli käännetty suoritettavaksi ohjelmaksi TwinCAT-ympäristössä.

Avattiin Simulink-esimerkki, jolle tehtiin oleellisimmat asetusmuutokset. Esimerkki avattiin syöttämällä Matlab-ympäristön komentoriville seuraava komento: openExample('simulink_general/sldemo_bounceExample'). Tämän jälkeen avattiin kuvion 14 mukaisesti Simulink-mallin asetusvalikko Modeling-välilehdeltä valitsemalla Model Settings.



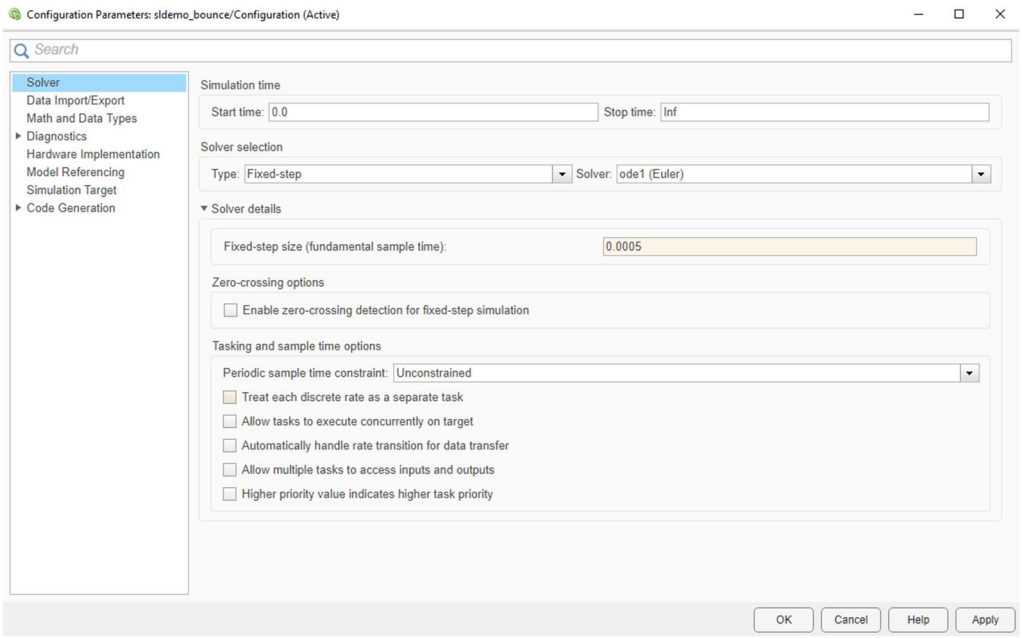
Kuvio 14, Simulink-mallin asetusvalikon avaus

4.2.1 Solver- ja Code Generation-asetukset sekä testikoodin generointi

Solver asetukset määrittivät millä tavalla simulink-mallia suoritettiin ja oleellisia asetuksia tällä sivulla ovat solver selection ja sen jälkeen fixed-step size. Solver määritti millä tavalla simulaation yhtälöt ratkaistiin ja minkä pituisella aika-askeleella laskentaa suoritettiin. Kun kyseessä oli reaaliaikasimulaatio, tuli aika-askeleen olla vakio ja siten valittiin solver selection-tyypiksi fixed-step ja ode1.

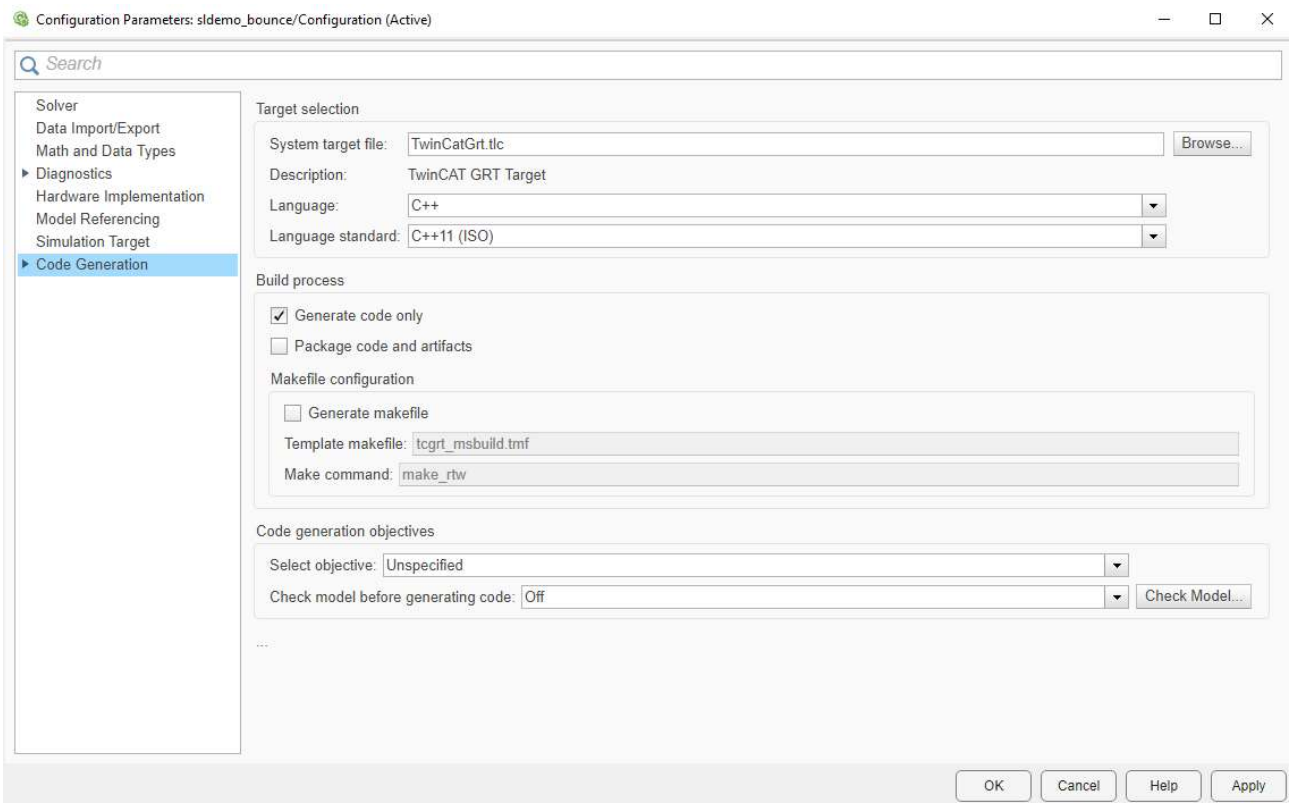
Fixed-step solver valintoja oli useita, mutta koska työssä testattavassa mallissa oli käytössä ode1, valittiin se myös testimallin asetuksiin. Tämä valinta vaikutti siihen millä matemaattisilla tekniikoilla simulaation yhtälöitä ratkaistiin. Ode1 on laskennallisesti kevyin ja samalla myös epätarkin. (Choose a Solver n.d.)

Solver details kohtaan fixed-step size valittiin 0.0005 sekuntia, sillä se oli myös työssä käsitellyssä mallissa käytössä. Fixed-step size määritti aika-askeleen pituuden, jonka aikana yksi simulaation sykli suoritetaan. Asettamalla samat asetukset kuin työssä käsitellyssä mallissa varmistuttiin niiden toiminnasta tässä vaiheessa (kuvio 15).



Kuvio 15, Testimallin Solver-asetusten määrittäminen vastaamaan työssä käsitellyn mallin asetuksia

Code generation-asetukset määrittivät tarkemmin millä tavalla koodin generointi suoritettiin ja minkälaisia rakenteita tai ratkaisuja generoidussa koodissa oli (Code generation configuration n.d.). Oleellisin asetus tällä sivulla oli Target selection kohdan valinta system target file. Tämän valinnan tuli vastata sitä kohdetta, jolle koodia generoitiin. Tähän valittiin TwinCatGrt.tlc tiedosto ja se sisälsi erilaisia Beckhoffin valmiiksi määrittämiä parametreja ja muuttujamäärittelyjä (kuvio 16).



Kuvio 16, Testimallin code generation-asetukset

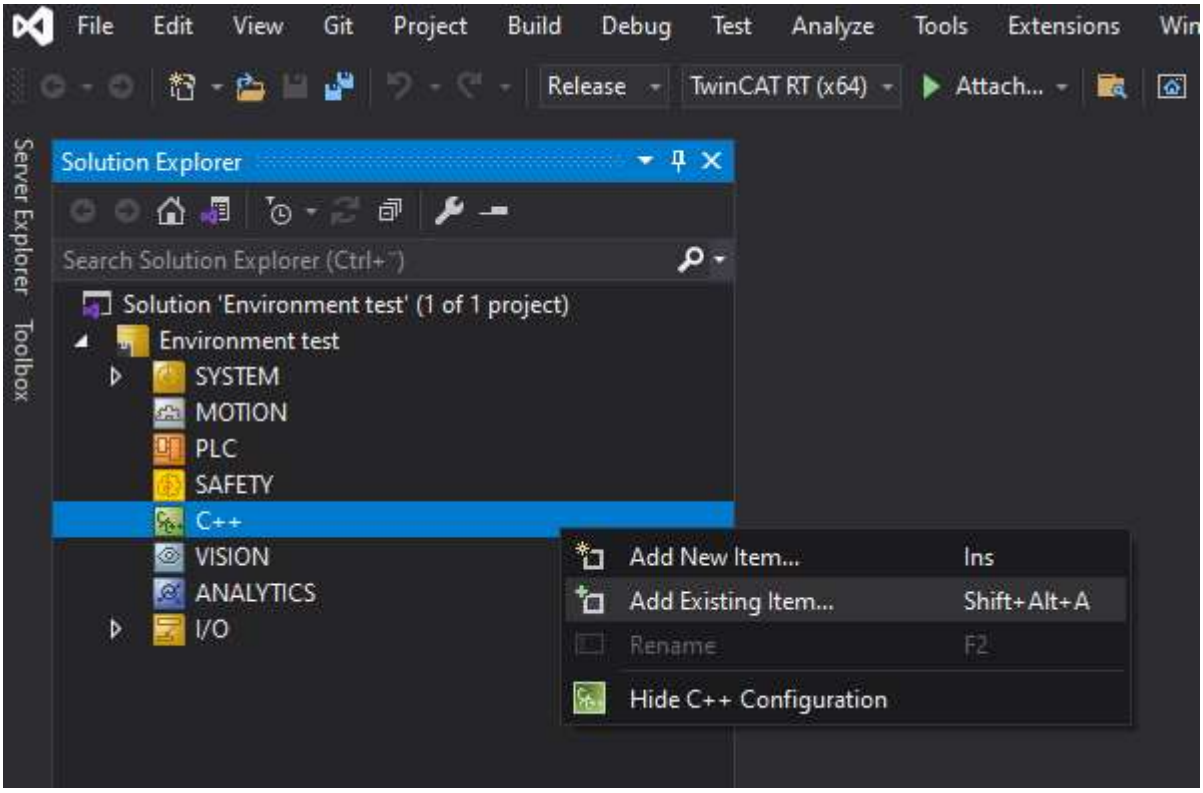
Edellisen valinnan jälkeen code generation-valikon sisältö muuttui vastaamaan valittua kohdetiedostoa ja valikossa siirryttiin TC Build kohtaan. Täältä poistettiin ensimmäinen valinta run the publish step after project generation. Tämä tehtiin siksi, että tavoitteena oli vain generoida koodi Simulink-ympäristöstä, kun valinnan ollessa aktiivinen Simulink-ympäristö pyrkii myös kääntämään generoidun koodin ja julkaisemaan valmiin sovelluksen. Kehitysvaiheessa koodin kääntäminen on hyvä suorittaa TwinCAT-ympäristössä laajempien debug-ominaisuuksien vuoksi. (Module generation n.d.)

