Tampereen ammattikorkeakoulu

Tuotantoympäristön Industry 4.0 -tietoliikennejärjestelmän kehittäminen

Topi Kärki

OPINNÄYTETYÖ
Toukokuu 2022

Konetekniikka
Koneautomaatio

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Konetekniikka
Koneautomaatio

KÄRKI, TOPI:
Tuotantoympäristön Industry 4.0 -tietoliikennejärjestelmän kehittäminen

Opinnäytetyö 77 sivua, joista liitteitä 10 sivua
Toukokuu 2022

Opinnäytetyön tavoitteena oli tutkia TAMK:n FieldLab-tilan tämänhetkisten robottien MQTT-pohjaista kommunikaatiota. Tutkittiin ABB:n IoT Gateway -ohjelmaa ja LabVIEW LVMQTT -kirjastoon perustuvia ohjelmia NI myRIO-1900 -laitteilla. Tutkimuksen lisäksi suunniteltiin ja rakennettiin kommunikaatiojärjestelmä robottien välille.

Laitteiston tutkimuksessa ohjelmilla lähetettiin viestejä ja tutkittiin, kuinka suurella viestinlähettämis- tai -käsittelytaajuudella viestintää voidaan suorittaa. Lisäksi tarkkailtiin, kuinka luotettavaa viestintä on. IoT Gateway -sovelluksen kohdalla keskitytään Timer- ja RapidVariable-pohjaisiin viestintätapahtumiin. Tarkastelussa hyödynnettiin viestinumerolaskureita ja aikaleimoja.

Tutkimuksen tuloksiksi saatiin seuraavia tietoja. IoT Gatewayn ajastettu lähetys toimii nopeimmillaan n. 100 ms ajastimella. Muuttujiin perustuva vaatii lähemmäs 400 ms tapahtumien välille. Monet samanaikaiset viestintätapahtumat häiritsevät toisiaan suuresti. myRIO pystyy kirjaston avulla taas lähettämään viestejä n. 2 kHz taajuudella ja viestien vastaanottaminen on noin 1,5 kHz taajuudella.

Suunnitellun kommunikaatiojärjestelmän LabVIEW-ohjelmaa varten saatiin rakennettua kaikki toiminnot onnistuneesti, mutta kokonaisuudessa on huomattavia ongelmia. Yhteys ei pysy toiminnassa asetettujen määreiden mukaisesti ja IRQ-perusteinen viestin lähettäminen ja monien viestien vastaanottaminen rikkovat ohjelman. IoT Gateway -sovelluksella saatiin luotua täysin toimiva kommunikaatioelin.

Työn tavoitteisiin päästiin melkein täysin, ainoa epäonnistuminen oli LabVIEW-ohjelman haasteet. Rajallisen aikataulun vuoksi asiaa ei ehditty ratkaista, mutta työn tilaaja silti hyväksyy tuloksen. Koska viat ovat selkeitä ja tutkittavat kohteet selvillä, tilanne on jatkokehityksen kannalta otollinen. Ohjelman muokattavuutta voitaisiin myös parantaa. Muitakin tilan robotteja voisi koettaa lisätä järjestelmään, ohjelman korjaamisen jälkeen.

Asiasanat: mqtt, myRIO, iiot, labVIEW

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Mechanical Engineering
Machine Automation

KÄRKI, TOPI:
Retrofitting an Industry 4.0 System onto Production Environment

Bachelor's thesis 77 pages, appendices 10 pages
May 2022

The purpose of this thesis is to get a general understanding of the MQTT based communication capability for ABB's IoT Gateway software and a NI myRIO-1900 device using the LVMQTT LabVIEW library, and to design and build a communication system between set robots.

The capabilities measured are message sending/receiving frequency and reliability. Test messages log the send/receiving numbers and timestamps. For IoT Gateway only Timer and RapidVariable triggers are tested.

The measurement results are as follows: IoT Gateway Timer trigger can send a message at 9 Hz and RapidVariable at 2.5-2.8 Hz. myRIO can send messages at 2 kHz and receive them at 1.5 kHz. There are some errors in these numbers as they are averages.

The software development for a LabVIEW MQTT client was a partial success. The development of all necessary functions was successful but combining them created problems. The IRQ message sender crashed the program, and it could not maintain a connection to the broker according to the configuration. IoT Gateway software had no issues.

Overall, the project was a success. The lack of a working product was expected. The project has very clear paths for continued development e.g., fixing the bugs in the LabVIEW program.

Key words: mqtt, myRIO, iiot, labVIEW

SISÄLLYS

LYHENTEET JA TERMIT

IIoT	Industrial Internet of things
MQTT	Message Queuing Telemetry Transport
TCP/IP	Transmission Control Protocol/Internet Protocol
OASIS	Organization for the Advancement of Structured Information Standards
MB	MegaByte, Megatavu
QoS	Quality of Service, MQTT viestiliikenneasetus
NI	National Instruments
RIO	Reconfigurable Input/Output
I/O	Input/Output
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
FPGA	Field Programmable Gate Array
DI/O	Digital Input/Output
AI/O	Analog Input/Output
AUX	Auxiliary connector
PLC	Programmable Logic Controller
TTC	Time To Completion
IRQ	Interrupt Request
regex	Regular Expression
NIST	National Institute of Standards and Technology
WLAN	Wireless Local Area Network
ms	millisekunti
s	sekunti
Hz	Hertsi

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on selvittää Tampereen Ammattikorkeakoulun F0-kerroksen FieldLab-tilan nykylaitteiston valmiuksia robottien väliseen tiedonsiirtoon sekä suunnitella ja pilotoida robottien välinen tietoliikennejärjestelmä siinä määrin, mitä nykyinen laitteisto sen mahdollistaa. Pilotoitu järjestelmä dokumentoidaan tulevia käyttäjiä ja jatkokehitystä varten. Lisäksi pilotin aikana tapahtuneiden havaintojen ja testien tuloksien avulla pyritään luomaan jatkokehitysehdotuksia. Työn tuloksena syntyneen järjestelmän on tarkoitus toimia pohjana, jota voidaan jatkokehittää TAMK:n tuleviin tarpeisiin.

TAMK:lla on halukkuutta tutkia erityisesti MQTT-kommunikaatioprotokollan sekä National Instruments yrityksen NI myRIO-1900 -tuotteiden käyttöä tiedonsiirron suorittamisessa. Tästä syystä ne ovat hyvin keskeisiä analyysin kohteita, sekä pilotointijärjestelmän keskeisiä osia.

Ensimmäinen tehtävä projektissa on suorittaa nykyisen laitteiston toiminnallisuuden selvittäminen. Tarkoituksena on saada korkean tason käsitys siitä, millaisia rajoitteita ja mitä toiminnallisia mahdollisuuksia laitteistolla on. Haluttu tieto on esimerkiksi, kuinka nopeasti viestintää voidaan suorittaa, sen luotettavuutta esim. kuinka usein viestejä ei lähetetä tai vastaanoteta ja viestinnän eri toteutustavat esim. millä metodeilla viestilähetyksiä voidaan suorittaa sekä miten se vaikuttaa toimintaan. Projektin aikataulutus ei mahdollista kaikkien parametrikombinaatioiden perusteellista tutkimista, jolloin tutkinta painottuu optimaalisten olosuhteiden ympärille. Saatavat tulokset luovat pohjan sille, onko kyseinen työtehtävä edes mahdollista suorittaa järjestelmällä.

Toinen osa työtä on suunnitella tietoliikennejärjestelmä ja suorittaa sen toteuttamiseen vaadittava ohjelmistokehitys. Tähän kuuluu robottien ohjelmalliset muokkaukset sekä kommunikaatiolaitteiden konfigurointi ja myRIO-1900:lla käytettävien LabVIEW-ohjelmien luominen. Ohjelmistoa luodaan paloittain jolloin lähetys ja vastaanottaminen kehitetään ja testataan erikseen, jonka jälkeen ne yhdistetään kokonaiseksi ohjelmaksi.

Projektin järjestelmä on rajattu käyttämään vain MQTT-protokollaa. Muita kommunikaatioprotokollia ei siis arvioida. Laitteistollisesti projekti on rajattu käsittelemään FieldLab-tilan ABB IRB 4600 -, ABB IRB 140 - ja Universal Robots UR10 -robotteja. IRB 140 ja UR10 -robottien kommunikaatio suoritetaan myRIO-laitteiden avulla ja IRB 4600 käyttää ABB:n omaa IoT Gateway -sovellusta.

2 KÄYTETTÄVÄT TEKNOLOGIAT JA LAITTEISTO

2.1 Industry 4.0

Industry 4.0 on termi, jonka tarkka tarkoitus voi vaihdella henkilöiden ja yrityksien välillä, mutta yleisesti sen avulla kuvataan neljättä teollisuuden vallankumousta tai älykkäiden koneiden tai tehtaiden ja IIoT:n nousua. Teollisista kokonaisuuksista syntyy kyberfyysisiä järjestelmiä, jotka keräävät ja jakavat dataa kasvavissa määrissä ja skaaloissa. Koneet ja tilat yhdistyvät läheisemmiksi kokonaisuuksiksi. Ei ole yksittäisiä työpisteitä vaan yhden kokonaisuuden vuorovaikutuksessa olevia toimielimiä. Industry 4.0 mahdollistaa samanaikaisesti valmistusprosessien, valmistuslaitteiden ja tuotteiden paranemista sekä lisäksi se tarjoaa mahdollisuuksia luoda uutta arvoa näihin valmistusprosesseihin ja tuotteisiin. (Gilchrist 2016, Chapter 13)

Industry 4.0 alkaa valmistustasolta, jossa hyväksikäytetään kehittyneitä ohjaus-, tiedonkeruu- ja kommunikaatioteknologioissa. Nämä teknologiat mahdollistava automaation ja digitalisaation tason nostamista teollisissa prosesseissa. Näiden toimien avulla saadaan luotua tehokkaampia valmistusprosesseja, jolloin tuotteiden laatu nousee ja hukka vähenee. Vaikka tuotantokustannukset voisivat nousta niin tuotteen laadun ja arvon nousulla saadaan pidettyä asiakassuhteita parempina ja myynnin kokonaismäärä saadaan kasvamaan. (Gilchrist 2016. Chapter 13)

Tuotetasolla Industry 4.0 avaa myös uusia mahdollisuuksia valmistajille ja asiakkaille. Industry 4.0 teknologioilla saadaan mahdollistettua uusien ja parempien tuotteiden ympärille rakennettavien palveluiden tarjoaminen. Palveluilla saadaan luotua uutta arvoa asiakkaalle esimerkiksi data-analytiikan avulla, pidentämällä elinikää ja tarjoamalla suurempaa joustavuutta asiakkaiden omiin tarpeisiin ja haluihin. (Gilchrist 2016. Chapter 13)

Glichrist (2016, 207–210) sanoo Industry 4.0 hyödyntävien valmistusprosessien perustan koostuvan kuudesta suunnittelun periaatteesta.

1. Yhteentoimivuus
2. Visualisointi
3. Desentralisaatio
4. Reaaliaikainen valmius
5. Palveluihin keskittyminen
6. Modulaarisuus

Yhteentoimivuudella pyritään siihen, että kaikki valmistusprosessin osat, työntekijät, laitteet sekä tuotteet ovat jatkuvassa vuorovaikutuksessa toistensa kanssa ja pystyvät kommunikoimaan ja operoimaan toistensa kanssa IoT teknologioiden avulla. Virtualisointi mahdollistaa prosessin manipuloinnin halutulla tavalla simuloimaan muutoksia ja tilanteita. Tällä mahdollistetaan tehokasta ja häiriötöntä kehitys- ja kokeilutyötä. Desentralisaatio tarkoittaa keskeisten ohjausjärjestelmien purkamista, mahdollistaen yksittäisten solujen tekevän omatoimisia päätöksiä lopputuloksen saavuttamiseen. Reaaliaikaisella valmiudella halutaan painottaa ohjausta ja monitorointia reaaliaikaisesti, joka vaatii valmistusprosesseihin datan keruuta ja takaisinkytkentöjä. Palveluihin keskittyminen siirtää valmistajan keskittymisen fyysisestä tuotteesta tai ohjelmistosta sen ympärille rakennettaviin asiakkaalle tarjottaviin tuotteeseen liittyviin palveluihin esimerkiksi huoltopalvelut tai ohjelmistokehitys. Palvelut voivat luoda luotettavia rahavirtoja pienelläkin määrällä asiakassuhteita. Modulaarisuus haluaa, että järjestelmät ovat joustavia ja helposti muokattavissa uusien vaatimusten mukaisesti. Näiden toiminnan suunnitteluperiaatteita noudattamalla Industry 4.0 järjestelmät pystyvät tehokkaammin ja nopeammin reagoimaan muutoksiin markkinoissa. (Gilchrist 2016. Chapter 13)

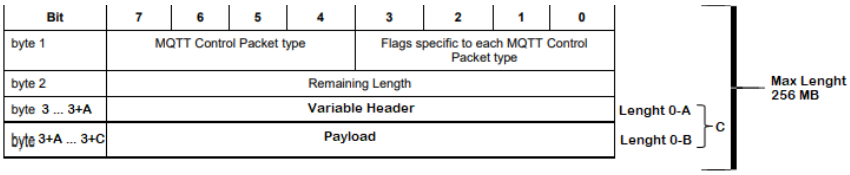
Glichristin (2016, 207–210) mukaan Industry 4.0 kehitys on sidonnainen tiettyihin teknologian aloihin. Näihin kuuluu robotiikka ja IoT valmiit laitteet, uudet valmistusmenetelmät, kuten 3D-tulostaminen, Big data eli datankeruu, data-analytiikka sekä pilvipalvelut. Robotiikan ja IoT valmiiden laitteiden kehitys mahdollistaa useampien laitteiden liittymismahdollisuuden osaksi industry 4.0 tietoverkkoja. Viestintäprotokollien kehittyessä ja niiden standardisoinnin tason kasvaessa laitteiden yhteistyö helpottuu. Big data kattaa kaiken datan keräämiseen ja käsittelemiseen käytettävät teknologiat. Kerättyä ja käsiteltyä dataa voidaan käyttää moniin käyt-

tötarkoituksiin kuten huoltojen suunnitteluun tai prosessin kehittämiseen. Digitalisaation mukana tulee kuitenkin uusia uhkia, ja näihin vastataan kehittämällä kyberturvallisuusteknologioita, joiden avulla pidetään tietokannat ja tietoverkot turvallisina ja luotettavina.