Asetusten tallennuksen jälkeen testimallista generoitiin koodi valitsemalla Simulink-ympäristön yläpalkista apps-välilehti ja lisäosa Simulink coder. Ympäristön yläpalkkiin ilmestyi uusi välilehti C code, josta valittiin generate code.

Koodin generointi suoriutui ilman virheitä ja Simulink-ympäristö loi valmiin C++ projektikansion samaan kansioon, jossa auki oleva malli sijaitsi, tässä tapauksessa kansiopolussa C:\Users\Twincat\Documents\MATLAB\Examples\R2022b\simulink_general\sldemo_bounceExample. Seuraavaksi siirryttiin TwinCat-ympäristöön, jonne generoitu projekti tuotiin

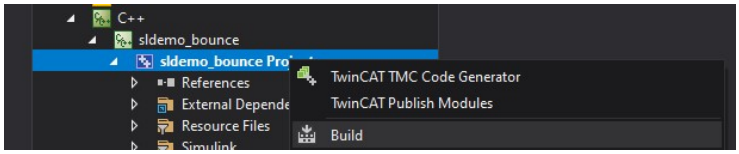
4.2.2 Koodin kääntäminen suoritettavaksi sovellukseksi

Visual Studiossa luotiin uusi TwinCAT XAE Project (XML format) ja generoitu koodi tuotiin ympäristöön lisäämällä Simulink-mallista TE1400-ohjelmistolla luotu .vcxproj projektitiedosto Visual Studion C++ valikkoon (kuvio 17). Seuraavaksi käännettiin malli suoritettavaksi sovellukseksi ja lisättiin se TwinCAT-ympäristöön.



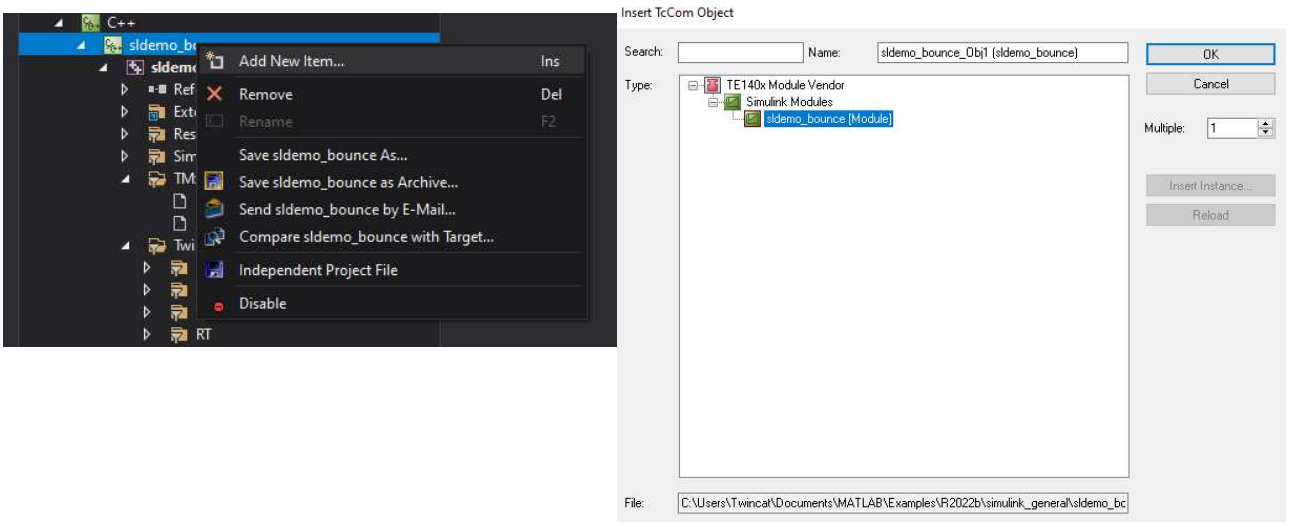
Kuvio 17, Simulink-mallista generoidun koodin lisääminen TwinCAT-ympäristöön

Koodi käännettiin valitsemalla lisätty .vcxproj tiedosto hiiren oikealla painikkeella ja valitsemalla build (kuvio 18).



Kuvio 18, Koodin kääntäminen

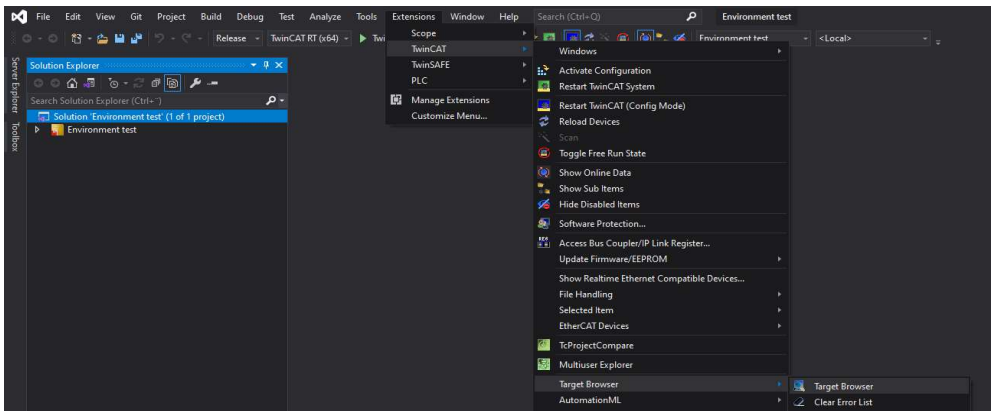
Koodi käännettiin onnistuneesti ja luotu sovellus lisättiin TwinCAT-ympäristöön valitsemalla C++ projekti hiiren oikealla painikkeella ja valitsemalla Add new item (kuvio 19). Luotu sovellus oli TcCom-objekti, eli TwinCAT component object, joka luotiin Microsoftin standardoiman component object mallin (COM) pohjalta. COM on Microsoftin määrittelemä standardi suoritettavien ohjelmistojen luomiselle. (The Component object model n.d.)



Kuvio 19, Luodun TcCom-objektin lisääminen TwinCAT-ympäristöön

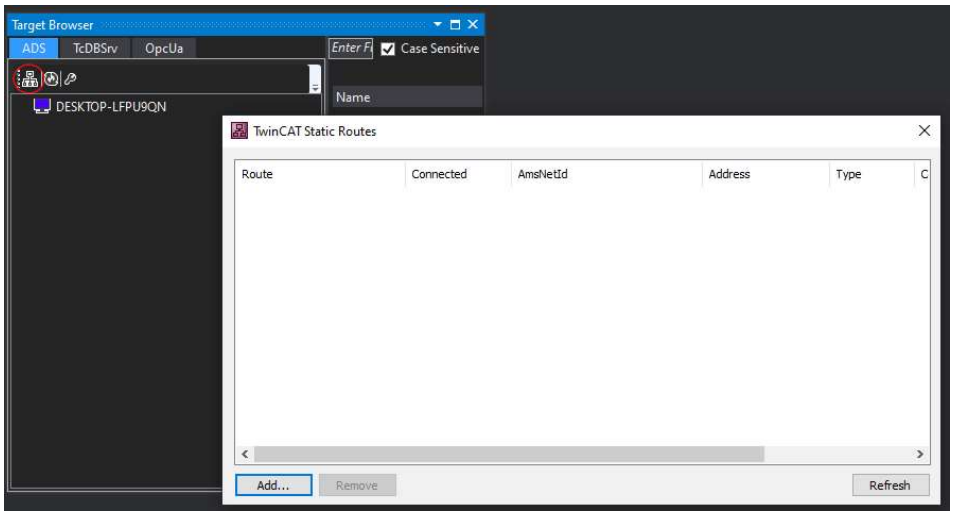
4.2.3 IPC:n yhdistäminen, reaaliaikaisuuteen liittyvät määrittelyt ja järjestelmän testaus

Seuraavaksi yhdistettiin erillinen IPC TwinCAT-ympäristöön. IPC yhdistettiin suoraan verkkokaapelilla kehityskoneen vapaaseen verkkoporttiin, koska IPC haluttiin pitää erillään julkisesta verkosta testitilan vuoksi. IPC:n ja kehityskoneen IP-asetukset määriteltiin samalle alueelle Windowsin verkkoasetuksista ja IPC lisättiin TwinCAT-ympäristöön valitsemalla TwinCAT-ympäristön ylävalikosta extensions – TwinCAT- Target browser (kuvio 20).

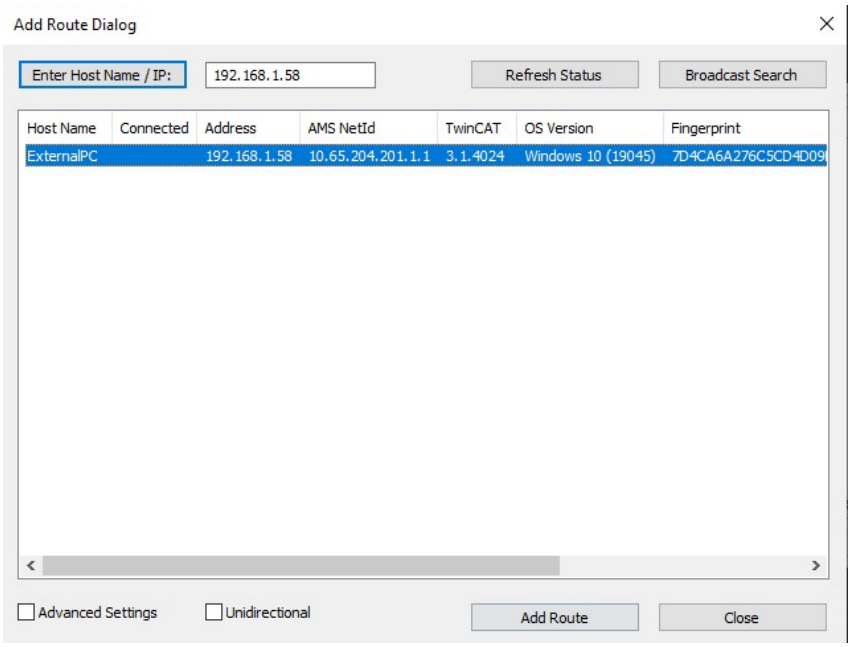


Kuvio 20, Erillisen kohdekoneen lisääminen TwinCAT-ympäristöön

Avautuvasta ikkunasta valittiin add and remove ADS routes ja uudelleen avautuvasta ikkunasta add (kuvio 21). Avautuneeseen add route dialog-ikkunaan syötettiin IPC:lle määritelty IP-osoite ja valittiin painike enter host name / ip, jonka jälkeen IPC ilmestyi näkymään (kuvio 22).

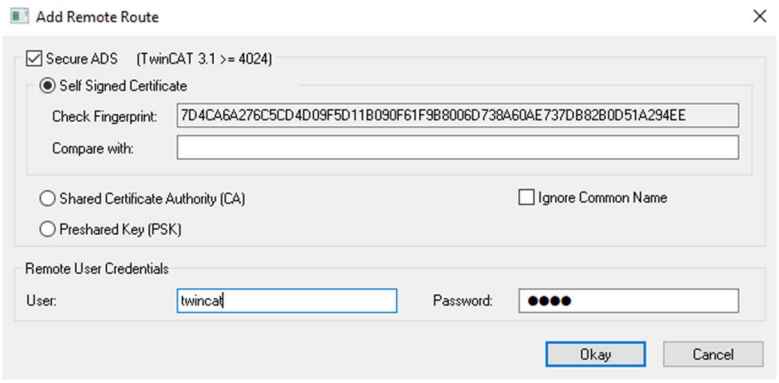


Kuvio 21, ADS reitin tallennus



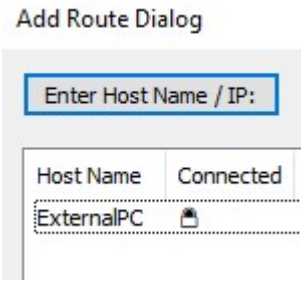
Kuvio 22, IPC löytyi ja se voidaan lisätä TwinCAT-ympäristöön

Valittiin add route-painike, jonka jälkeen avautuvaan ikkunaan lisättiin kohdekoneen kirjautumis-tiedot remote user credentials kohtaan ja valittiin okay-painike (kuvio 23).



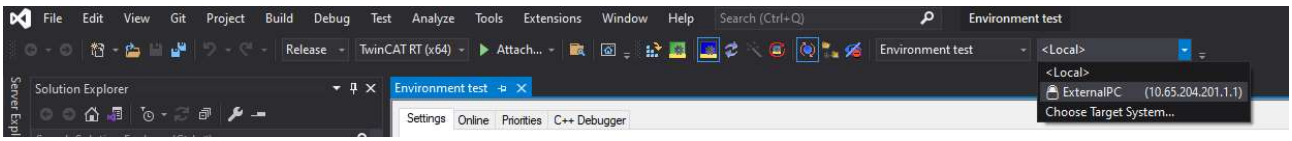
Kuvio 23, Kohdekoneen kirjautumistietojen lisääminen etäyhteyttä varten

Add route-näkymään, sarakkeeseen connected, ilmestyi lukon kuvake merkiksi siitä, että yhteyden luominen oli onnistunut (kuvio 24). Seuraavaksi edettiin tekemään loput ympäristömäärittelyt ennen sovelluksen testaamista.



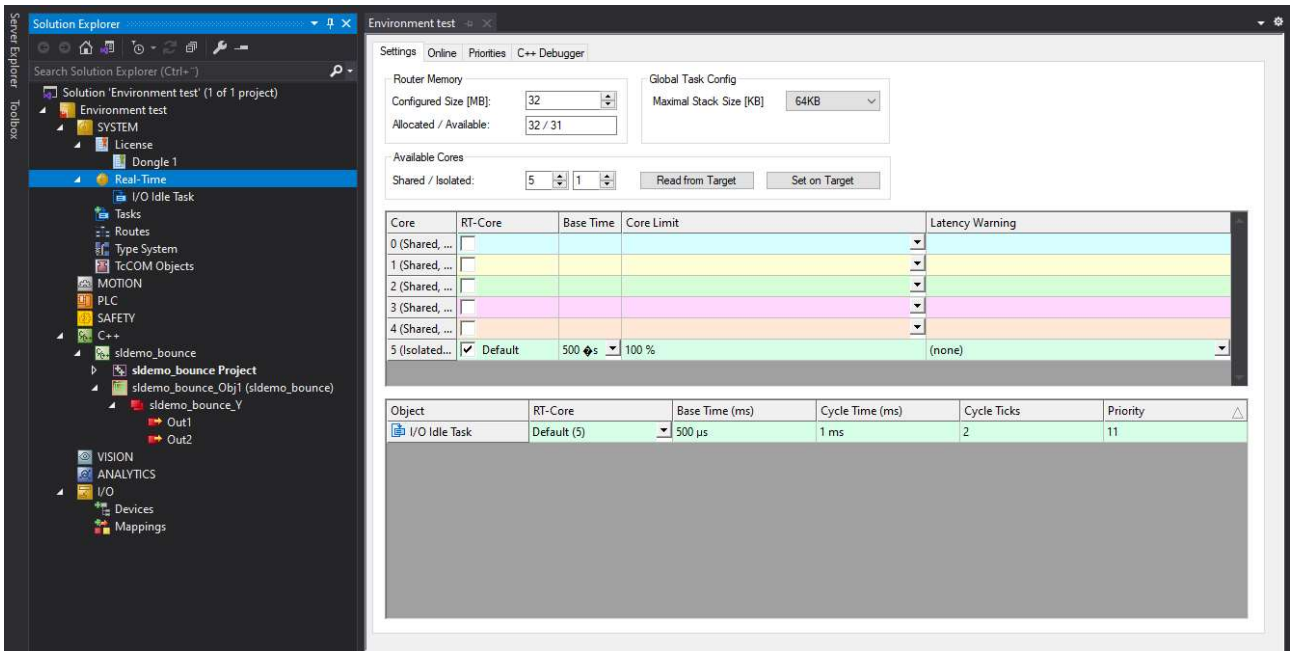
Kuvio 24, Yhteyden luominen onnistui TwinCAT-ympäristön ja kohdekoneen välillä

Seuraavaksi valittiin lisätty IPC aktiiviseksi TwinCAT-ympäristöön valitsemalla kohdejärjestelmä alasvetovalikosta kohdasta choose target system (kuvio 25). Tämän jälkeen tehtiin tarvittavat määrittelyt reaaliaika-asetuksissa kohdasta system – real-time (kuvio 26).



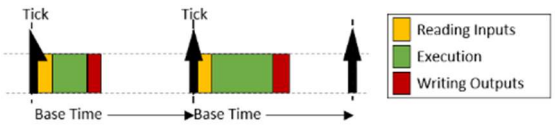
Kuvio 25, Lisätyn kohdekoneen aktivointi TwinCAT-ympäristössä

Valitsemalla painikkeen read from target ympäristö luki kohdekoneelle määritettyjen jaettujen ja eristettyjen prosessoriytimien jaon (kuvio 26). Kohdekoneella oli yhteensä 6 prosessoriydintä, joista puolet oli jaettuja ja puolet eristettyjä. Jaetut ytimet olivat koko järjestelmän käytettävissä kohdekoneella, kun taas eristetyt ytimet olivat täysin eristetty järjestelmästä siten, ettei käyttöjärjestelmä nähnyt niitä ollenkaan (TwinCAT 3 Real-time n.d.).