2.2 MQTT-viestintäprotokolla

MQTT on laitteiden välinen viestintäprotokolla, jonka toiminta perustuu Julkaisu/Tilaus periaatteeseen (Publish/Subscribe). Tämä tarkoittaa kommunikaation tapahtuvan viestien sisältyvän Topic-nimisen lisätiedon mukaisesti. Kaikki MQTT-protokollan avulla julkaistut laitteiden väliset viestit sisältävät tämän Topic-tiedon, ja serverille saapuvat viesti ohjataan niille asiakaslaitteille (Client), jotka ovat tilanneet tämän kyseisen Topic:n. MQTT-protokollassa serveri, toiselta nimeltään Broker, hallinnoi asiakkaiden Topic-tilauksien organisointia ja toimii viestien välittäjänä asiakaslaitteiden välillä. (Banks & Gupta 2020, 1–2, 9–10)

MQTT-protokolla on rakenteeltaan kevyt: viestipaketteihin sisältyvä protokollan toiminnalta pakollinen tieto on vähäistä, minimissään 2 tavua. Viestin maksimikoko on 256 MB. Tämä mahdollistaa sen käyttöä prosessointiteholisesti rajoitetuissa ympäristöissä. Protokollassa pakettien viestityypit määräytyvät niiden MQTT control packet -tyyppi ja Flag-tyyppi bittien perusteella. Viestityypit ovat ennalta määritelty ja niitä on 14. Nämä viestityypit määrittelevät paketin rakenteen ja jokaisella tyypillä on oma käyttötarkoitus. Viestityyppien control packet listaus löytyy liitteestä 1 ja viestityyppien ominaiset Flag-bitit löytyvät liitteestä 2. Viestipaketin rakenne esitellään kuvassa 1. (Banks & Gupta 2020, 1–2, 16–22)



KUVA 1. MQTT-viestipaketin rakenne. (Muokattu Banks & Gupta 2020, 16, muokattu)

Paketin pakollisissa osissa selviää paketin viestityyppi ensimmäisen tavun biteistä 7–4, viestikonfiguraatiot selviävät biteistä 3–0 ja toisessa tavussa on aina

paketin asiasisällön pituus. Loppu viestistä sisältää viestistä riippuen Variable header-osuuden, lisätietoa viestin toiminnasta (QoS > 0, Retain jne), ja/tai payload-osuuden eli itse laitteiden välisen viestinnän sisällön. Koska näitä tietoja ei tarvita viestissä, on protokolla siten täysin riippumaton kommunikaatioviestin sisällöstä. (Banks & Gupta 2020, 1–2, 16–22)

Protokolla sisältää toimintoja jotka mahdollistavat toiminnan heikoissa tietoverkollisissa ympäristöissä, laajentaen mahdollisia käyttökohteita. Toiminnot ovat Will message -viesti ja QoS-asetus. Will message on brokerille kerrottava viesti, joka lähetetään asiakkaan yhteyden katketessa ilman yhteyden katkaisemiskäskyn lähetystä. (Banks & Gupta 2020, 1–2, 26, 52–56)

QoS, Quality of Service, on viestipakettiyhteyden konfiguraatioasetus, jonka tarkoituksena on määritellä viestin vastaanottamisen varmistustapa. Asetusta on kolme eri tasoa ja se pätee sekä lähetykselle että tilaamiselle, toimintatavan ollessa molemmille samanlainen. Ainoa ero on se, että lähetyksessä Broker toimii vastaanottajana ja tilauksessa Broker toimii lähettäjänä. (Banks & Gupta 2020, 1–2, 26, 52–56)

Tasolla 0, ”At most once delivery”, ei ole mitään varmistusviestintää ja paketti saapuu vastaanottajalle, jos tietoverkko sen pystyy lähetys- tai tilaushetkellä kuljettamaan. (Banks & Gupta 2020, 1–2, 26, 52)

Tasolla 1, ”At least once delivery”, viesti saapuu vastaanottajalle ainakin kerran. Lähettäjä lähettää viestiä, kunnes vastaanottaja on lähettänyt vastaanottamisen varmistusviestin. Tästä johtuen viesti voi lähteä ja saapua monta kertaa, mutta saapuminen on varmistettua. (Banks & Gupta 2020, 1–2, 26, 53)

Tasolla 2, ”Exactly once delivery”, viestin lähettäminen varmennetaan tapahtuvan vain kerran, eli viesti saapuu varmasti ja ylimääräisten viestien lähetys estetään. Tämä saadaan aikaiseksi neljäosaisella viestinnällä, jossa kommunikaatioviestin lisäksi tapahtuu kolme varmistusviestiä. Vastaanottaja varmistaa saaneensa alkuperäisen viestin lähettäjälle, jolloin lähettäjä varmistaa vastaanottajalle saaneensa tämän vastaanottamisen varmistuksen, johon vastaanottaja vielä varmistaa saaneensa lähettäjän varmistuksen. (Banks & Gupta 2020, 1–2, 26, 54–55)

Vaikka MQTT voi toimia heikoissa tietoverkkoympäristöissä, sillä on tiettyjä vaatimuksia tietoverkolle, jotta protokollan toiminta on mahdollista. MQTT-protokollaa voidaan käyttää vain verkkoyhteysprotokollalla, joka takaa häviöttömän, viestisisällön järjestyksen säilymisen sekä molemmin suuntaisen viestinnän. Yleisesti käytössä on TCP/IP, mutta se ei ole ainoa mahdollinen protokolla. (Banks & Gupta 2020, 1–2)

MQTT-protokolla on standardisoitu mutta siitä on eri verisoita, jotka ovat eri organisaatioiden standardisoimia. MQTT on tällä hetkellä OASIS-organisaation OASIS MQTT Technical Committee -elimen hallinnoima ja uusin organisaation ratifioitu versio protokollasta on MQTT 5. Tässä työssä käytetään kuitenkin MQTT 3.1.1 versiota ja se on molempien OASIS - ja ISO/IEC 20922:2016 -standardien mukainen. Standardointi mahdollistaa protokollan olevan valmistajariippumaton ja kaikki samaan Brokeriin yhdistyneet laitteet voivat kommunikoida toistensa kanssa. (MQTT 2022)

2.3 IRB 140 ja DeviceNet

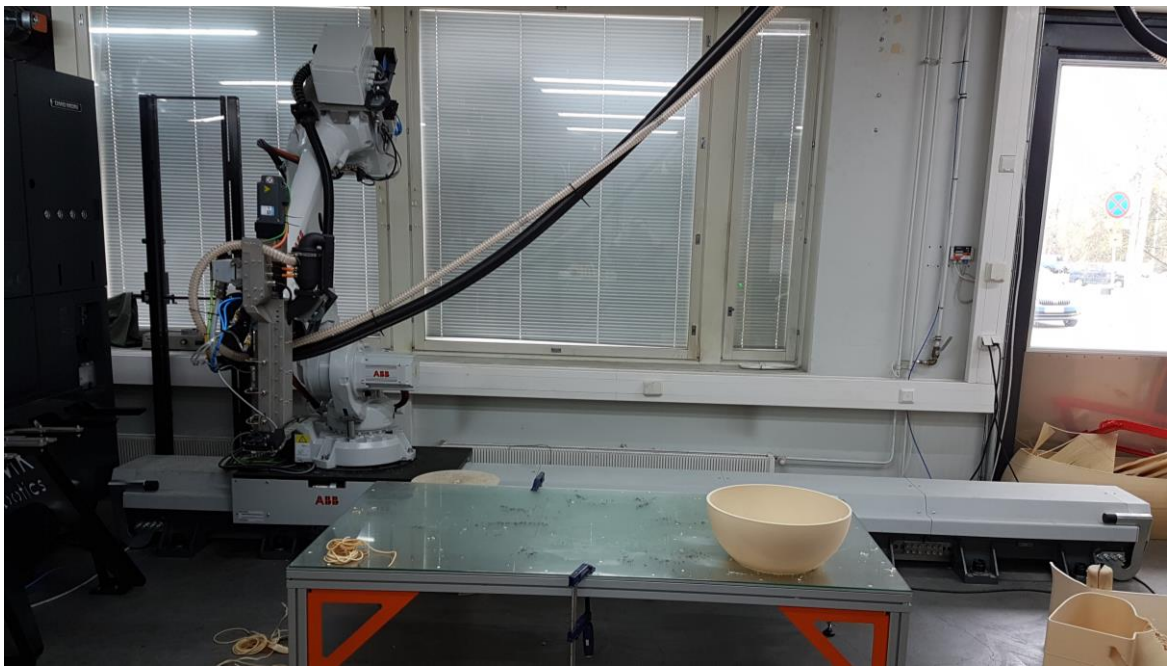
IRB 140 on ABB:n 6 akselinen teollisuusrobottimalli, joka on suunniteltu yleisiin teollisiin käyttötarkoituksiin. IRB 140 on pienikokoinen ja suunniteltu mahdollistamaan asennuksen missä tahansa asennossa laajentaen mahdollisia käyttötiloja. Robotin kantokapasiteetti on 6 kg ja kantama on 810 mm. Robottiohjain on malliltaan IRC5, jonka ohjausohjelman RobotWare käyttöversio on 5.15. Kaikkia ABB:n robotteja ohjelmoidaan ABB:n itse kehittämällä RAPID ohjelmointikielellä. TAMK:n robotti on opetuskäytössä asetettu liukuhihnan sisältävään robottisoluun (Kuva 2). (IRB 140 2021, 1–2)

KUVA 2. TAMK:n IRB 140 robottisolu

IRB 140:n I/O-kommunikaatio suoritetaan DeviceNet-moduulien avulla. Laitteiston konfiguraatio esitetään luvussa 3.1. DeviceNet on digitaalinen kenttäväylä-verkko, jota käytetään I/O-laitteiden ja ohjainlaitteiden väliseen kommunikaatioon. DeviceNet on ODVA-standardijärjestön kehittämä ja hallinnoima teknologia. Standardointi mahdollistaa valmistajariippumattomaa laitteiden välistä toimintaa. Kommunikaatio suoritetaan hyväksikäyttämällä toisia ODVA-järjestön standardoituja teknologioita, Controller Area Network CAN ja Common Industrial Protocol CIP. (DeviceNet® n.d)

2.4 IRB 4600 ja IoT Gateway

IRB 4600 on ABB:n 6 akselinen teollisuusrobottimalli, joka on suunniteltu monenlaisiin teollisiin käyttötarkoituksiin. IRB 4600 on suunniteltu viemään mahdollisimman pienen tilan ja samalla tarjoavan hyvin pitkän kantaman. TAMKin omistama versio robotista on IRB 4600-40/2.55. Version kantokapasiteetti on 40 kg ja kantama on 2.55 m. Robottiohjain on malliltaan IRC5, jonka ohjausohjelman RobotWare käyttöversio on 6.13. TAMK:n robotti on asennettu IRBT-2005 radalle ja siinä on CEAD 3D-tulostin lisäosan avulla 3D-tulostustarkoitukselliseen robottisoluun (Kuva 3). (IRB 4600 2021, 1–2)

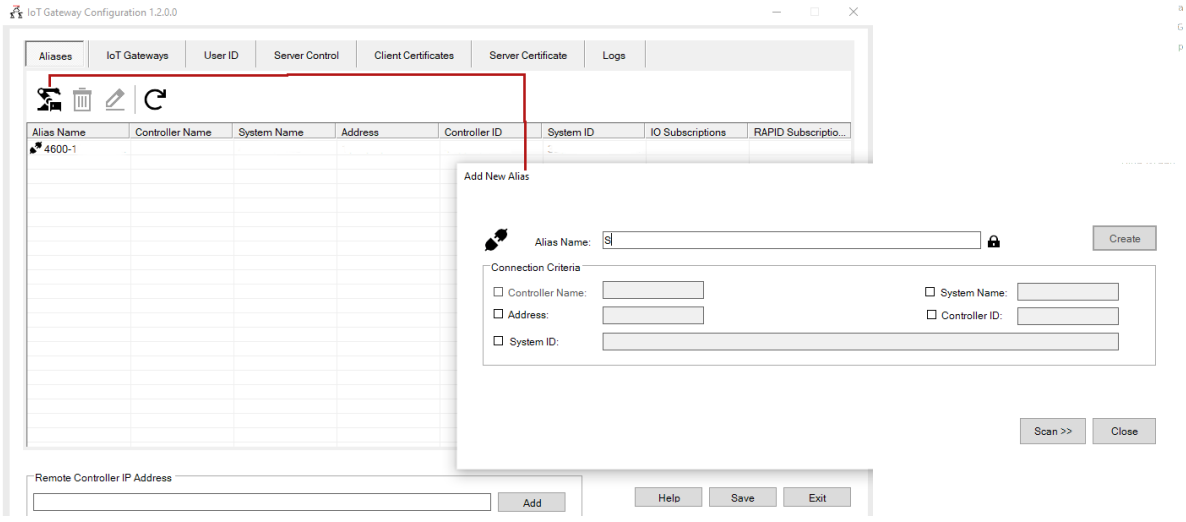


KUVA 3. TAMK:n IRB 4600 robottisolu.

Kommunikaatiojärjestelmän takia on robottiin hankittu vaadittavat lisenssit ABB:n IoT Gateway -sovelluksen käyttämiseen. IoT Gateway on ABB:n OmniCore- ja IRC5-robottiohjaimille suunniteltu laitteiden välisen kommunikaation mahdollistava ohjelma. IRC5-ohjaimet tarvitsevat RobotWare-version 6 tai korkeampi, jotta voivat käyttää IoT Gateway -ohjelmistoa. (Application manual - IoT Gateway 2022, 11)

Ohjelma on pääasiallisesti tarkoitettu OPC UA -pohjaiseen kommunikaatioon, mutta tarjoaa myös MQTT-lähetintoimintoja. Ohjelma ei tue MQTT-pohjaisen viestinnän vastaanottamista. Itse ohjelma toimii täysin tietokoneella ja vaatii Anniversary Updaten tai myöhemmän päivityksen 64-bittisen Windows 10 -käyttöjärjestelmän. (Application manual - IoT Gateway 2022, 11–12)

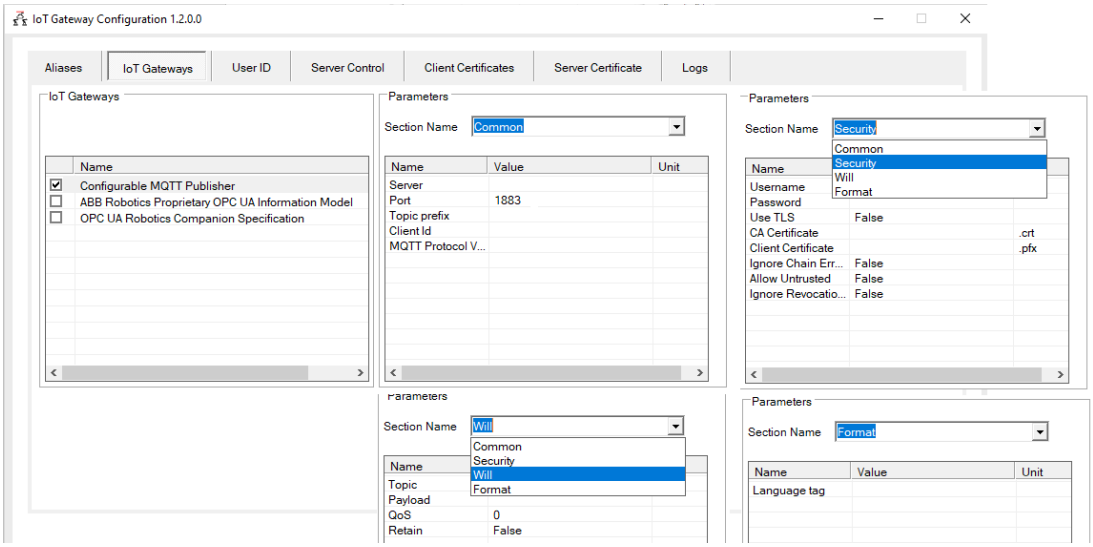
Ohjelman avulla voidaan yhdistää samassa tietoverkossa oleviin ABB:n robottiohjaimiin ja pystytään käsittelemään niissä sijaitsevaa dataa. Yhteys voidaan muodostaa langattomasti tai kytkeytymällä Ethernet-kaapeli robottiohjaimen Management-porttiin. Ohjaimen lisäämiseksi IoT gatewayn toimintaan sille luodaan Alias. Uusi Alias luodaan Add New Alias toiminnon avulla, joka on esitetty kuvassa 4. Aliakselle annetaan nimi ja halutut ohjaimen tiedot. Luotuun aliakseen otetaan automaattisesti yhteys, jos yhteys pystytään muodostamaan. (Application manual - IoT Gateway 2022, 11, 18-28)



KUVA 4. IoT Gateway -Aliaksen luontitoiminto.

IoT Gateway -toimintojen konfiguraatioasetuksia voidaan muuttaa käyttöliittymässä IoT Gateways -välilehden alla (Kuva 5) tai IoTGateway.config-tiedostosta. MQTT-viestinnälle ominaisia asetuksia ovat (Application manual - IoT Gateway 2022, 30–31)

- Common-Server: Yhdistettävän MQTT Brokerin IP-osoite
- Common-Port: Yhdistettävän MQTT Brokerin portti
- Common-Topic prefix: Kaikkien lähetettävien viestien topic-tiedon eteen lisättävä yhteinen etuosa. Mitään ei lisätä, jos kenttä on tyhjä.
- Security-Kaikki: Yhteyden vaatiessa joitain näitä tietoja on pakko tässä määritellä.



KUVA 5. Kaikki MQTT Publisher ominaisuuden konfiguraatioasetukset.

Client Certificates- ja Server Certificates -välilehdet liittyvät pelkästään OPC UA -kommunikaatioon. User ID -välilehti sisältää robottiohjaimelle kirjautumisen käytettävän käyttäjänimen ja salasanan. Kirjautumistiedot voidaan asettaa vain yleisesti, eikä Alias-kohtaisesti. Logs-välilehdessä esitetään käyttöliittymän sisällä ohjelman toiminnan aikana syntyneet log-tiedot. Nämä tiedot voidaan halutessaan tallentaa tekstitiedostoksi. Server Control -välilehdellä voidaan sammuttaa, tai uudelleen käynnistää ohjelman toiminta. Tämä pitää tehdä esimerkiksi muutoksien käyttöönottamisen takia. (Application manual - IoT Gateway 2022, 31–33, 39–44)

MQTT-viestintää hallinnoidaan C:\ProgramData\ABB\IoT Gateway\MQTT-polun sisälle tallennettavien tiedostojen avulla. Tiedostot jaetaan kolmeen ryhmään:

Controllers, Triggers ja Templates. Controllers:t ovat Aliaksiin sidottuja .xml-tiedostotyypin tiedostoja, jotka ohjaavat miten kyseisestä laitteesta lähetettävät viestit suoritetaan kutsumalla Trigger-tiedostoja. Triggers:t ovat .xml-tiedostotyypin tiedostoja, jotka määrittävät viestin lähetystapahtuman, viestisisällön eli millä Templates-tiedostolla ja QoS-asetuksella viesti lähetetään sekä minkä Topicin alla viesti lähetetään. (Application manual - IoT Gateway 202, 74–75, 76–77, 91–92)

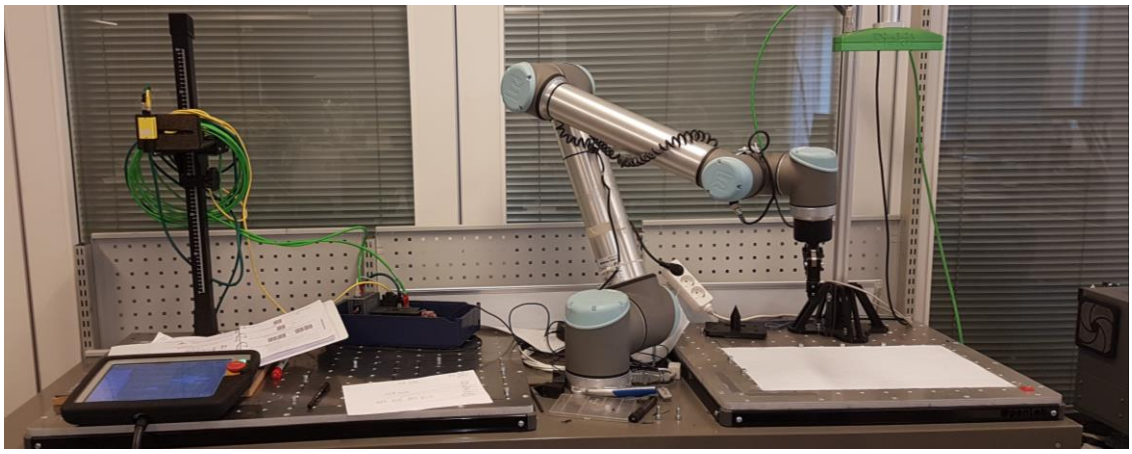
Tiedostoja voidaan muokata joko manuaalisesti halutulla tekstinmuokkausohjelmistolla tai ABB:n MQTT Engineering Tool -sovelluksella. Tämä sovellus asennetaan IoT Gatewayn mukana. Liitteissä 3 esitetään IoT Gateway -sovelluksen luomat esimerkkitiedostot jokaisesta tiedostotyypistä (Application manual - IoT Gateway 202, 74–75, 76–77, 91–92)

Trigger-tapahtumat ovat ennalta määritettyjä robottiohjaimen-tila tai muuttujapohjaisia tapahtumia, joita voidaan käyttää viestin lähettämisen käynnistykseen. Täysi lista Trigger-tapahtumia löytyy Application manual - IoT Gateway dokumentin sivuilla 79–88. Projektissa käytettävät Trigger-tapahtumat ovat Rapd-muuttujan muuttumiseen ja ajastimeen reagoivia. Rapid-muuttujista voidaan seurata vain Persistent-tyyppisiä (PERS) Rapid-muuttujia. Ajastin toimii ohjelmaa pyörittävällä tietokoneella, ja se voidaan määritellä 0,001 s tarkkuudella (Application manual - IoT Gateway 202, 76–90)

Template-tiedostoista rakennetaan MQTT-viestin asiasisältö käyttämällä Handlebars Template -kieltä. Handlebars-kielessä kaksinkertaisten aaltosulkeiden sisällä oleva teksti, {{esimerkkiteksti}}, käsitellään viittauksena johonkin ulkopuoliseen tekstinmukaiseen tietoon esimerkiksi {{Alias}} viittaa robotin IoT Gatewayssä määriteltyyn Alias-nimeen, {{Now}} viittaa ohjelman sisäiseen kellonaikaan ja {{Rapid.Tasks.T_ROB1.Modules.Welding.Vars.Cycle.Value}} viittaa robotin T_ROB1 taskissa olevan Welding-nimisen moduulin sisäiseen Persistent-muuttujan nimeltä Cylce tietosisältöön. Tietosisältö voi olla esimerkiksi numero tai merkkijono. Viesti saa tämän erikoistapauksen ulkopuolella olla vapaasti muotoiltu. Viesti voidaan silloin lähettää valmiiksi esimerkiksi XML-, JSON- tai Markdown-formaatissa. (Application manual - IoT Gateway 202, 91–117)

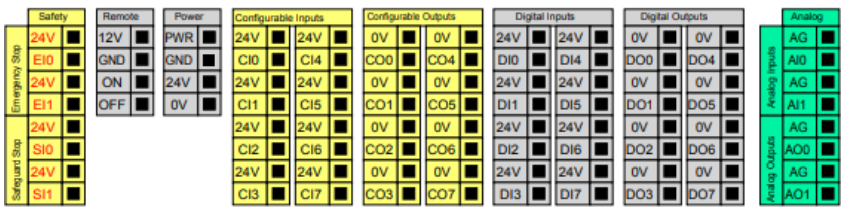
2.5 UR10

UR10 on Universal Robotsin 6-akselinen teollinen yhteistyörobotti. Yhteistyörobottina laite on suunniteltu ja sisältää toiminnot ihmisen kanssa samanaikaiseen työskentelemiseen. Toimintoihin kuuluu esimerkiksi törmäystarkastelu ja alemmat voimat robotilla. Robotin kantokapasiteetti on 10 kg ja kantama on 1.3 m. UR10 robotin ohjelmoiminen suoritetaan UR script -ohjelmointikielellä. TAMK:n UR10 sijaitsee avoimessa kahdella eri konenäköpisteellä varustetussa robottisolussa (KUVA 6). (UR10e n.d)



KUVA 6. TAMK:n UR10 robottisolu.

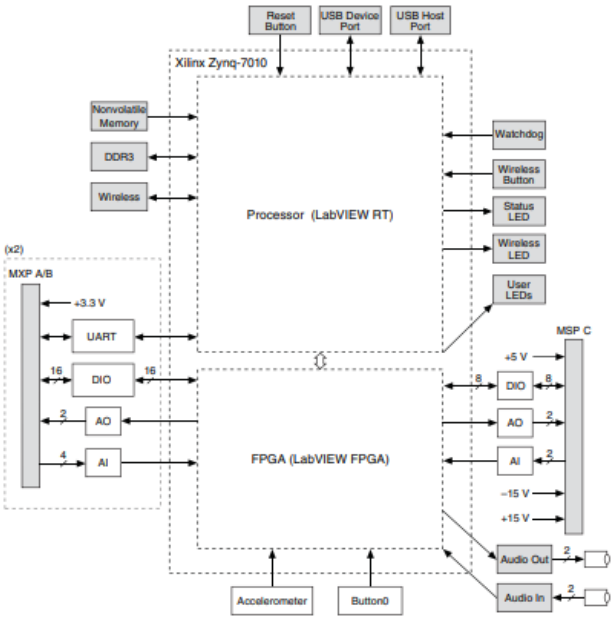
UR10 ohjainyksikkö sisältää sisäänrakennettuja I/O-liitäntöjä. Liitännät esitetään kuvassa 7 ja ne ovat vasemmalta oikealle: pysäytykselle omatut signaalit, laitteiston virtalähteet, ohjelmoitava DI/O (voidaan käyttää turvallisuuslogiikassa), yleisen käytön DI/O ja yleisen käytön AI/O. Yhteensä signaaleja on 16 DO, 16 DI, 2 AO ja 2 AI. Digitaaliulostulot tuottavat 24 V signaalin. DI-signaalit sammuvat alle 5 V jännitteellä ja käynnistyvät yli 11 V jännitteellä. AI/O toimii joko 0 V … 10 V jännitealueella tai 4 mA … 20 mA virta-alueella. Jos käytetään ohjaimen sisäisistä 24 V voimalähdettä, pitää silloin Power-kytkennän alla olevat PWR ja 24V sekä GND ja 0V yhdistää toisiinsa. (User manual UR10/CB3 n.d, 31–38)



KUVA 7. UR10 robottiohjaimen I/O liitännät. (User manual UR10/CB3 n.d, 31)

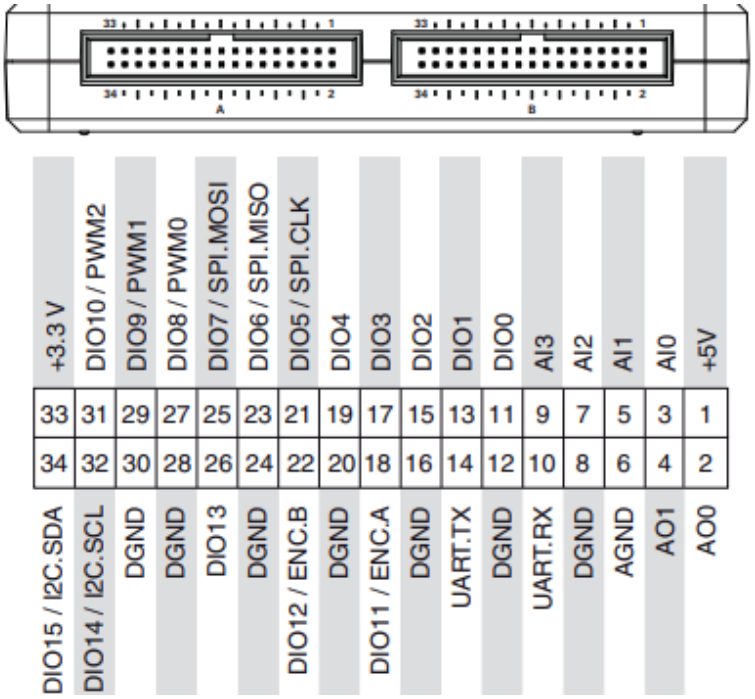
2.6 NI myRIO-1900

NI myRIO-1900 on National Instruments -yrityksen RIO-laiteperheen jäsen. Laite on opetuskäytön näkökulmasta suunniteltu käyttämään National Instrumentsin RIO-arkkitehtuuria, jossa yhdistetään prosessori, FPGA ja I/O yhdeksi LabVIEW-ohjelmointikielellä ohjelmoitavaksi kokonaisuudeksi. FPGA on ohjelmoitava puolijohdematriisi, joka käsittelee I/O-dataa, prosessori käsittelee ohjelmiston. Muillakin ohjelmointikielillä on mahdollista suorittaa ohjelmointia, mutta projektissa käytetään vain labVIEW-kieltä. Kuvassa 8 on esitetty laitteen komponenttien lohkokaavio. (National Instruments 2022)

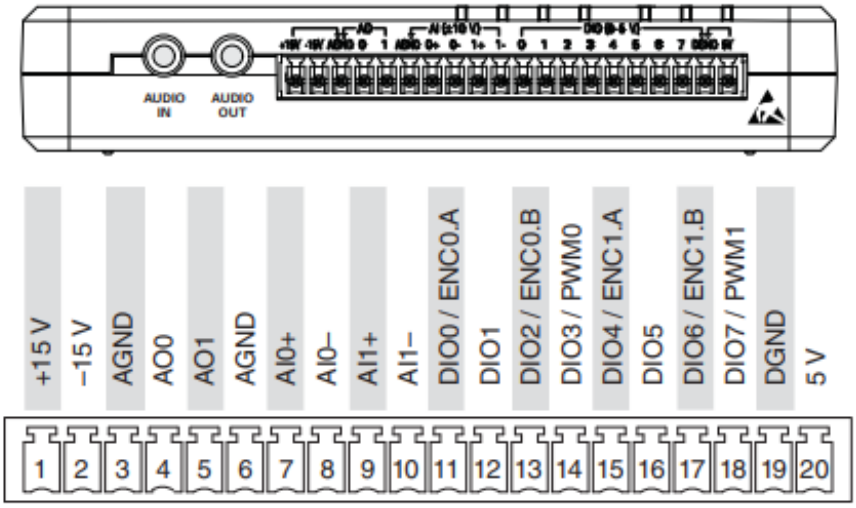


KUVA 8. NI myRIO-1900 komponenttien lohkokaavio (User guide and specifications NI myRIO-1900 n.d, 3)

NI myRIO-1900 -laite on rakennettu Xilinx Zynq-7010 -järjestelmäpiirin ympärille. Piiri sisältää kaksiytimisen ARM Cortex-A9 -prosessorin ja Artix-7 FPGA:n. I/O-laitteisto koostuu kahdesta MXP Expansion Portista (Kuva 9), yhdestä MSP Mini System Portista (Kuva 10), yhdestä painikkeesta ja audion sisään- ja ulostulokytkimestä. Yhteys laitteeseen saadaan muodostettua joko USB 2.0 -kytkennän avulla, tai langattomasti WiFi-yhteydellä. NI myRIO-1900 sisältää neljä kappaletta LED-valoja, joilla voidaan informoida käyttäjää virheistä, tai niitä voidaan käyttää ohjelmistollisesti määritettävissä diagnostisissa käyttötarkoituksissa. Lisäksi laite sisältää sisäänrakennetun kolmeakselisen kiihtyvyysanturin. (User guide and specifications NI myRIO-1900 n.d, 1–12)