Kuvio 26, Yksi prosessoriydin eristettiin IPC:llä ja sen pohja-ajaksi asetettiin 0.5ms

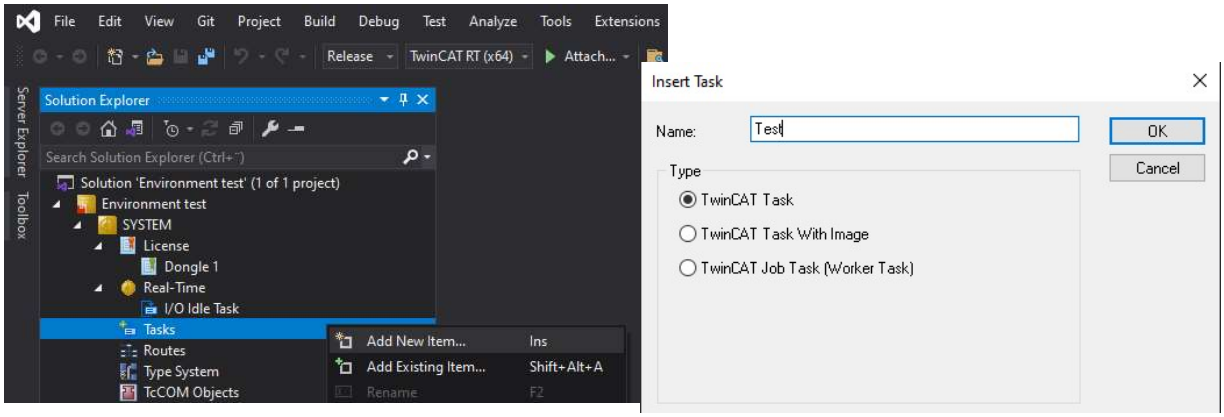
Kohdekoneen prosessoriytimet jaettiin yhteen eristettyyn ytimeen ja viiteen jaettuun ytimeen valitsemalla set on target-valinta, jonka jälkeen IPC käynnistyi uudestaan. Yhdelle eristetylle ytimelle asetettiin base time, eli pohja-aika (kuvio 26), joka oli sama kuin aikaisemmin Simulink-malliin määritelty time-step, eli simulaation aika-askel. TwinCAT-ympäristön pohja-ajan periaatteen näkee tarkemmin kuviosta 27.



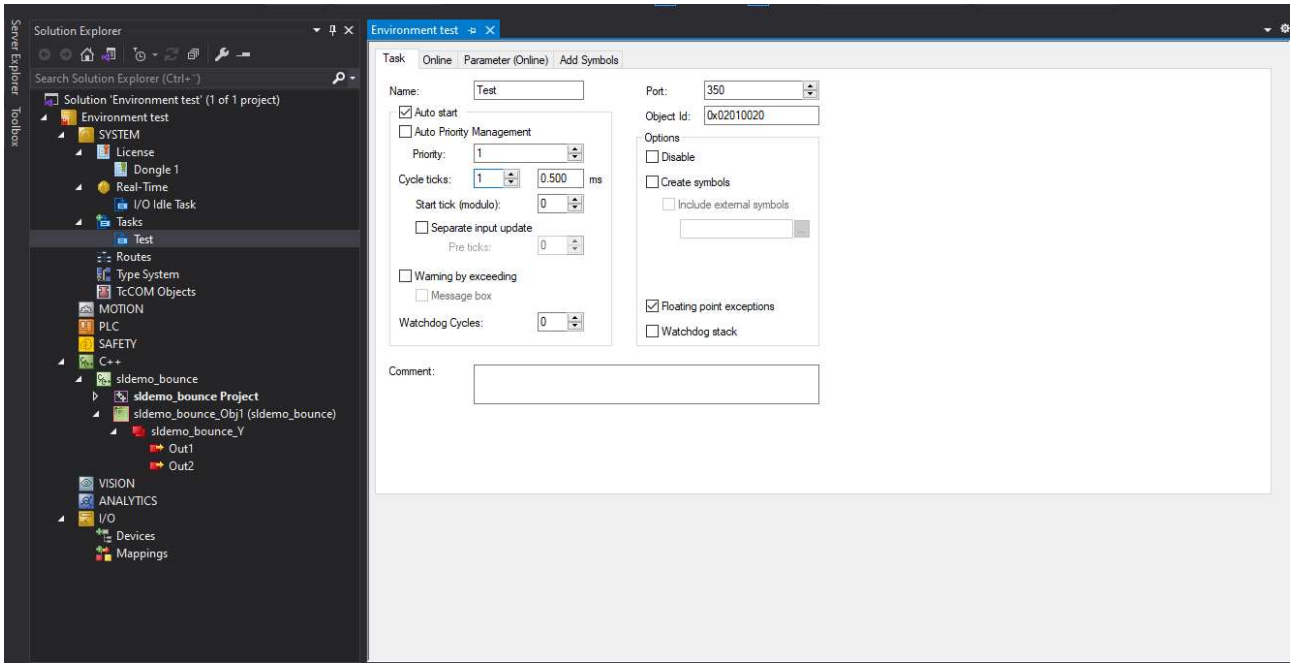
Kuvio 27, Pohja-ajan periaate (TwinCAT 3 real-time n.d.)

TwinCAT-ympäristöön lisättiin seuraavaksi task-objekti ja sille tehtiin tarvittavat määrittelyt. Task-objekti luotiin valitsemalla TwinCAT-ympäristön system-valikosta kohta tasks hiiren oikealla painikkeella ja valitsemalla add new item (kuvio 28).

Task-objekti on aikayksikkö tai kehys, jossa sovellus suoritetaan ja se automaattisesti lisättiin eristetylle ytimelle suoritettavaksi. Task-objektin asetuksista valittiin valinta auto start, jotta se asetettiin käynnistymään heti IPC:n ohjelmasuorituksen alussa ja valittiin cycle ticks-valintaan arvo 1, jolloin se asetettiin suoritettavaksi jokaisella prosessoriytimen aikajaksolla (kuvio 29).



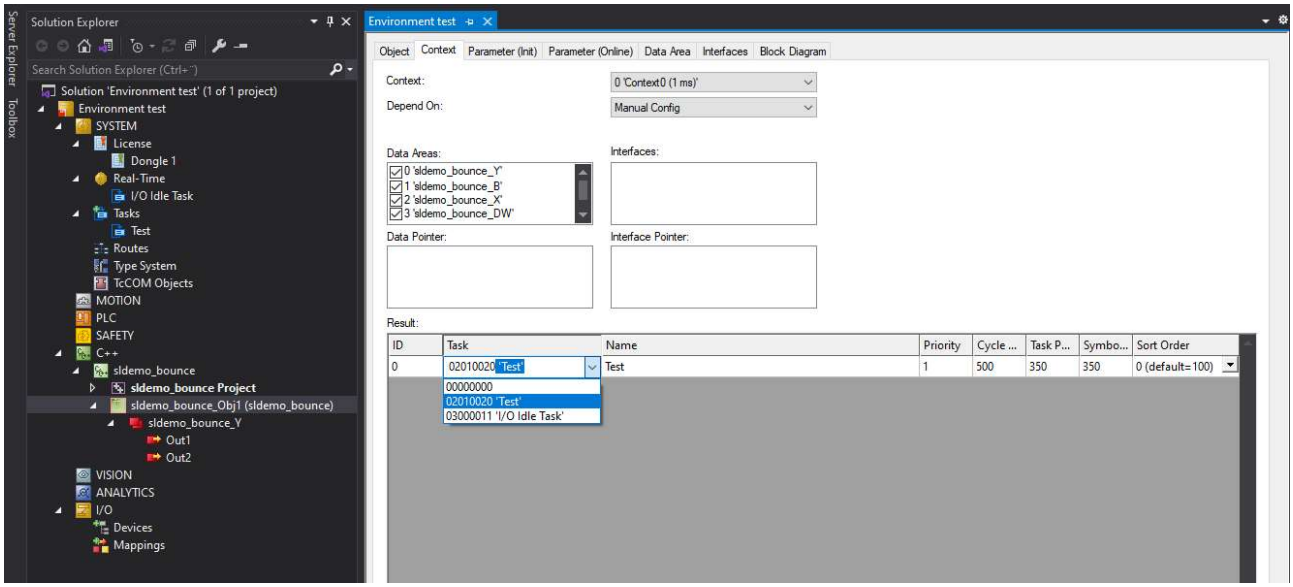
Kuvio 28, Task-objektin luominen



Kuvio 29, Task-objektin määrittelyt

Viimeisenä ennen sovelluksen ja kohdekoneen testaamista osoitettiin luotu task-objekti Simulink-mallin generoidusta koodista käännetylle TcCom-objektille. TcCom-objekti osoitettiin siten suoritettavaksi IPC:llä.

Task-objekti osoitettiin TcCom-objektille valitsemalla aikaisemmin lisätty TcCom-objekti TwinCAT-ympäristön projektinäkymästä ja valitsemalla context-välilehti asetusnäkymän yläreunasta. Tämän näkymän alaosassa valittiin luotu task-objekti task-sarakkeeseen (kuvio 30).



Kuvio 30, Task-objektin osoittaminen TcCom-objektille

TwinCAT-ympäristössä tehdyt määrittelyt olivat siten valmiit ja edettiin suorittamaan sovellus kohdekoneella. Valitsemalla TwinCAT-ympäristön yläpalkista valinta activate configuration (kuvio 31) ympäristön määritykset ja suoritettava sovellus siirtyivät IPC:lle suoritettavaksi.

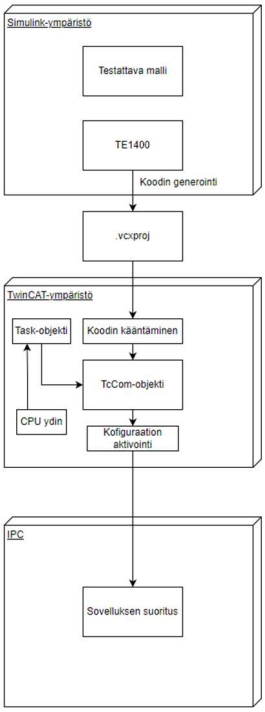
Kuvio 31, Määrittelyjen aktivointi ja ohjelman suoritus IPC:llä

Ohjelman suoritus päättyi virheilmoitukseen ja TwinCAT-ympäristön output-ikkunasta nähtiin virheen liittyvän siihen, ettei aikaisemmin luotu sertifikaatti ole IPC:llä luotettujen sertifikaattien joukossa. IPC:n kansiopolkuun C:\TwinCAT\3.1\Target\OemCertificates\ oli generoitunut .reg päätteinen tiedosto, joka tunnistettiin aikaisemmin luoduksi sertifikaatiksi nimen perusteella. Nimi sisälsi sähköpostiosoitteen, joka sertifikaattiin lisättiin sen luontivaiheessa. Kaksoisklikkaamalla tiedostoa sertifikaatti lisättiin automaattisesti järjestelmään luotettujen sertifikaattien listalle.