KUVA 9. NI myRIO-1900 MXP -portin pinnien konfiguraatio (User guide and specifications NI myRIO-1900 n.d, 4)



KUVA 10. NI myRIO-1900 MSP -portin pinnien konfiguraatio (User guide and specifications NI myRIO-1900 n.d, 6)

DI/O-kytkentöjä on 40 kappaletta ja ne ovat porttityyppien välillä toiminnaltaan identtisiä. Kaikki kytkennät voivat toimia molempina sisääntuloina sekä ulostuloina. Ulostulosignaali on 3.3 V ja sisääntulosignaali toimii 3.3/5 V signaaleilla. Jotkin DIO-pinnit voivat toimia myös PWM käyttötarkoituksiin, tarkat pinnit näkyvät aikaisemmissa kuvissa. Kytkentöjen piirikaaviot löytyvät liitteestä 4 (User guide and specifications NI myRIO-1900 n.d, 1–12)

Analogisignaalit ovat jaoteltu erikseen ulostulo- ja lähtöpinneihin, lisäksi signaalin jännitealue vaihtelee MXP- ja MSP-porttien välillä. MXP-portin analogisignaalit ovat 0–5 V jännitealueella. Porttia kohden analogiulostuloja on 2 ja -sisääntuloja 4. MSP-portin analogisignaalit ovat ± 10 V jännitealueella. Sisääntulosignaalit ovat differentiaalisia. Portilla on analogiulostuloja 2 ja -sisääntuloja 2. Audio kytkennät suoritetaan AUX-liittimillä. Kytkentöjen piirikaaviot löytyvät liitteestä 5. (User guide and specifications NI myRIO-1900 n.d, 1–12)

NI myRIO-1900 -laitteessa käytetyn analogidigitaalimuuntimen resoluutio on $2^{12} = 4096$. Tämä mahdollistaa myRIO:n esittävän halutun jännitealue 4096 arvolla. Tämä vaikuttaa käsiteltävien lukujen tarkkuuteen ja myös käsiteltävien lukujen suuruuteen. Saadaan pienin jännitearvoväli kaavan 1 avulla. Kaavan jälkeen on esitetty pienin jännitteen arvo molempien AIO-porttien jännitealueille. V_{min} on pienin mahdollinen muutoksen aiheuttava jännite ja V_{alue} on jännitealueen amplitudi. (User guide and specifications NI myRIO-1900 n.d, 10)

$$V_{min} = \frac{V_{alue}}{2^{12}} \tag{1}$$

$$MXP\ V_{min} = \frac{5\ V}{2^{12}} = 1.221\ mV$$
$$MSP\ V_{min} = \frac{20\ V}{2^{12}} = 4.883\ mV$$

MXP-porttien pienin huomattava jännitemuutos on 1.22 mV ja MSP-portin pienin huomattava muutos on 4.88 mV. (User guide and specifications NI myRIO-1900 n.d, 10)

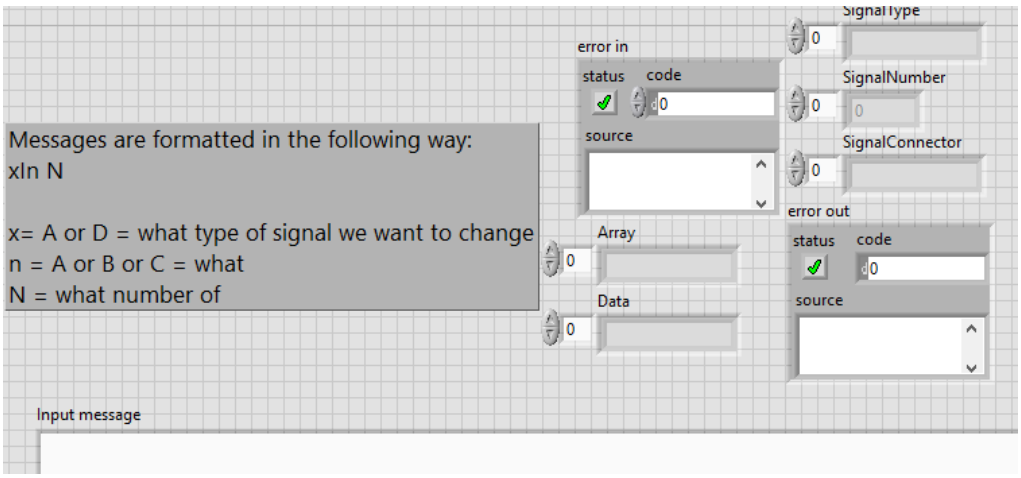
2.7 LabVIEW

LabVIEW on graafinen ohjelmointiympäristö, jossa käytetään G-ohjelmointikieltä. LabVIEW-ohjelmointiympäristö koostuu kahdesta osasta: Front Panel ja Block Diagram. Graafisena ohjelmointikielenä paljon informaatiosta ohjelmoidessa tulee visuaalisesti. LabVIEW-muuttujat ovat kaikki valmiiksi värikoodattuja. Taulukossa 1 esitetään nämä värit. (LabVIEW Environment Basics n.d)

TAULUKKO 1. LabVIEW Block Diagram -muuttujatyyppien värit. (Meaning of Different Wire Colors in LabVIEW 2019)

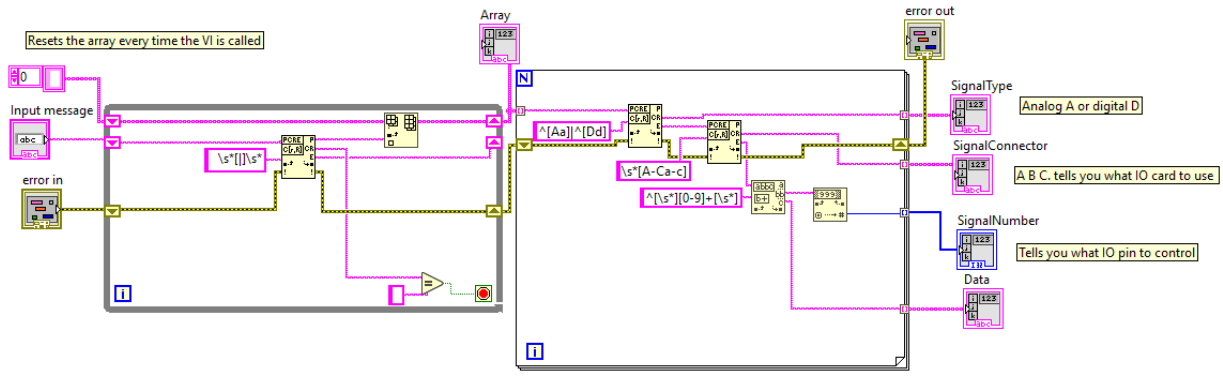
Muuttujan tyyppi	Yksittäinen arvo (Scalar)	1D taulukko (1D Array)	2D taulukko (2D Array)	Klusteri (Cluster)
Liukuluku (Floating Point)				
Kokonaisluku (Integer)				
Boolean				
Merkkijono (String)				
Polku (Path)				
Referenssi (Reference)				
Laitteistoresurssi (Hardware recource)				
Geneerinen data (Variant)				
Aaltomuoto (Waveform)				
Luokka (Class)				

Front Panel on alue, jossa käyttäjä voi manipuloida tai tarkastella ohjelman sisällä olevaa dataa enne ohjelman käyttöä tai käytön aikana. Tehtäviä operaatioita ovat esimerkiksi seurata analogisignaalin aaltomuotoa, manuaalisesti vaihdella Boolean-muuttujia tai tarkastella ohjelman sisällä olevia tekstimuuttujia. Front Panel -osiossa ei ole itse ohjelmointia vain pelkästään ohjelman datan tarkkailua tai manipulointia. Front Panel -toiminnot perustuvat ohjelmaan asetettuihin Control - ja Indicator-funktioihin. Control-funktiot mahdollistavat arvojen manuaalisen muokkauksen Front Panelista ja Indicator-funktiot mahdollistavat arvojen esittämisen Front Panel -näkymässä. Kuvassa 11 esitetään esimerkki Front Paneelista. (LabVIEW Environment Basics n.d)



KUVA 11. LabVIEW Front Panel esimerkki, sisältää indikaattoreita ja kommentin.

Ohjelmointi suoritetaan Block Diagram -osiossa. Block Diagrammin sisällä käyttäjä luo ohjelman toiminnot valmiiden funktioiden avulla. LabVIEW-ohjelma koostuu muuttujista ja ohjelmistofunktionaalisista palikoista, VI:stä tai Virtual Instrumentseista. VI:t ottavat sisäänsä muuttujatietoa ja suorittavat niille määritellyt funktiot, kuten suorittaa kertolaskun, yhdistää kaksi String-muuttujaa tai lisää listaan arvon. Käynnissä olevaa ohjelmaa ei voida Block Diagrammin sisällä muokata. Kuviossa 1 esitetään projektissa käytettävä MQTT-viestisisällön tekstinkäsittelyyn käytettävän ohjelma Block Diagram. (LabVIEW Environment Basics n.d)

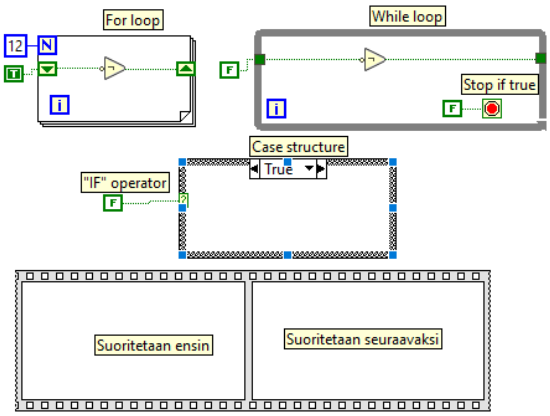


KUVIO 1. Tekstinkäsittelyohjelman Block Diagram.

LabVIEW, toisin kuin tekstipohjaiset ohjelmointikielet, ei seuraa tarkkaa ohjeiden järjestyshierarkiaa, vain ohjelmien toteutus perustuu dataflow-malliin. Dataflow-mallin mukaisesti ohjelman osat suoritetaan vasta sitten, kun niiden suoritukseen vaadittava tieto on saatavilla, kuten laskuoperaation kaikki vaadittavat tekijät. Tämä toteutustapa voi aiheuttaa ongelmia, jos kahdella tai enemmällä osalla oh-

jelmaa on samanaikaisesti kaikki tarvittava tieto käytettävissä. Tällaisissa tilanteissa ei voida tietää osien suoritusjärjestystä. Näissä tilanteissa joudutaan muilla metodeilla vaikuttamaan ohjelman toimintajärjestykseen, esimerkiksi lisäämällä ylimääräisiä järjestystä määrääviä tietovaatimuksia, kuten aikaisempien funktioiden error-tiedot. (Graphical Programming n.d)

LabVIEW-ohjelmien toimintaa voidaan vielä vaikuttaa Structure-tekijöiden avulla. Structure-tekijöitä ovat esimerkiksi While - ja For-loopit jotka ovat tekstiperäisten ohjelmointikieliä vastaavia toimintoja. If lauseita kuvataan Case Structureilla. Lisäksi on Flat Sequence Structure, joka siirtyy seuraavaan osaan vasta, kun kaikki aikaisemman osan toiminnot ovat suoritettu. Kuviossa 2 esitellään kaikki nämä structuret Block Diagrammissa. (Execution Structures in LabVIEW n.d)



KUVIO 2. LabVIEW Structureja.

Kuvion For- ja While-loopeissa näkyy myös datan käsittely loop-rakenteissa. Labview ei säilytä ohjelmakierrosten välillä dataa. Jos dataa halutaan säilyttää, tarvitaan Shift Registeriä. Kuvion For-loop Boolean-muuttuja on Shift Register muodossa ja While-loopin Boolean-muuttuja ei ole. Jokaisessa While-loopin kierroksessa muuttuja tulee Not-portille False-arvolla taas For-loopissa Not vaihtaa arvon jokaisella kierroksella.

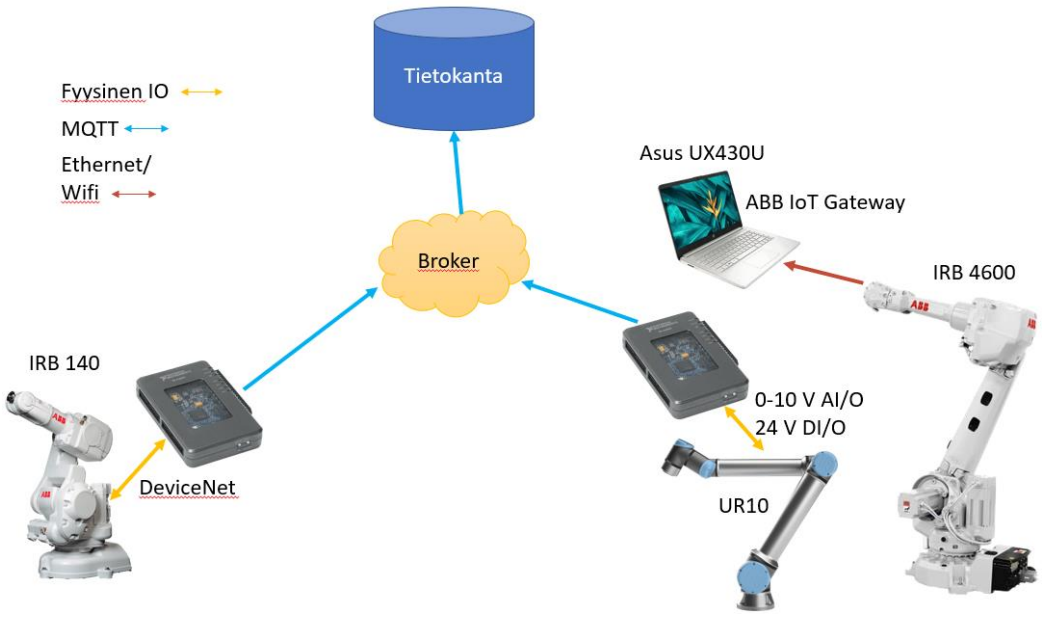
LabVIEW ei tue natiivisti MQTT-toimintoja. MQTT-toiminnallisuus on silti mahdollista toteuttaa kolmannen osapuolen ohjelmistolla. Projektissa käytetään vapaan lähdekoodin LVMQTT-kirjastoa (MQTT Protocol in LabVIEW 2020). LVMQTT-kirjosta tarjoaa kaikki MQTT-viestintään tarvittavat toiminnot valmiina funktioina. Kirjasto tukee MQTT 3.1.1 -versiota. (LVMQTT 2017)

3 KÄYTÄNNÖN TUTKIMUS

3.1 Alkutilanne

FieldLab-tilassa ei ole olemassa olevaa järjestelmää robottien väliselle kommunikaatiolle. Projektin järjestelmän kehittäminen ei kuitenkaan ala täysin tyhjältä pohjalta. Tilassa on käytössä aikaisemmin kehitetty MQTT-pohjainen datankeruujärjestelmä, jota tässä työssä laajennetaan kommunikaatiojärjestelmäksi.

Olemassa oleva datankeruujärjestelmä koostuu UR10- ja IRB 140 -robotteihin kytketyistä NI myRIO-1900 -laitteista, Eclipse Mosquitto MQTT Brokerista sekä SQL-tietokannasta. Mittausdata kuljetetaan TAMK:n sisäisessä TITE-verkossa sijaitsevalle virtuaalikoneella toimivalle Brokerille. Järjestelmässä viestien ainoa määränpää on ulkoinen SQL-tietokanta jota käytetään datan visualisointiin ja analysointiin. Tietokantaan kohdistuva MQTT-viestintä tapahtuu ”Flab1” ja ”Flab1_mm” topic-nimien alaisena. Mitään muita topic-nimiä ei ole brokerilla varattu. Järjestelmän osat ovat kykenemättömiä vastaanottamaan viestejä, eli järjestelmä on täysin yksisuuntainen tietokantaa kohti. IRB 4600 -robotti on varustettu tarpeellisilla lisensseillä IoT Gateway -ohjelman käyttämiseen mutta sitä ei ole kytketty datankeruujärjestelmään. Nykytilanteen topologia esitetään kuviossa 3.



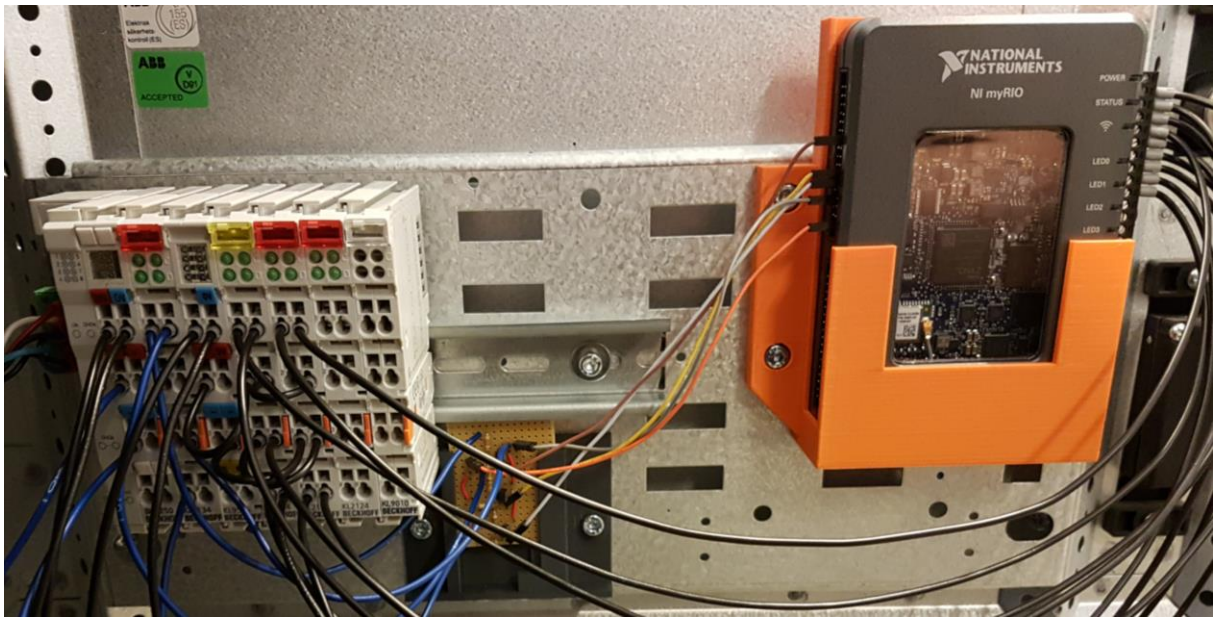
KUVIO 3. Projektin alkutilanteen topologinen kuvaus.

Olemassa oleva myRIO:lla toimiva datankeruuseen valmistettu LabVIEW-ohjelma on esiteltynä liitteessä 6. Ohjelman toiminta perustuu XML-tiedoston mukaisiin mittauskonfiguraatioasetuksiin. Asetuksissa kerrotaan mittaustapahtuman tarpeelliset tiedot kuten nimi, mittausdatan yksikkö, mitä myRIO I/O -portteja luetaan. Ohjelma sitten lähettää sekunnin välein asetuksen mukaisesti formatoidut I/O-tiedot JSON-formaatissa Brokerille.

NI myRIO-1900 on yhdistetty IRB 140:een beckhoff DeviceNet I/O -moduulien avulla. Käytössä on seuraavat moduulit:

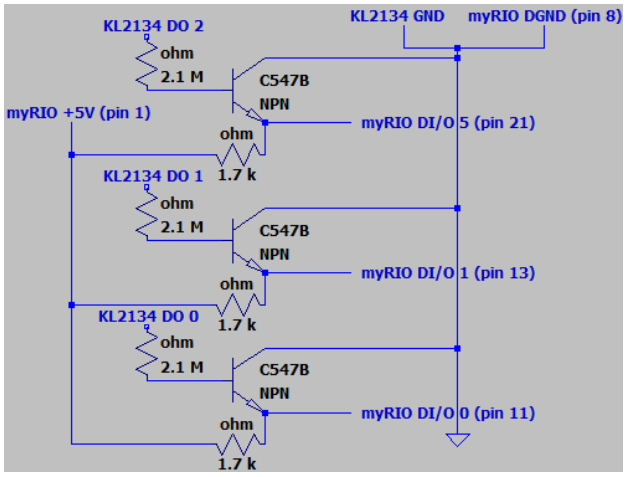
- 1x BK5250 (Bus coupler)
- 1x KL2134 (4x 24 V DC, DO)
- 1x KL9505 (24 V DC to 5 V DC, power supply)
- 1x KL1124 (4x 5 V DC, DI)
- 2x KL2124 (4x 5 V DC, DO)
- 1x KL9010 (Bus end terminal)

Tällä asetelmalla voidaan samanaikaisesti välittää IRB 140:stä lähtevän viestin sisällä parhaimmillaan 12 digitaalisignaalia. Vastaavasti IRB 140:een kohdistuvassa viestissä voidaan enintään jakaa 4 digitaalisignaalia. Fyysiset kytkennät nähdään kuvasta 12. Ylhäällä listatut moduulit ovat kuvassa samassa järjestyksessä vasemmalta oikealle.



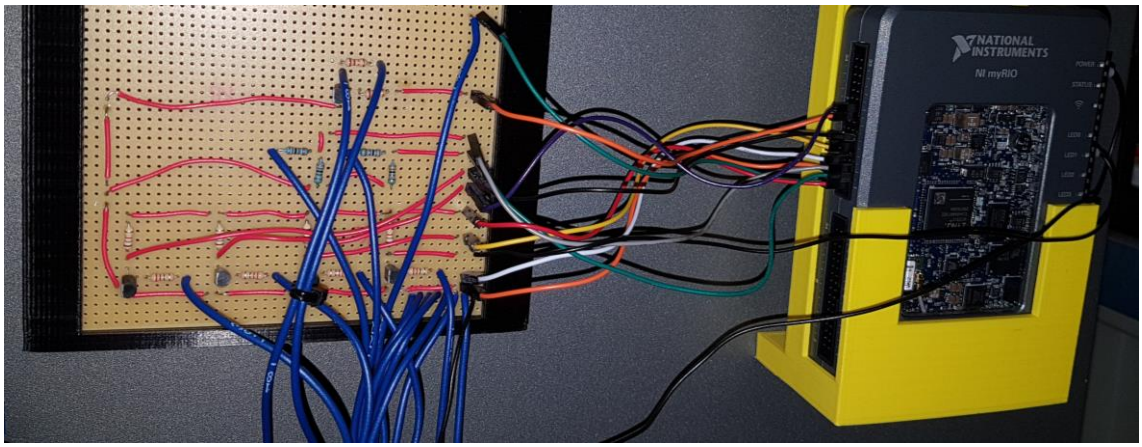
KUVA 12. IRB 140 kytkentä NI myRIO-1900 -laitteeseen

Yksi DeviceNet DO -korteista muodostaa 24 V signaaleja, jotka ovat yhteensopimattomia NI myRIO-1900 DI/O -kytkentöjen kanssa. Jotta laitteet voidaan kytkeä toisiinsa, tarvitaan I/O-kortin ja myRIO:n väliin välikytkentöjä. Datankeruujärjestelmää varten on aikaisemmin valmistettu tähän tarkoitukseen omatekoinen piirilevy. Kuviossa 4 on esitetty piirilevyn LTspice XVII -ohjelmalla piirretty piirikaavio.

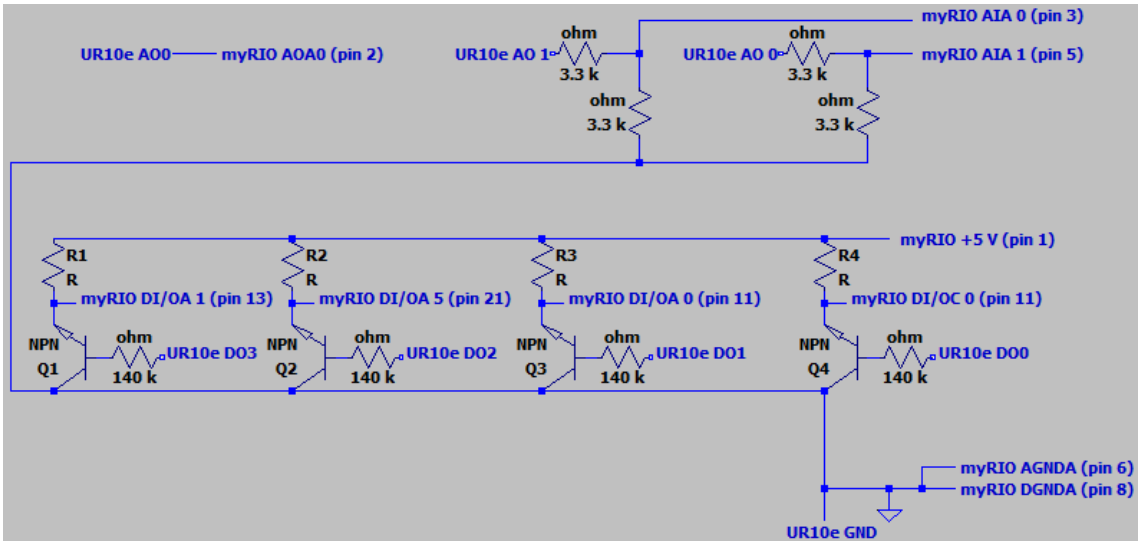


KUVIO 4. KL2134 ja NI myRIO-1900 välisen piirilevyn piirikaavio.

UR10 kanssa NI myRIO-1900 on yhdistetty UR10-ohjainyksikön natiiveihin I/O-kytkentöihin. Jotta UR10 natiivi I/O voidaan yhdistää NI myRIO-1900-laitteeseen, tarvitaan taas jännitteen alentamista tai muita tapoja ohjata signaaleja. Datankeruujärjestelmään on aikaisemmin valmistettu omatekoinen piirilevy, joka mahdollistaa kaiken, paitsi digitaalisignaalien lähettämisen NI myRIO-1900:lta UR10:lle. Kuvassa 13 on esitetty myRIO:n ja piirilevyyn väliset fyysisset kytkennät ja kuviossa 5 on esitetty piirilevyn piirikaavio.



KUVA 13. MyRIO:n ja UR10:n välinen kytkentä piirilevyllä.



KUVIO 5. Piirilevyn piirikaavio.

3.2 Laitteiston toiminnallinen selvitys

Toiminnallisuuden selvittämisen suoritettavien mittausten kohteina ovat LVMQTT-kirjasto ja IoT Gateway -sovellus. Kommunikaatioprotokollien ja -laitteiden arvioiminen on hyvin vaikeaa, johtuen niiden toimintaan vaikuttavien muuttujien määristä. Määrittämisen vaikeudesta johtuen käytetään mittauksien päättämisen auttavana aineistona artikkeleja ”A Performance Analysis of Internet of Things Networking Protocols: Evaluating MQTT, CoAP, OPC UA” sekä ” Pub/Sub: A Google-Scale Messaging Service”. Artikkeleissa nostetaan kommunikaatioverkkojen tärkeiksi tekijöiksi packet loss, TTC, topologia sekä viestien lähetettävä määrä tai viestien sisällön koko.

Artikkelit keskittyvät kuitenkin enemmän protokollatason arviointiin eikä yksittäisten client-laitteiden toimintaan. Tästä johtuen mittauksissa voidaan pohjautua artikkeleihin, mutta joudutaan soveltamaan mitattavia suureita protokollatasolta laitetasolle esimerkiksi nopeustarkasteluissa tutkitaan laitteen toiminnan nopeutta eikä viestin kuljetusnopeutta. Näitä seikkoja huomioiden toiminnallisuus näissä mittauksissa on asetettu tarkoittamaan laitteiston viestin lähettämisen/vastaanottamisen suorittamisnopeutta sekä luotettavuutta. Luotettavuudessa tarkastellaan sitä, kuinka hyvin laitteet reagoivat viestinlähetystapahtumiin ja kuinka luotettavasti viestit saapuvat Brokerille.

Aikataulusyistä toiminnallisuuden testaamiseen kohdistetaan laitteiden/ohjelmiston ideaaliseen toimintaan. Tämän tutkinnan avulla saadaan selville laitteiston paras mahdollinen toiminnan taso, jota voidaan käyttää aloituspisteenä selvittämään, millaisiin työtehtäviin ja toimintoihin laitteet sopivat käytettäväksi.

3.3 IoT-Gateway tutkimuksen selvityskohteet ja metodit

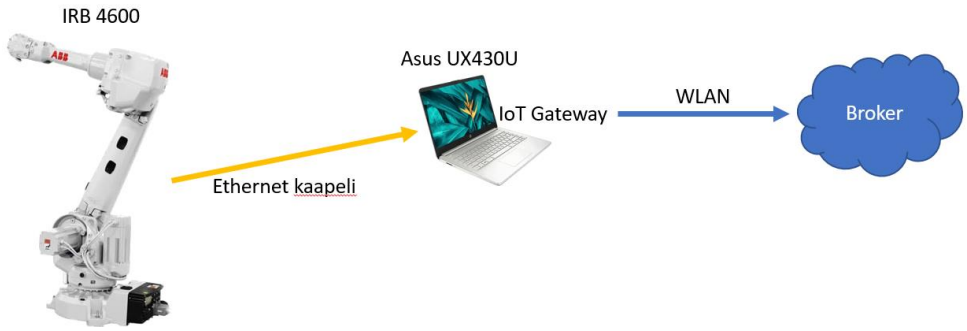
Kommunikaatiojärjestelmän näkökulmasta kiinnostavat viestinlähetysmetodit IoT-Gateway -ohjelmassa ovat ajastimella toimiva lähetys (Timer Trigger) sekä Rapid-muuttujaan reagoiva lähetys (RapidVariable Trigger). Viestintätapojen toiminnallisuutta tutkitaan yksittäisinä tapahtumina, jotta selvitetään lähetysmetodin toiminnallinen alaraja. Lisäksi tutkitaan toimivuutta monien samanaikaisten lähetyksien yhteydessä, jotta saadaan selville, miten lähetyksen voivat vaikuttaa toisiinsa ja miten päällekkäisten lähetyksien kanssa toimitaan.