Activate configuration valittiin uudestaan TwinCAT-ympäristöstä, jonka jälkeen sovellus käynnistyi IPC:llä. Ohjelman suoritusta kuvasi kuvion 32 mukainen vihreä ikoni TwinCAT-ympäristön alareunassa ja sama ikoni oli nähtävissä IPC:n Windows-ympäristön alapalkissa.

Kuvio 32, Vihreä ikoni kuvaa aktiivista ohjelmansuoritusta kohdekoneella

Järjestelmien todettiin olevan nyt oikein määritelty ja voitiin edetä varsinaisen työssä käsitellyn mallin testaamiseen. Edellä luetellut askeleet pätivät pääsääntöisesti kaikkien testattavien simulink-mallien osalta ja eri vaiheet etenivät kuvion 33 mukaisella tavalla.

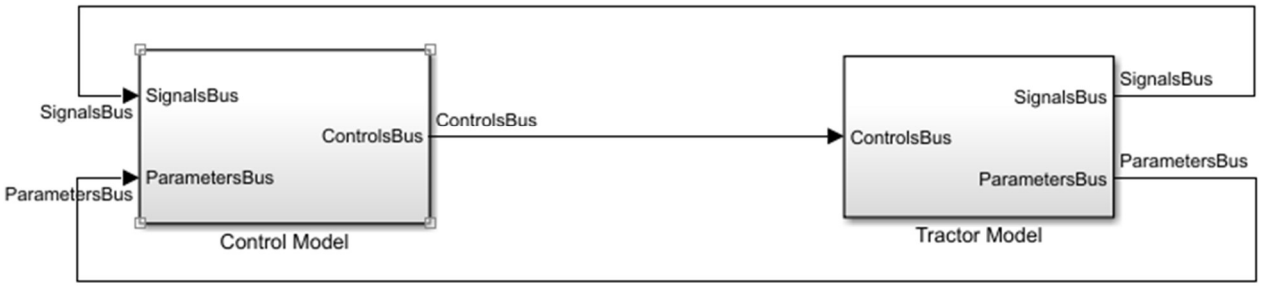


Kuvio 33, Periaatekuva Simulink-mallin testaamisesta Beckhoff-alustalla

5 Valtran simulaatiomalli

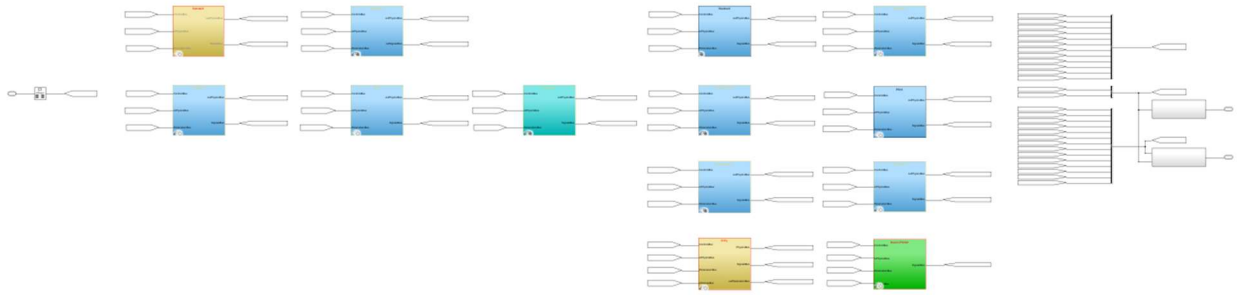
Valtran simulaatiomalli oli laaja ja monitasoinen sen sisältäessä sekä traktrorin simulaation, tractor model, että simulaation fyysisen ohjausrajapinnan, control model. Control model sisälsi nykyisenä simulaatioalustavan toimivan Speedgoat-laitteiston rajapinnan ohjauksen ja linkityksen fyysisten analogisten ja digitaalisten tulo- ja lähtökanavien sekä simulaation välillä, kuvion 34 mukaisesti.

Tässä työssä testattiin tractor model-osuuden toimintaa Beckhoff-alustalla ja control model kommentoitiin ulos kokonaisuudessaan. Signaalit, jotka liittyivät control model osuuteen poistettiin tai korvattiin mahdollisuuksien mukaan kiinteillä arvoilla ja työssä keskityttiin vain testaamaan simulaation ratkaisujen toiminta uudella alustalla eikä simulaation toimintaa kokonaisuudessaan.



Kuvio 34, Mallin päätaso, vasemmalla ohjausrajapinta ja oikealla opinnäytetyössä käsiteltävä malli

Tractor model oli rakennettu moduulipohjaisesti siten, että traktorin eri toiminnallisuudet olivat jaettu osakokonaisuuksiin ja ne yhdistyivät toisiinsa virtuaalisilla bus-elementeillä (kuvio 35). Virtuaalinen bus on Simulink-ympäristössä yksinkertaistettu esitys signaalinipusta, jonka tarkoituksena on pitää mallin sisäiset kytkennät siistinä, mutta sille ei sovellusteknisesti esimerkiksi varata muistipaikkaa tai luoda muuttujaa itsessään (Group signals or messages into virtual buses n.d.). Sen voi ajatella vastaavan reaalimaailmassa johtonippua, joka on sidottu nippusiteillä tai muulla tavalla.



Kuvio 35, Tractor model toiseksi ylimmältä tasolta nähtynä

5.1 Mallin testaaminen ja löydetyt ongelmat

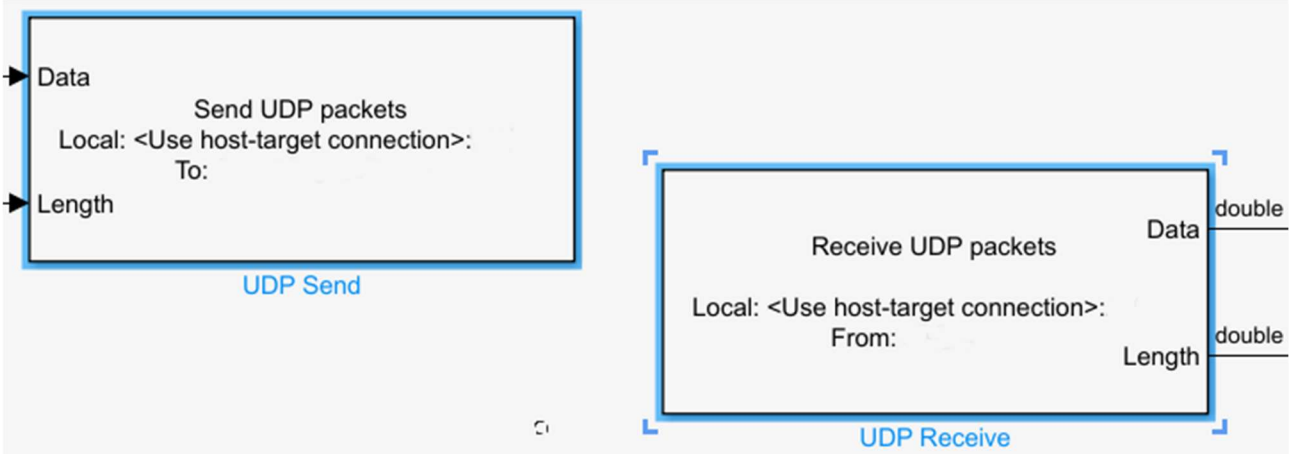
Testaaminen aloitettiin generoimalla koodi koko traktorimallin sisällöstä, mutta se päättyi määrittelemättömään virheeseen, jonka jälkeen siirryttiin testaamaan mallia moduuli kerallaan kommentoimalla kaikki muut paitsi testattava osuus ulos. Tämä osoittautui hankalaksi tavaksi testata mallia, sillä moduulit sisälsivät usein signaaleita muista moduuleista ja puuttuvat signaalit usein estivät koodin generoimisen.

Moduulit saatiin testattua korvaamalla puuttuvat signaalit kiinteillä arvoilla tai poistamalla signaalit kokonaan, joskin tämä tapa oli usein työläs ja aikaa vievä, kun moduuli saattoi olla monitasoinen ja sisään tulevia signaaleita saattoi olla useilla tasoilla. Koodin generoinnin jälkeen siirryttiin kääntövaiheeseen TwinCAT-ympäristöön, josta sovellus siirrettiin IPC:lle suoritettavaksi. Mallin laajuudesta huolimatta ongelmia tuottivat signaaliriippuvuuksia lukuun ottamatta pääasiassa vain kaksi moduulia: Lohkot, jotka hoitavat kommunikoinnin simulaation ja graafisen käyttöliittymän välillä sekä mallin fysiikkaan ja monikappaledynamiikkaan liittyvä moduuli.

5.1.1 UDP-kommunikointi

Mallissa käytettiin UDP-send ja UDP-receive nimisiä lohkoja (kuvio 36) kommunikoinnissa simulaation ja graafisen käyttöliittymän välillä. Tämä kommunikaatio sisälsi esimerkiksi traktorin asentoon liittyviä koordinaatteja, jotka lähetettiin käyttöliittymäympäristölle piirrettäväksi tai käyttöliittymästä hallittavien muuttujien, kuten polttoaineen määrän tai erilaisten anturivikojen, lähettämisen simulaatioon.

Koodin generointi moduulista, jossa UDP-lohkot olivat, päättyi epäselvään virheeseen ja tarkempi tutkimus osoitti, että nämä lohkot olivat Speedgoat-spesifejä lohkoja. Nämä aiheuttivat epämääräisen virheen, sillä TE1400-sovellus ei osaa generoida koodia Speedgoat-lohkoista.