Mittaustapahtuman aikana robotti asetetaan suorittamaan ohjelmaa ja lähetettävien MQTT-viestien sisällä kuljetetaan lähetystapauksen aikaleima, jota käytetään mittauksien suorittamiseen. Mittausdata kerätään etäyhteydessä MQTT Brokerilta PuTTY-terminaalin avulla. PuTTYn avulla luodaan MQTT-client, joka tilaa testiviestien Topicin ja viestien sisältö kopioidaan terminaalista. Rajoitteita tällä metodilla on se, että terminaalissa voi olla maksimissaan 2000 riviä tekstiä samanaikaisesti. Toinen tapa tutkia IoT Gateway -sovelluksen toimintaa on sovelluksen oman log-tiedoston avulla. Log kertoo, jos lähetystapahtuma Trigger-tapahtumaan ei reagoitu tai viestiä ei voitu lähettää esimerkiksi toisen viestilähetyksen takia. Tässä metodissa haittapuolena on se, että joudutaan luottamaan siihen, että Log-tiedosto on todenmukainen.

IoT Gateway -sovelluksen selvitettävät suureet ovat molemmille Trigger-tyypeille viestienlähetystiheys ja miten muuttujat kuten viestin koko vaikuttavat toimintaan. Lisäksi tutkitaan monien päällekkäisten Timer Trigger -tapahtumien interaktiota. Lähetystapahtumiin liittyvä mittausanalyysi suoritetaan selkeyden takia hertseinä, jos se on mahdollista.

Mittauksissa pyritän pitämään tutkittavan muuttujan ulkopuolella mittaustilanteen muut muuttujat mahdollisimman samanlaisina mittauksien välissä, jotta voidaan erotella yksittäisten muuttujien aiheuttamia vaikutuksia. Halutaan myös vähentää robotin toimintojen vaikutusta, jolloin robotilla suoritettavat ohjelmat ovat mahdollisimman yksinkertaisia ja prosessorille kevyitä.

Kaikki suoritettavat mittaukset tapahtuvat topologisesti identtisessä tilassa. Mittaustapahtumien topologia on esitetty kuviossa 6. IoT Gateway -sovellusta käytetään Asus UX430U -tietokoneella, jonka tekniset tiedot ovat taulukon 2 mukaiset.



KUVIO 6. IoT Gateway mittausten topologia

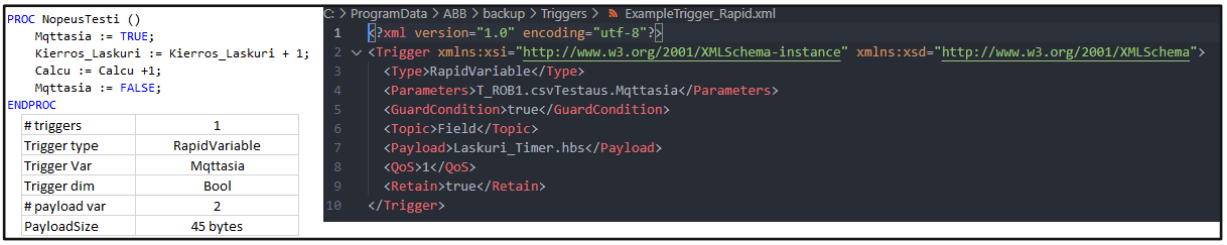
TAULUKKO 2. Asus UX430U tietokoneen tekniset tiedot.

Processor	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz
GPU	Inter(R) HD Graphics 620 4 GB
Installed RAM	8,00 GB (7,89 GB usable)
System type	64-bit operating system, x64-based processor

3.4 IoT-Gateway mittaustulokset

Ensiksi selvitetään IoT Gateway -sovelluksen RapidVariable Trigger -lähetystapauksen toimintanopeus. Kuvassa 14 esitetään robotinohjelma, IoT Gateway Trigger-tiedosto ja payload-tiedot taulukossa. Robottiohjelma laskee kahta laskuria ja vaihtelee Trigger-tapahtuman aiheuttavaa muuttujaa laskujen välissä. Näitä toimintoja suoritetaan niin nopeasti mitä prosessori pystyy. Payload sisältää kaksi

Handlebars-muuttujaa: {{Rapid.Tasks.T_ROB1.Modules.csvTestaus.Vars.mqtt-teesti.Value}},{{now}}. Lisäksi viestiin on lisätty ylimääräisiä plain text -kirjaimia, jotta 45-byte koko saavutetaan.



```
PROC NopeusTesti ()
    Mqttasia := TRUE;
    Kierros_Laskuri := Kierros_Laskuri + 1;
    Calcu := Calcu +1;
    Mqttasia := FALSE;
ENDPROC
```

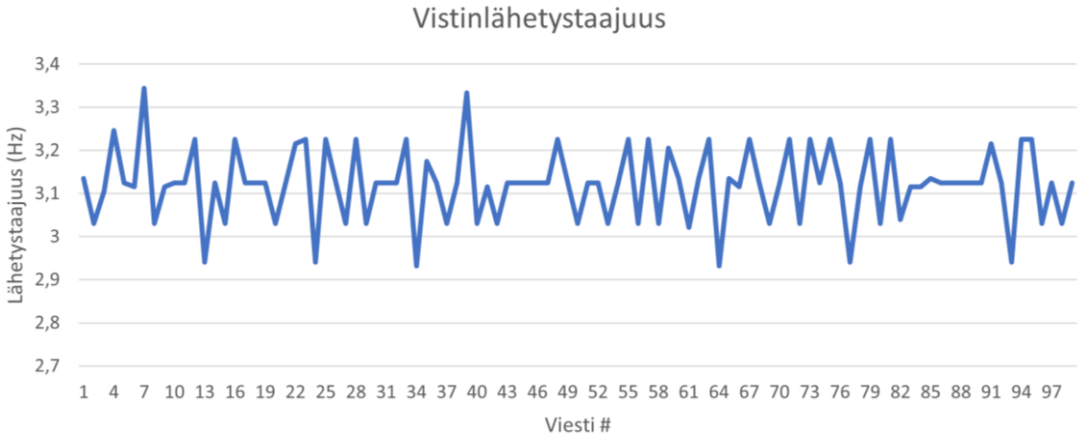
# triggers	1
Trigger type	RapidVariable
Trigger Var	Mqttasia
Trigger dim	Bool
# payload var	2
PayloadSize	45 bytes

C: > ProgramData > ABB > backup > Triggers > ExampleTrigger_Rapid.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <Type>RapidVariable</Type>
    <Parameters>T_ROB1.csvTestaus.Mqttasia</Parameters>
    <GuardCondition>true</GuardCondition>
    <Topic>Field</Topic>
    <Payload>Laskuri_Timer.hbs</Payload>
    <QoS>1</QoS>
    <Retain>true</Retain>
</Trigger>
```

KUVA 14. IoT Gateway RapidVariable Trigger mittauksen muuttujat.

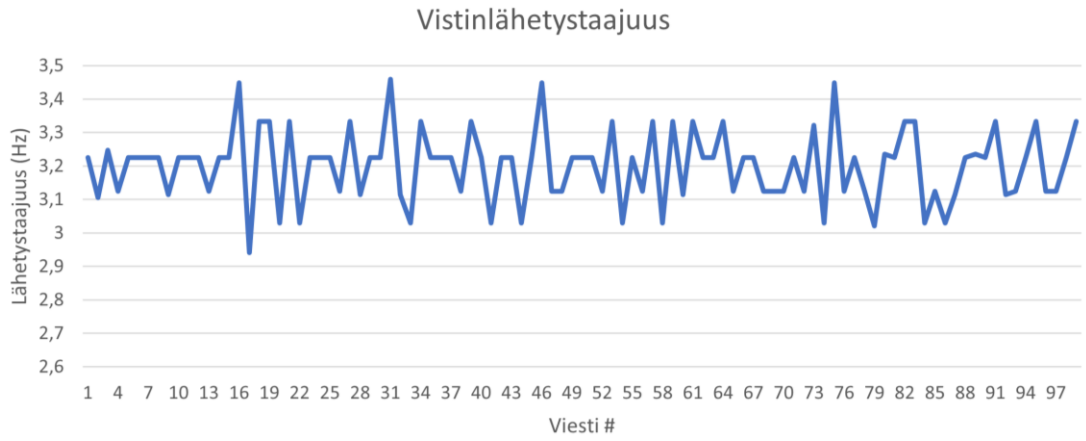
Mittauksia suoritettiin kymmenen kappaletta. Jokaisessa mittauksessa kerättiin sadan viestin aikaleimatieto. Kuviossa 7 esitetään yhden mittauksen lähetyksien väliseen aikaan perustuva lähetystiheys hertseinä.



KUVIO 7. Sadan IoT Gateway Mqttasia RapidVariable Trigger MQTT 45-byte viestin lähetystaajuus.

Kaikkien kymmenen mittauksen viestinlähetystaajuuden keskiarvo oli 3.125 Hz (viestinlähetysväli 0.320 s). Mittauksien viesteissä lyhyin viestinlähetysväli oli 0.290 s ja pisin 0.342 s.

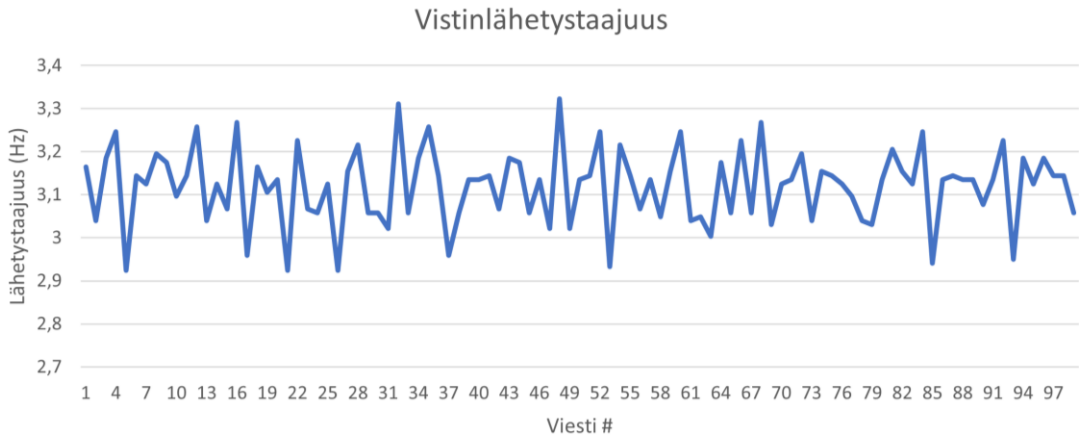
Halutaan selvittää, voiko tarkkailtavalla muuttujalla olla merkitystä lähetystapahtumaan. Suoritettiin uudet kymmenen samanlaista mittausta muuten samoilla asetuksilla, paitsi Trigger-muuttuja vaihdettiin boolean-muuttujasta ”Mqttasia” numeeriseen muuttujaan ”Kierros_Laskuri”. Tälle mittaukselle vastaava viestinlähetystaajuus kuvaaja esitetään kuviossa 8.



KUVIO 8. Sadan IoT Gateway Kierros_Laskuri RapidVariable Trigger MQTT 45-byte viestin lähetystaajuus.

Kaikkien kymmenen mittauksen viestinlähetystaajuuden keskiarvo oli 3.21 Hz (viestinlähetysväli 0.312 s). Mittauksien viesteissä lyhyin viestinlähetysväli oli 0.280 s ja pisin 0.340 s. Huomataan pientä eroa boolean-pohjaiseen Trigger-tapahtumaan mutta ei tarpeeksi huomattavaa eroa, jotta voidaan todeta numeerisen lähetyksen olevan parempi.

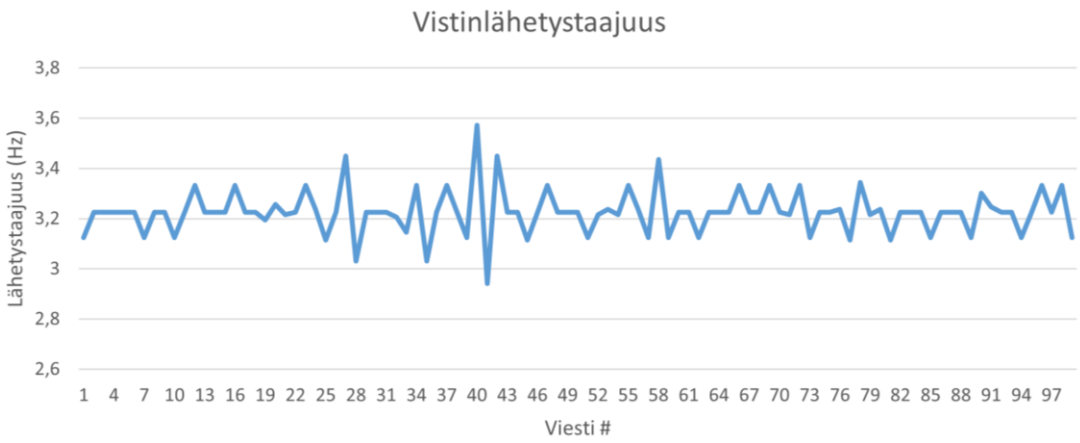
Halutaan myös selvittää viestin suuruuden vaikutusta viestinlähetykseen. Suoritetaan uudet kymmenen mittausta tälläkertaa käyttäen RapidTrigger-muuttujana taas ”Mqttasia”. Nostetaan Payload tiedoston plain text -kirjainten määrää, kunnes viestin koko on 450-byte. Tälle mittaukselle vastaava viestinlähetystaajuus kuvaaja esitetään kuviossa 9.



KUVIO 9. Sadan IoT Gateway Mqttasia RapidVariable Trigger MQTT 450-byte viestin lähetystaajuus.

Kaikkien kymmenen mittauksen viestinlähetystaajuuden keskiarvo oli 3.125 Hz (viestinlähetysväli 0.320 s). Mittauksien viesteissä lyhyin viestinlähetysväli oli 0.289 s ja pisin 0.350 s. Ei huomattavaa eroa 450-byte ja 45-byte kokoisten viestien välillä.

Viimeinen tutkittava RapidVariable Triggeriin vaikuttava muuttuja on tutkittavien Handlebars placeholderien määrä viestin tietosisällössä. Placeholderien määrä määrittää kuinka monta muuttujaa IoT Gateway hakee omasta- tai robotin muistista ja tämä voi hidastaa ohjelman toimintaa. Suoritetaan uudet kymmenen mittausta samoilla asetukilla, mutta muutetaan Payload-tiedostoa siten, että se sisältää 13 Handlebars-placeholderia ja viestin koko on väliltä 359–362 -Byte. Tälle mittaukselle aiempia vastaava viestinlähetystaajuus kuvaaja esitetään kuviossa 10.



KUVIO 10. Sadan IoT Gateway Mqttasia RapidVariable Trigger MQTT 13 placeholder viestin lähetystaajuus.

Kaikkien kymmenen mittauksen viestinlähetystaajuuden keskiarvo oli 3.215 Hz (0.311 s). Mittauksien viesteissä lyhyin viestinlähetysväli oli 0.280 s ja pisin 0.340 s. Kuviossa käytetty mittaus on tasaisempi, kuin aikaisemmat, mutta keskiarvollisesti samantasoinen.

Testien perusteella RapidVariable Trigger -toimintoon perustuvat viestinlähetys on tyhjiössä hyvin epätasaista peräkkäisten lähetysten välillä. Pidemmällä aikavälillä keskiarvo tasoittuu hyvin samanlaiseksi arvoksi, mutta ei ole mitään varmuutta peräkkäisten tapahtumien välisestä suoritusajasta. Tarvitaan ainakin 350

ms, luultavasti lähempänä 400 ms, viestinlähetystapahtumien välille, jotta voidaan alkaa luottamaan viestin lähettämiseen. Lisäksi vaikuttaa, että alle 1 MB tiedostokoissa viestin koko tai etsittävien muuttujien määrä ei erityisesti vaikuta ohjelman toimintaan.

Seuraava tutkimuksen kohde on Timer-perusteinen viestinlähetyksen toimintanopeus. Kuvassa 15 esitetään robotinohjelma, IoT Gateway Trigger-tiedosto ja payload-tiedot. Robottiohjelma odottaa asetetun ajan ja korottaa laskuria ja vaihtelee boolean-muuttujaa. Robottiohjelman sisäinen odotusaika on 0.1 s. Payload-tiedosto sisältää kolme Handlebars-muuttujaa: {{Rapid.Tasks.T_ROB1.Modules.csvTestaus.Vars.mqttteesti.Value}},Iot {{now}} ja {{Rapid.Tasks.T_ROB1.Modules.csvTestaus.Vars.Kierros_Laskuri.Value}}. Lisäksi viestiin oli lisätty ylimääräisiä plain text -kirjaimia, jotta saavutettaisiin vähintään 45-byte koko viestille.

```
PROC NopeusTesti_Timer ()
    WaitTime ajastin;
    Mqttasia := TRUE;
    Kierros_Laskuri := Kierros_Laskuri + 1;
    WaitTime ajastin;
    Mqttasia := FALSE;
ENDPROC
```

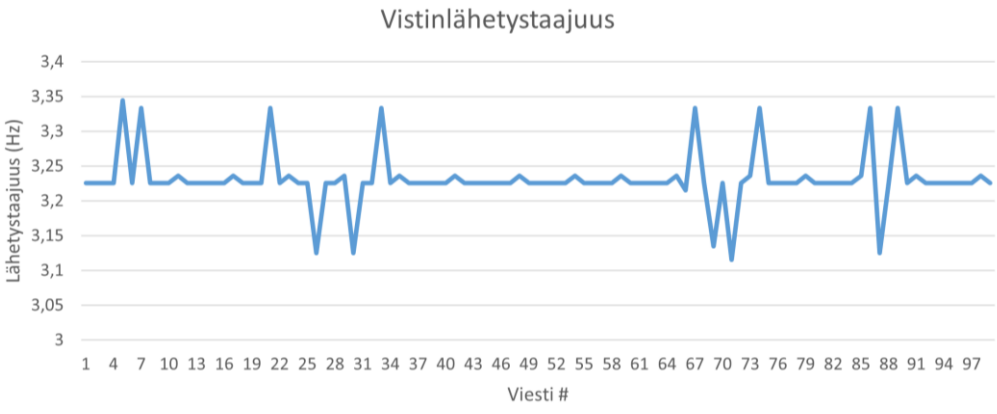
Trigger type	Timer
Trigger Var	0.3
Trigger dim	s
# payload var	3
PayloadSize	45 bytes
QoS	0
Retain	1

C: > ProgramData > ABB > backup > Triggers > ExampleTrigger_Timer.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<Type>Timer</Type>
<Parameters>0.3</Parameters>
<GuardCondition>true</GuardCondition>
<Topic>TestiTopic</Topic>
<Payload>Laskuri_Timer.hbs</Payload>
<QoS>0</QoS>
<Retain>true</Retain>
</Trigger>
```

KUVA 15. IoT Gateway Timer Trigger speed test muuttujat.

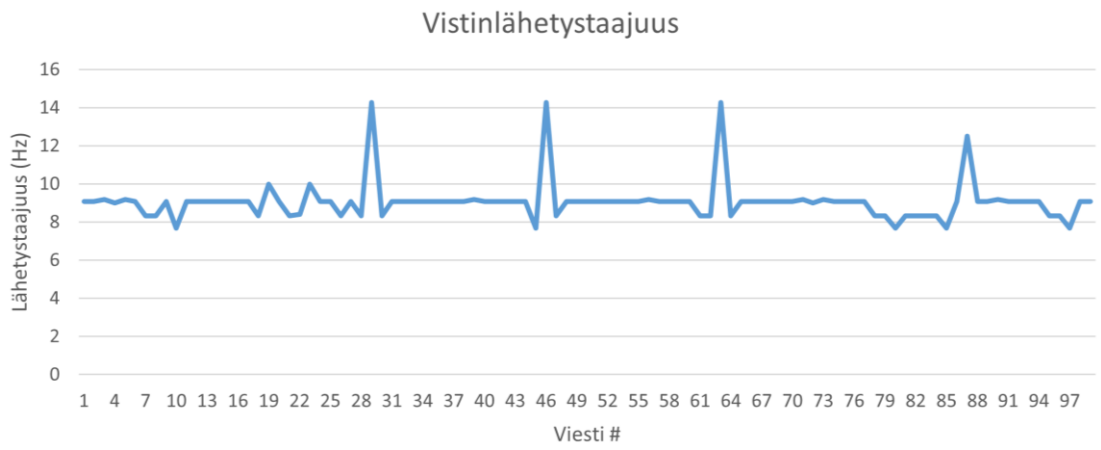
Mittauksessa kerättiin sata viestiä ja mittauksia suoritettiin kymmenen kappaletta. IoT Gateway asetettiin lähettämään viesti Timer Triggerillä RapidVariable Trigger mittaustapahtumien mukaisesti 0.3 s välein. Aloituspiste on valittu, koska RapdiVariable Trigger pystyi noin 0.3 s viestilähetystiheyteen. Kuviossa 11 on esitelty yhden mittauksen lähetystiheys Hz arvoilla.



KUVIO 11. Sadan IoT Gateway 0.3 s Timer Trigger MQTT viestin lähetystaajuus.

Kaikkien kymmenen mittauksen viestien viestinlähetystaajuuden keskiarvo oli 3.236 Hz (0.309 s). Mittauksien viesteissä lyhyin viestinlähetysväli oli 0.289 s ja pisin 0.330 s. Kuviossa käytetty mittaus on tasaisempi verrattuna RapidVariable Trigger -kuvioihin. Loputkin mittaukset ovat myös tasaisempia. Timer Trigger näyttää lisäävän Trigger-tiedostossa asetetun ajastimen pituuteen noin 10 ms.

Tutkitaan ajastinlähetyksen alarajaa laskemalla Trigger-tiedoston ajastinarvoksi 0.05 s. Muuten mittaus on identtinen aikaisempaan mittaukseen. Kuviossa 12 on esitelty yhden mittauksen lähetystiheys Hz arvoilla.



KUVIO 12. Sadan IoT Gateway 0.05 s Timer Trigger MQTT viestin lähetystaajuus.

Kaikkien kymmenen mittauksen viestien viestinlähetystaajuuden keskiarvo oli 8.85 Hz (0.113 s). Mittauksien viesteissä lyhyin viestinlähetysväli oli 0.06 s ja pisin 0.140 s. Tässä mittauksessa huomataan, että Timer pohjainen Trigger ei pysty luotettavasti yli 9 Hz lähetystaajuuteen. Kuvio on taas tasaisempi verrattuna RapidVariable Triggereihin. Testauksen perusteella pienin käytettävä Timer Trigger arvo on siis jossain noin 100 ms ympärillä.

On tähän mennessä selvitetty yksittäisten lähetystapahtumien toimintaa. Seuraavaksi tarkastellaan monien Trigger-tapahtumien samanaikaista suorittamista. Jotta saadaan parempaa kuvaa pitkältä aikaväliltä, annetaan mittausten kestää niin pitkään, että PuTTY-terminaalin 2000 rivin raja ylittyy reilusti. Tämän takia joudutaan luottamaan IoT Gateway log-tiedostoon toiminnan analysoimisessa. Voidaan PuTTYn avulla silti suorittaa tarkempaa tarkastelua valituista kohdista mittausta.

Mittauksissa halutaan selvittää, kuinka luotettavasti samanaikaiset Trigger-tapahtumat toimivat. Tämän takia mittauksien ajanjaksoista lasketaan jokaisen Trigger-tapahtuman esiintymismäärä ja sitä verrataan aikaisemman mittausdatan perusteella laskettuun odotettuun esiintymismäärään. PuTTY-mittauksesta saadaan datan avulla laskettua oikea esiintymismäärä, mutta koko mittauksen analysoinnissa joudutaan käyttämään Trigger-tapahtumille asetetusta tapahtumisaikavälistä laskettavalla odotetulla esiintymismäärällä. IoT Gateway Log-tiedosto kertoo meille mittauksen aikana suorittamatta jätettyjen Trigger-tapahtumien määrän, joita verrataan teoreettiseen laskelmaan. Viestien oletettu lähetysväli asetetaan olemaan 10 ms pidempi, kuin asetettu Timer Trigger -ajastimen arvo perustuen aikaisempiin mittaustuloksiin. Laskettu lähetettyjen viestien määrä saadaan jakamalla mittausväli oletetulla lähetysvälillä

Ensimmäinen monien Trigger-tapahtumien mittaus sisältää viisi Timer Trigger -tapahtumaa ja yhden RapidVariable Triggerin. Timer Triggerien -ajastimet ovat kymmenen millisekunnin välein välitä 0.09 s – 0.13 s. Robottiohjelma on taas laskuri ja Boolean-muuttujan vaihteleva ohjelma. Tässä mittauksessa robottiohjelmaan on lisätty ajastin, joka hidastaa ohjelman suorittamista 320 ms verran jokaisen kierroksen välissä. Tällä ajastimella RapidVariable Trigger mittaustuloksien pohjalta ollaan kyseisen Trigger-tapahtuman toiminnan äärirajoilla. Kuvassa 16 on esitetty mittaustapahtuman robottiohjelma ja IoT Gatewayllä käytetyt Controller- ja Trigger-tiedostot. Trigger-tiedostoista on esitetty kokonaan yksi ja muista vain ajastimen asetukset. MQTT-viestin sisältö on Kierros_Laskurin arvo, IoT Gateway timestamp ja Trigger-tapahtuman indikaattori, tässä tapauksessa kuvaava kirjain.

KUVA 16. Robottiohjelma ja IoT Gateway Timer Trigger MultiTrig mittauksen muuttujat.

Mittausta suoritettiin 35 min 03,476 s ajanjakson ajan. Tässä välillä otettiin yksi kahdentuhannen rivin tarkempi otanta PuTTYn avulla. Tarkempi otanta kattaa 46,141 sekunnin ajanjakson. PuTTY-mittauksessa saadaan taulukon 3 mukaiset tulokset.

TAULUKKO 3. 6 Trigger tapahtuman PuTTY-mittauksen tulokset.

Trigger	viestien oletettu Lähetysväli (s)	Mitattu lähetettyjen viestien määrä	Oletettu lähetettyjen viestien määrä	Puuttuvat viestit	Puuttuvien prosentti
A	00,110	414	419,4636364	5	1,2 %
B	00,100	450	461,41	11	2,4 %
C	00,120	379	384,5083333	5	1,3 %
D	00,130	350	354,9307692	4	1,1 %
E	00,140	321	329,5785714	8	2,4 %
R	00,320	144	144,190625	0	0 %

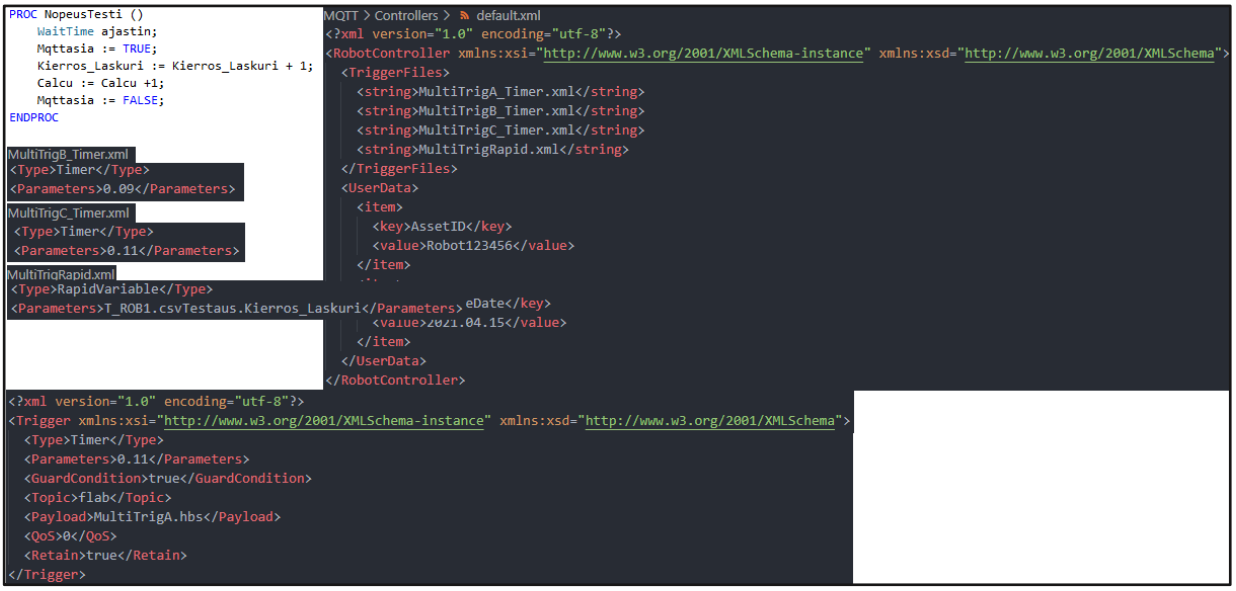
Mittauksesta nähdään näinkin lyhyellä aikavälillä jo kaikissa paitsi RapidVariable Trigger -tapahtumassa puuttuvia viestejä. Puuttuvat viestit eivät myöskään ole jakautuneet erityisen tasaisesti tai lähetysvälin mukaisesti. Eniten kadonneita viestejä on lyhyimmällä ajastimella, joka on aikaisempia mittauksia vastaavan alimman toimitarajan alapuolella. Tutkitaan vertailuksi koko mittauksen aikaväliä. Taulukossa 4 esitetään laskennan ja Log tiedoston tulokset.

TAULUKKO 4. 6 Trigger tapahtuman koko mittauksen tulokset.