Kuvio 36, Speedgoat spesifejä lohkoja

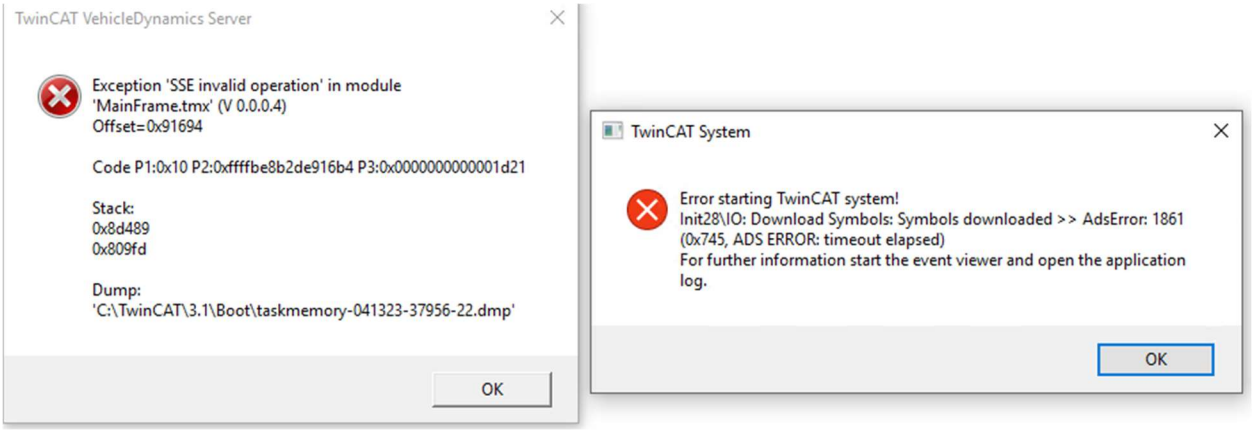
Beckhoff-alustalla UDP-lohkot korvattaisiin simulaatiossa ulos- ja sisääntuloporteilla ja TwinCAT-ympäristössä tehtäisiin sovellus käyttämällä hyväksi Beckhoff TE6311-ohjelmistomoduulia, joka hoitaisi UDP-kommunikoinnin (TF6311 | TwinCAT 3 TCP/UDP Realtime n.d.). Tehdylle sovellukselle osoitettaisiin simulaation sisään- ja ulostuloportit TwinCAT-ympäristössä. Työhön ei liittynyt tämän testaaminen käytännössä.

5.1.2 Multibody

Malliin kuului moduuli, jossa simuloitiin traktorin monikappaledynamiikka käyttäen hyödyksi Simulink-ympäristöön lisäosana saatavaa Simscape Multibody-laajennusta. Monikappaledynamiikka tarkoittaa järjestelmän mallintamista erillisinä kappaleina, jotka vuorovaikuttavat toisiinsa erilaisten nivelien ja kontaktipintojen kautta (Multibody dynamics n.d.).

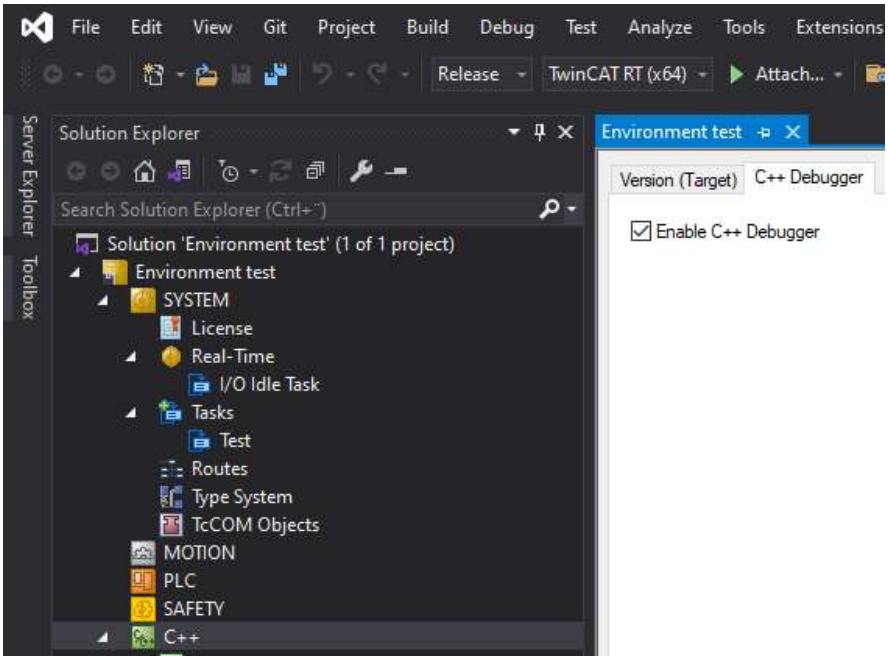
Moduulin koodin generointi ja kääntäminen onnistui ongelmitta, mutta sen suorittaminen johti IPC:n kaatumiseen välittömästi koodin suorituksen alettua. Järjestelmä ei antanut selvää vikakoodia vaan päätyi käynnistymään uudelleen. Päätettiin lähteä tutkimaan multibody-osuutta tarkemmin ja kommentoimaan ulos moduulin sisäisiä lohkoja.

Kommentoimalla ulos traktorin hyttiin liittyvä osuus saatiin koodi suoritettua IPC:llä ilman sen kaatumista, mutta ohjelman suoritus keskeytyi virheeseen. IPC ja TwinCAT-ympäristöön tulostui virheelliseen SSE-operaatioon liittyvä viesti (kuvio 37).



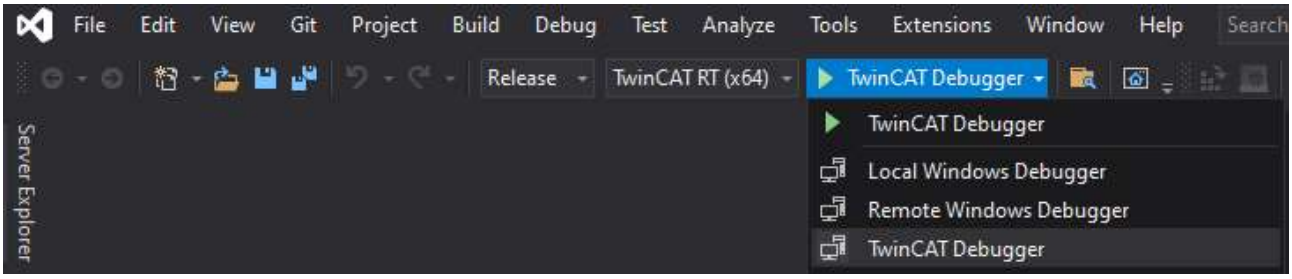
Kuvio 37, Virheellinen SSE-operaatio

Tässä vaiheessa, kun koodin suoritus päättyi virheeseen IPC:n pysyessä kuitenkin käynnissä, toisin kuin aikaisemmin, voitiin TwinCAT-ympäristössä aloittaa debug-sessio. Debug-session edellytyksenä on, että ominaisuus on aktivoitu TwinCAT-ympäristön C++ asetuksissa (kuvio 38) ennen generoidun koodin kääntämistä (TwinCAT 3 | C/C++ n.d.).

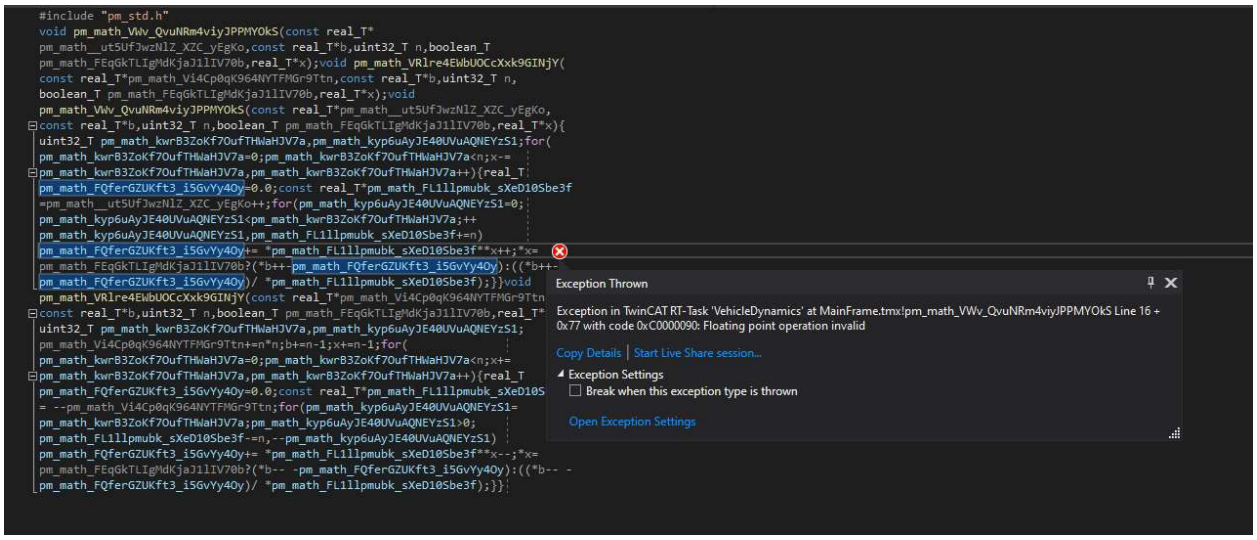


Kuvio 38, Debug-ominaisuuksien käyttöönotto

Debug-sessio avattiin TwinCAT-ympäristön yläpalkista valitsemalla TwinCAT debugger (kuvio 39). Tämä osoitti virheeseen johtaneen kohdan TcCom-objektin lähdekoodista (kuvio 40).

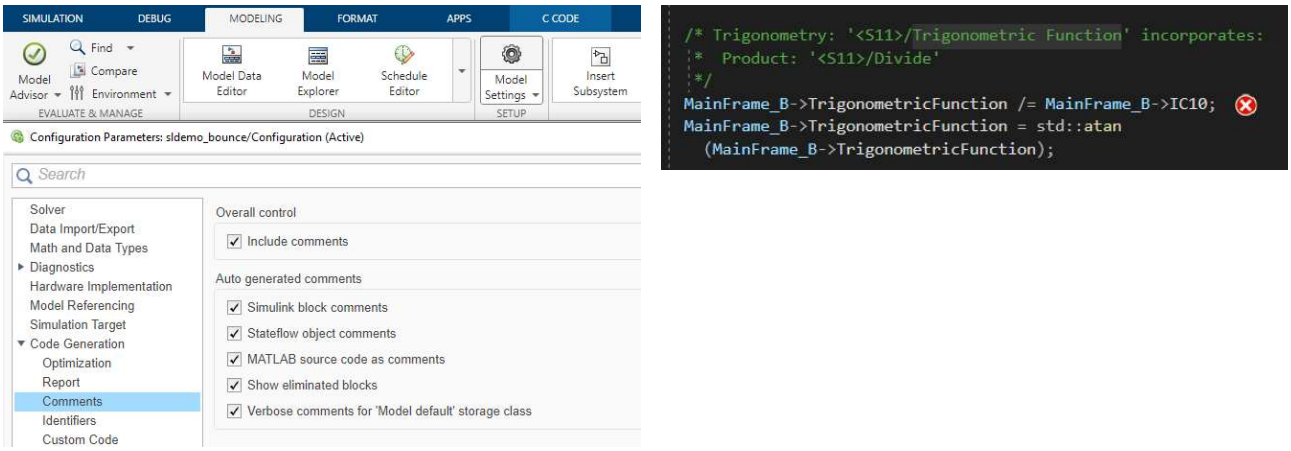


Kuvio 39, Debug-session avaaminen



Kuvio 40, Generoidun koodin tulkinta on lähes mahdotonta

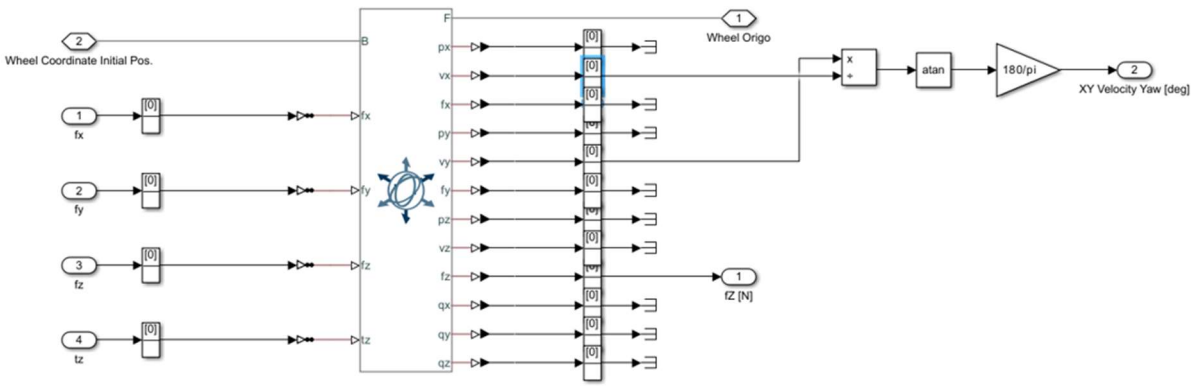
Generoitu koodi ei ollut juurikaan luettavaa, mutta sen luettavuutta voitiin helpottaa aktivoimalla generoidun koodin kommentointiin liittyvät asetukset simulink-mallista (Configure code comments n.d.). Asetukset löytyivät simulink-mallin code generation asetusten alta, kohdasta comments (kuvio 41). Asetusten muuttamisen jälkeen koodi generoitiin uudestaan simulink-mallista.



Kuvio 41, Generoidun koodin kommentointiin liittyvät asetukset Simulink-mallissa ja virheeseen johtanut koodi kommentoituna

Tämän jälkeen koodi oli huomattavasti helpommin luettavissa (kuvio 41) ja virheen nähtiin liittyvän jakolaskuun. Aikaisemmin TwinCAT-ympäristöön auennut ilmoitus virheellisestä SSE-operaatiosta antoi viitteitä mahdollisesta nollalla jaosta. SSE-virhe TwinCAT-ympäristössä tarkoittaa virheellistä liukulukuoperaatiota, yleensä laskuoperaatiota, jonka tulos ei ole määriteltävissä, kuten nollalla jaettaessa (TwinCAT 3 Real-time n.d.).

Multibody-moduulista löytyi lopulta initial condition-lohko, joka oli määritelty asettamaan signaalin arvoksi nolla simulaation alussa (kuvio 42). Tämä lohko oli kytketty jakolaskuun, joka aiheutti virheen. Vastaavia tilanteita löytyi moduulista yhteensä viisi kappaletta ja ne korjattiin määrittämällä initial condition-lohkon arvoksi yksi.



Kuvio 42, Sinisellä korostettu pieni lohko on "Initial condition" joka johtaa nollalla jakamiseen

Näiden korjausten jälkeen jättämällä hytin osuus pois moduulista simulaatio käynnistyi ilman virheitä myös IPC:llä. Tämän myötä tultiin lopputulokseen, että simulaatio olisi mahdollista suorittaa kokonaisuudessaan Beckhoff-alustalla tietyin muokkauksin ja korvaamalla osa Simulink-mallin sisäisistä toiminnoista TwinCAT-ympäristössä toteutetuilla ratkaisuilla.

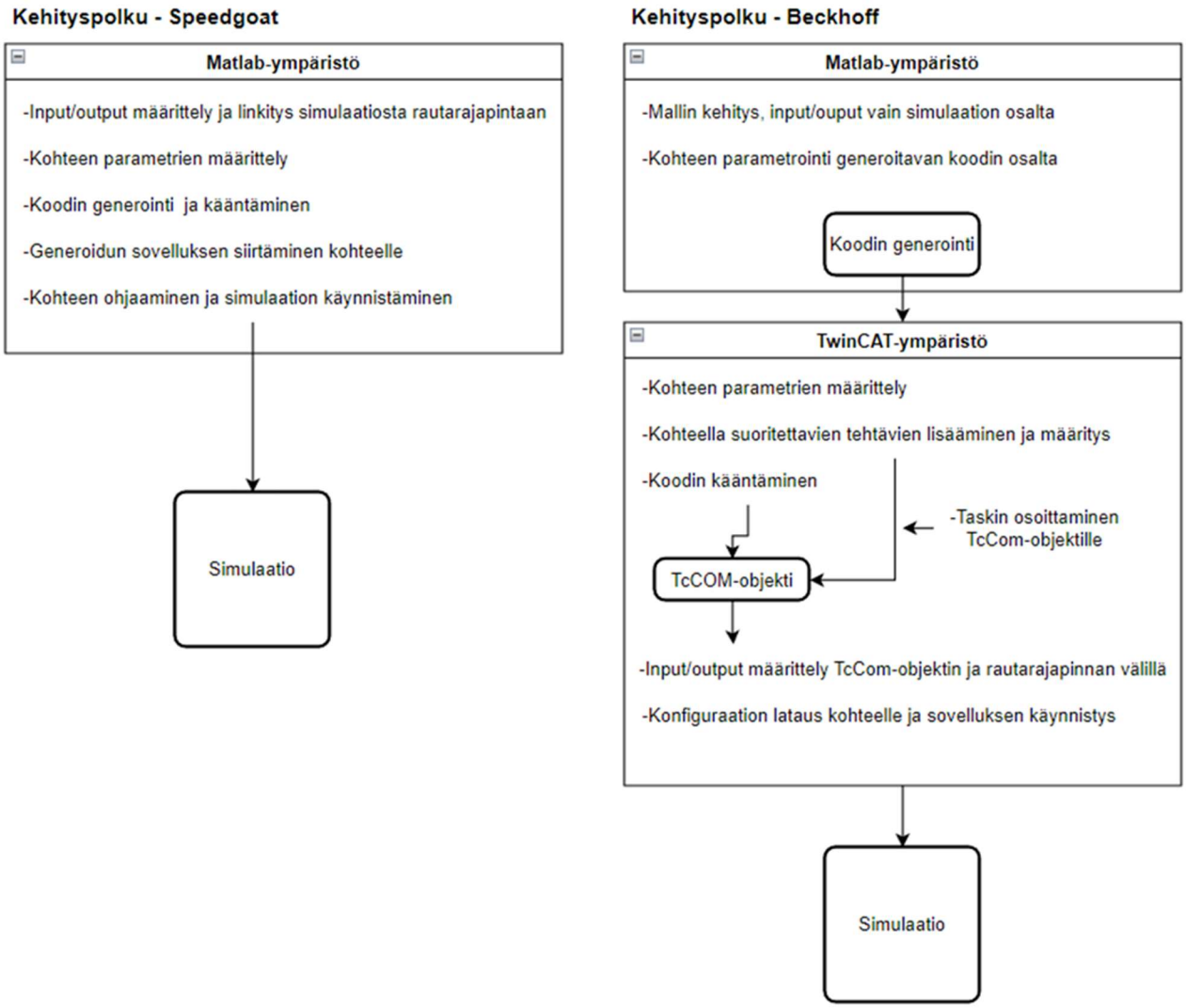
5.2 Alustojen erot kehitysnäkökulmasta

Työn aikana syntyi kohtalaisen kattava kuva siitä, kuinka simulaatiota kehitettiin TwinCAT-alustalle ja mitä askeleita siihen sisältyi. Kokonaisuudessaan kehittäminen Beckhoff-alustalle sisälsi enemmän vaiheita ja parametroitavia asetuksia kuin kehittäminen Speedgoat-alustalle, kun Simulink-ympäristön rinnalle tuli lisäksi TwinCAT-ympäristö.

Mallin kehitys Speedgoat-alustalle tapahtui kokonaisuudessaan Simulink-ympäristössä, kehityksestä julkaisuun, sillä kaikki tarvittavat toiminnallisuudet joko kuuluivat valmiiksi Simulink-ympäristöön tai ne olivat saatavina Simulink-laajennuksina. Lisäosana asennettava Simulink Real-Time Target Support Package mahdollisti Speedgoat-kohdejärjestelmän hallinnan Simulink-ympäristöstä.

Beckhoff-alustaa ajatellen mallin kehitys, lokaali testaus kehityskoneella sekä koodin generointi tehtiin Simulink-ympäristössä, jonka jälkeen siirryttiin TwinCAT-ympäristöön. Koodin generointia varten kehityskoneelle asennettiin Beckhoff TE1400-ohjelmistopaketti, joka mahdollisti koodin generoinnin Simulink-ympäristöstä. Ohjelmistopaketti vaati erikseen Beckhoffilta ostettavan lisenssin ja oli yksi harvoja Beckhoff sovellustuotteita, joita ei juurikaan voitu käytää kokeilulisenssillä. Tätä työtä varten TE1400-lisenssi saatiin ilmaiseksi työn ajaksi testiin Beckhoffilta, eikä sen aktivointia käsitelty tässä työssä.

Simulink-ympäristöstä voidaan luoda suoraan valmiita suoritettavia sovelluksia TwinCAT-ympäristöön, mutta kehitysvaiheessa on järkevämpää vain generoida koodi Simulink-ympäristöstä ja tuoda se TwinCAT-ympäristöön, jotta voidaan hyödyntää TwinCAT-ympäristön debug-ominaisuuksia. Kuviossa 43 on esitelty pääpiirteittäin alustojen väliset erot kehityksestä simulaatioon.



Kuvio 43, Vuokaavioesitys kehityspolkujen vertailusta

5.3 Johtopäätös

Työssä saavuttiin johtopäätökseen, että simulaation siirtäminen Speedgoat-alustalta Beckhoff-alustalle on teknisesti mahdollista, kun pääosat traktorin simulaatiomallista saatiin testattua Beckhoff-alustalla. Siirtäminen vaatisi kuitenkin mallin kehittämistä ainakin osittain uusiksi, jotta ratkaisut saataisiin toimimaan yhdessä myös Beckhoff-alustalla. Työn aikarajoitteista johtuen alustojen välistä suorituskykyä ei päästy testaamaan, eikä simulaatiomallia suoritettu kokonaisuudessaan vaan mallin toiminta testattiin moduuleittain, joten Beckhoff-järjestelmän suorituskyvystä tässä applikaatiossa ei saatu täyttä varmuutta.

Suurin osatekijä alustojen suorituskyvyssä oli niiden käyttämä prosessori, jotka tässä työssä molemmissa tapauksissa oli Intel-valmistajan kuluttajatasoinen prosessori. Tässä työssä Beckhoff-järjestelmän prosessori oli Intel Xeon-mallin prosessori, kun Speedgoat-järjestelmän prosessori oli Intel i7-7700 prosessori. Kun vertailtiin prosessorien yksittäisen ytimen suorituskykyä, todettiin i7 prosessorin ytimen olevan noin 20 % tehokkaampi kuin Xeon-ytimen. Tämän pohjalta todettiin, että jos tutkimusta jatketaan Beckhoff-alustaa ajatellen, tulee prosessorin valintaan kiinnittää tarkasti huomiota.

Simulaation kehittäminen Beckhoff-alustalle vaikutti aluksi hitaammalta kuin Simulinkistä suoraan Speedgoat-alustalle, mutta kun menetelmät työn aikana vakiintuivat ja TwinCAT-ympäristö parametrisoitiin oikein, todettiin että ajalliset erot simulaatiokehityksessä eivät välttämättä olleet merkittäviä.

6 Pohdinta

Vaikka työssä jokseenkin saavutettiin asetetut tavoitteet, heräsi työn aikana useita tutkittavia aiheita, joita tässä työssä ei käsitelty. HIL-simulaatioon liittyen työssä ei käsitelty aiheen rajaussyistä ollenkaan kontrollipuoleen liittyviä rautarajapinnan ratkaisuja, jonka vuoksi järjestelmien hintavertailu oli vaikeaa. Speedgoat-järjestelmässä oli esimerkiksi analogisia virtakanavia, joiden virta-alue on epätyypillisesti 0-100mA sekä kontrolloitavia resistanssikanavia, joille ei suoraan löydy Beckhoff-vastinetta. Speedgoat myös toimittaa kokonaisratkaisuja, kun Beckhoff on enemmän perinteinen komponenttitoimittaja. Beckhoff-laitteiston käyttöönottaminen vaatii siten syvempää sähkötekniistä osaamista kuin valmiiksi paketoitu Speedgoat-alusta.

Työ antoi paljon näkemystä TwinCAT-ympäristön mahdollisuuksista ja vaikka kehitysnäkökulmasta Speedgoat ja Simulink yhdessä on tehokkaampi ja suoraviivaisempi tapa rakentaa toimiva simulointiympäristö, tarjoaa Beckhoff joustavamman ympäristön simulaation suorittamiseen, kun simulaatio voidaan hajauttaa verrattain helposti usealle prosessoriytimelle. TwinCAT-ympäristön muut ominaisuudet ovat myös käytettävissä simulaation rinnalla.

Työn luotettavuuden ja eettisyyden kannalta haastavinta oli aiheen laajuus ja työn rajaus aiheutti haasteita järjestelmän monimutkaisuuden ja laajuuden vuoksi. Itse työ ja tutkittavat asiat olivat

laajempia kuin mitä tässä työssä on raportoitu, sillä kaiken työn sisällön sisällyttäminen koherenttiin ja luettavaan opinnäytetyöhön olisi ollut hyvin haastavaa. Lopulta työhön onnistuttiin sisällyttämään oleelliset asiat ja työ toivottavasti toimii pohjana uusille opinnäytetöille.

Työ toimi henkilökohtaisesti loistavana perehdytyksenä siihen, kuinka simulaatiojärjestelmä toimii, kuinka Matlab/Simulink-ympäristöjä käytetään tehokkaasti, millaista yleensä ottaen on Hardware-in-the-loop-simulaatio ja mitkä ovat sen mahdollisuudet. Oppimisen näkökulmasta työ oli tuloksiaan huomattavasti arvokkaampi kokonaisuus ja kiitän Valtra Oy Ab:ta tästä mahdollisuudesta.

Lähteet

About Speedgoat, N.d., Verkkosivu, Viitattu 13.12.2023, https://www.speedgoat.com/company/about-speedgoat

Beckhoff-konserni, N.d., Verkkosivu, Viitattu 13.12.2023, https://www.beckhoff.com/fi-fi/company/beckhoff-group/

Bélanger, J., Venne, P. & Paquin J.-N., N.d., The What, Where and Why of Real-Time Simulation, Artikkeli, Viitattu 11.12.2023, https://blob.opal-rt.com/medias/L00161_0436.pdf

Burnham, K., Copp, D., Mouzakitis, A. & Parker, R., 2009, Hardwarein-the-loop system for testing automotive ECU diagnostic software, verkkoartikkeli, julkaistu Measurement + Control-lehden julkaisussa 42/8 lokakuussa 2009 sivuilla 238 – 245, Viitattu 11.12.2023, https://www.researchgate.net/publication/270607640_Hardware-in-the-Loop_System_for_Testing_Automotive_Ecu_Diagnostic_Software

Choose a Solver, N.d., Mathworks verkkosivu, Viitattu 14.12.2023, https://se.mathworks.com/help/simulink/ug/choose-a-solver.html

Code generation configuration, N.d., Mathworks verkkosivu, Viitattu 14.12.2023, https://se.mathworks.com/help/rtw/ug/code-generation-configuration.html

Configure code comments, N.d., Mathworks verkkosivu, Viitattu 15.12.2023, https://se.mathworks.com/help/rtw/ug/configure-code-comments.html

Enable loading of test signed drivers, N.d., Microsoft verkkosivu, Viitattu 13.12.2023, https://learn.microsoft.com/en-us/windows-hardware/drivers/install/the-testsigning-boot-configuration-option

Group signals or messages into virtual buses, N.d., Mathworks verkkosivu, Viitattu 15.12.2023, https://se.mathworks.com/help/simulink/ug/getting-started-with-buses.html

Kernel-mode code signing requirements, N.d., Microsoft verkkosivu, Viitattu 13.12.2023, https://learn.microsoft.com/en-us/windows-hardware/drivers/install/kernel-mode-code-signing-requirements--windows-vista-and-later-

Module generation, N.d., Beckhoff verkkosivu, Viitattu 14.12.2023, https://infosys.beckhoff.com/english.php?content=../content/1033/te1400_tc3_target_matlab/189857803.html&id=

Multibody dynamics N.d., Mathworks verkkosivu, Viitattu 15.12.2023, https://se.mathworks.com/help/sm/multibody-dynamics.html

PC-based Control, 2019, Beckhoffin toimittama kalvosarja, Viitattu 13.12.2023

Setting up driver signing, N.d., Beckhoff verkkosivu, Viitattu 13.12.2023, https://infosys.beckhoff.com/english.php?content=../content/1033/te1400_tc3_target_matlab/10803301643.html&id=

Simulink Fundamentals, 2022, Mathworksin toimittama kalvosarja, Viitattu 13.12.2023

TE1400 | TwinCAT 3 Target for Simulink, N.d., Beckhoff verkkosivu, Viitattu 13.12.2023, https://infosys.beckhoff.com/english.php?content=../content/1033/te1400_tc3_target_matlab/189854731.html&id=

TF6311 | TwinCAT 3 TCP/UDP Realtime, N.d., Beckhoff verkkosivu, Viitattu 15.12.2023, https://infosys.beckhoff.com/english.php?content=../content/1033/tf6311_tc3_tcpudp/1076926091.html&id=%7D

The component object model, N.d., Microsoft verkkosivu, Viitattu 14.12.2023, https://learn.microsoft.com/en-us/windows/win32/com/the-component-object-model

Third-Party Connections, N.d., Verkkosivu, Viitattu 13.12.2023, https://www.speedgoat.com/products-services/third-party-connections

TwinCAT 3 Real-Time, N.d., Beckhoff verkkosivu, Viitattu 14.12.2023, https://infosys.beckhoff.com/english.php?content=../content/1033/te1010_tc3_realtime_monitor/6828869003.html&id=%7D

TwinCAT 3 | C/C++, N.d., Beckhoff verkkosivu, Viitattu 15.12.2023, https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_c/1028235531.html&id=

User mode and kernel mode, N.d., Microsoft verkkosivu, Viitattu 13.12.2023, https://learn.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/user-mode-and-kernel-mode