Trigger	viestien oletettu Lähetysväli (s)	Oletettu lähetettyjen viestien määrä	Puuttuvien viestien määrä	Puuttuvien prosentti
A	00:00,110	19122,50909	90	0.47 %
B	00:00,100	21034,76	87	0.41 %
C	00:00,120	17528,96667	84	0.48 %
D	00:00,130	16180,58462	88	0.54 %
E	00:00,140	15024,82857	81	0.54 %
R	00:00,320	6573,3625	243	3.7 %

Odotetusti puuttuvien viestin määrä on kasvanut verrattuna PuTTY-mittaukseen. Huomataan taas Timer Trigger -lähetyksissä suhteellisen tasainen jakauma menetettyjä Trigger-tapahtumia ja ei taaskaan näytä olevan mitään suoraa linkkiä puuttuvien lähetysten määrän ja lähetysnopeuden välillä. Tässä pidemmässä mittauksessa poistuu kaikki epäilykset nopeimman ajastimen olleen erityisen huonosti toimiva. Suurin ero on kuitenkin erittäin huomattavasti suurempi määrä RapidVariable Trigger -tapahtumien menetyksiä verrattuna muihin ja siihen, että PuTTy-mittauksessa niitä oli 0. Oletus menetyksen määrälle on: RapidVariable Trigger on epätasaisempi toiminnaltaan ja tässä mittauksessa on tarkoituksellisesti laitettu äärirajoilla oleva tapahtumaväli Triggereiden välille. Tämä on yksinkertaisin selitys ilmiölle ilman jatkotestejä. Muuten viestit ovat identtisiä niin asetuksellisesti ei pitäisi olla muita eroja. Timer Triggerien toiminta on tasaisempaa, jolloin ne pysyvät varmemmin pois toistensa päältä.

Oletuksia tutkitaan toisella mittauksella, jossa lasketaan Timer Triggereiden määrää kolmeen ja nostetaan RapidVariable Triggerin QoS-tasolle 1. Lisäksi nostetaan Timer Triggerien -ajastimien välit 20 ms pitkiksi. Tässä mittauksessa Timer Triggerit ovat väliltä 0.11 s – 0.15 s. Robottiohjelma ja Timer Trigger -ajastin pidetään samana, jotta nähdään vaikuttaako QoS-muutos toimintaan. Kuvassa 17 on esitetty mittaustapahtuman robottiohjelma ja IoT Gatewayllä käytetyt Controller- ja Trigger-tiedostot. Mittauksen viestisisältö on myös muutettu kooltaan suuremmaksi ja muotoon, jossa PuTTY-mittaus tuottaisi vähemmän mittausdataa. Jätetään tässä PuTTY-mittaus suorittamatta ja analysoidaan vain



KUVA 17. Robottiohjelma ja IoT Gateway Timer Trigger MultiTrig2 mittauksen muuttujat.

Mittausta suoritettiin 58 min 28.999 s ajanjakson ajan. Mittauksessa saatiin taulukon 5 mukaiset tulokset.

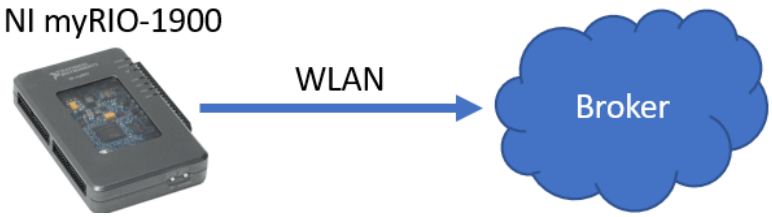
TAULUKKO 5. 4 Trigger tapahtuman koko mittauksen tulokset.

Trigger	viestien oletettu Lähetysväli (s)	Oletettu lähetettyjen viestien määrä	Puuttuvien viestien määrä	Puuttuvien prosentti
A	00:00,120	29241,65833	22168	75,8 %
B	00:00,140	25064,27857	17422	69,5 %
C	00:00,160	21931,24375	14211	64,8 %
R	00:00,320	10965,62188	294	2,7 %

Tässä mittauksessa huomataan erittäin suuri ero aikaisempaan mittaukseen molemmissa Trigger tyypeissä. RapidVariable Trigger -tapahtumia menetettiin hyvin lähelle sama määrä, kuin aikaisemmassa puolet lyhyemmässä mittauksessa. QoS 1 -asetus vaikuttaa tehostavan yksittäisen mittaustapahtuman suorittamista huomattavasti. Tämän havainnon kääntöpuolella on se, että Timer Trigger -tapahtumat kaikki menettivät vähintään noin kaksi kolmesta lähetystapahtumasta. Tämä muutos on erittäin suuri verrattuna aikaisempaan, jossa Timer Trigger -tapahtumat olivat kaikki alle prosentin tapahtumamenetyksissä. Lähetyksien QoS-asetukset ovat täten erittäin tärkeää tarkistaa, koska ne voivat aiheuttaa suurta vahinkoa muille lähetyksille.

3.5 NI myRIO-1900 tutkimuksen selvityskohteet ja metodit

NI myRIO-1900 pystyy suorittamaan molempia Publish- ja Subscribe-toimintoja LVMQTT-kirjaston avulla. Tässä vaiheessa tutkitaan näitä molempien toimintojen toiminnallisuutta yksittäisissä tilanteissa siten, että ohjelma toimii vain Publish- tai Subscribe-toimissa. Samanaikaista toimintaa arvioidaan pilotin yhteydessä. Tutkittavia suureita ovat viestienlähetys ja -vastaanottamistiheys. Tutkitaan myös mahdollista packet lossia Subscribe-toiminnallisuuden mittauksissa. Kaikki suoritettavat mittaukset tapahtuvat topologisesti identtisessä tilassa. Mittaustapahtumien topologia on esitetty kuviossa 13.



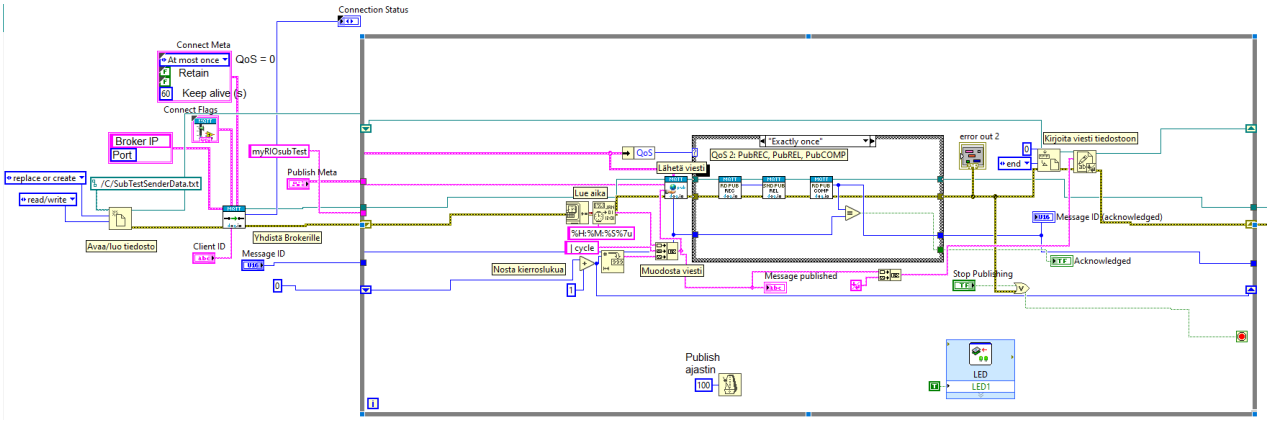
KUVIO 13. NI myRIO-1900 mittauksien topologia.

Mittaustapahtumissa ei kytketä tai lueta I/O-signaaleja, koska halutaan selvittää laitteiston ja kirjaston yleistä toimintaa. Käsiteltävät MQTT-viestit sisältävät niiden lähetys- ja vastaanottamisaikaleimat ja viestien sisältö kirjoitetaan NI myRIO-1900:n muistissa olevaan tekstitiedostoon lähetys- tai vastaanottamishetkellä. Viestien lähetysluotettavuutta tarkastellaan kierroslaskurilla, jota nostetaan jokaisella ohjelmakierroksella yhdellä.

Mittauksissa voi ilmetä vaikeasti havaittavaa virhettä siinä, että NI myRIO:n kello ei vaikuta olevan täysin luotettava. Sen laskenta voi mahdollisesti vaihdella ja hidastua laitteen toimintaan perustuen. Mittauksissa kuitenkin käsitellään aikaa kuten siinä ei olisi virheitä. Tämä johtuu siitä, että kellon virheet ovat erittäin vaikeita havaita/huomioida. Lisäksi laitteiden kellot pitää jokaisella uudelleenkäynnistyksellä asettaa uudelleen, jolloin laitteiden välillä on kellonajoissa eroa ja niiden kellot voivat toimia eri nopeuksilla eri hetkillä. Tämä voi vaikuttaa lähetys- ja tilaustapahtumien välisen desynkronisaation syntymiseen. Tämäkin virhe jätetään huomioimatta, koska aikaleimoja tutkittaessa oletetaan niiden toimivat täysin

yhdenmukaisesti. Aikojen vertaaminen toteutuu siten, että lasketaan aikaleimojen ero ensimmäisten käsiteltyjen viestin välillä ja erotuksen arvon avulla korjataan kaikkien viestien aikaleimojen erotus.

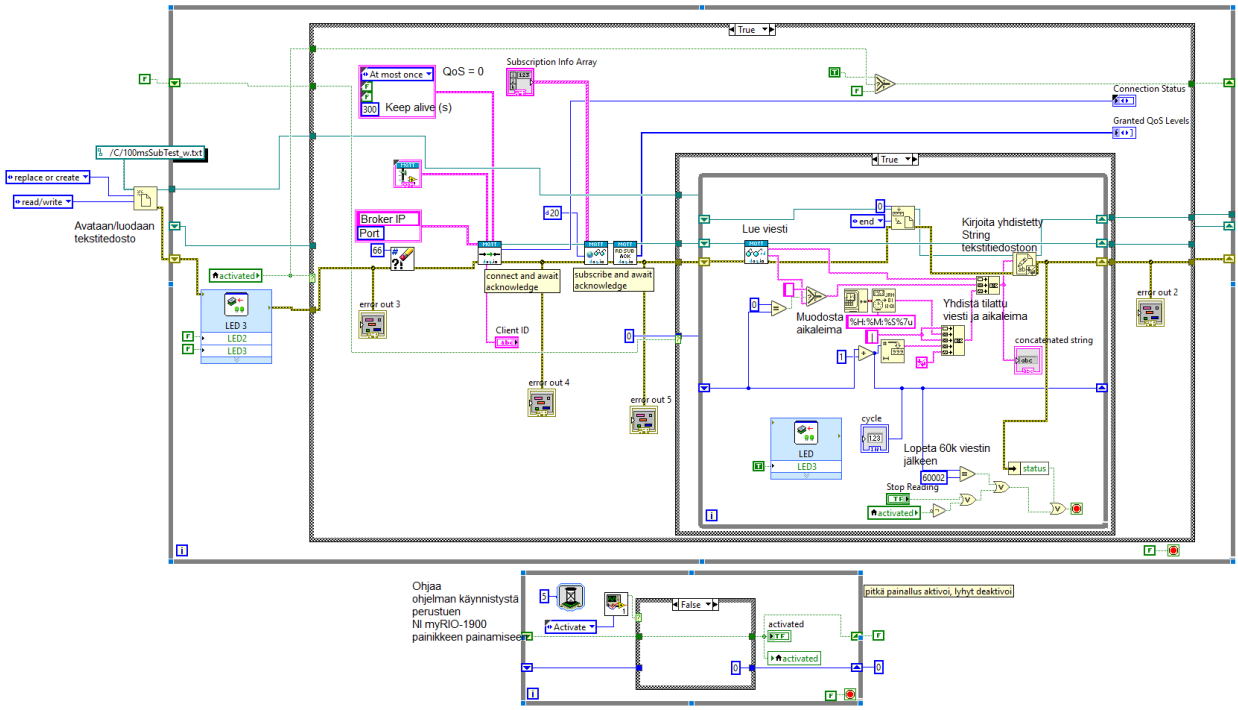
Mittauksissa käytettävät ohjelmat pidetään mahdollisimman yksinkertaisina. Ohjelmat pohjautuvat LVMQTT-kirjaston esimerkki Publish ”MQTT_Simple_Connect_and_Publish.vi”- ja Subscribe ”MQTT_Simple_Connect_and_Subscribe.vi” -clientteihin sekä liitteen 6 ohjelmaan. Kaikkiin paitsi ensimmäiseen Publish-toiminnon testaamiseen käytössä oleva LabVIEW-ohjelma on esitetty kuviossa 14.



KUVIO 14. Mittaustarkoituksiin käytettävä MQTT Publish LabVIEW-ohjelma.

Ohjelma avaa tai luo tekstitiedoston, johon kirjoitetaan mittausdata. Tämän jälkeen luodaan asetusten mukaisesti yhteys MQTT Brokerille. IP- ja Portti-asetukset ovat tietoturvasyistä sensuroitu kaikissa työn kuvissa. While-loopin sisällä ohjelma muodostaa viestin sisällön kellonajasta ja kierrosluvusta. Muodostettu String-muuttuja lähetetään kohti Brokeria ja sen jälkeen kirjoitetaan tekstitiedostoon. Lähetyksen QoS-asetus ohjaa Case Structure ja valitsee siihen perustuen lähetysmuodon. Kuviossa on esitetty QoS 2 -asetuksen Case Structure. While-loop sisältää ajastimen, jolla voidaan ohjata lähetysnopeutta. Ohjelma on niin yksinkertainen lähetin, kuin on mahdollista tehdä ja ei sisällä mitään ylimääräistä toiminnallisuutta indikaatio LED:ien ja viestin kirjoituksen ulkopuolella.

Subscribe toiminnon testeihin käytössä oleva LabVIEW-ohjelma on esitetty kuvissa 15.



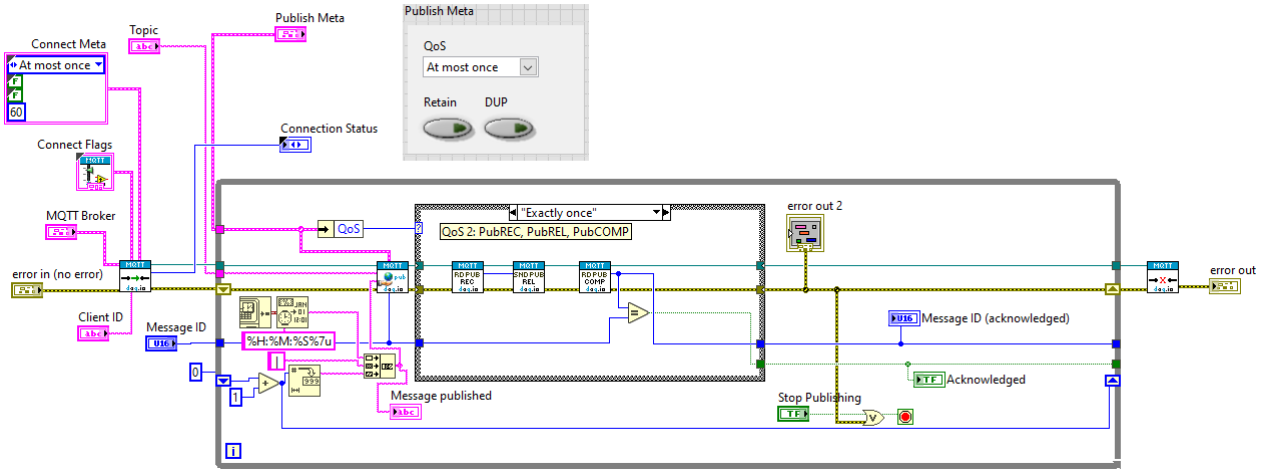
KUVIO 15. Mittaustarkoituksiin käytettävä MQTT Subscribe LabVIEW-ohjelma.

Ohjelma avaa tai luo mittausdatan kirjoitukseen tekstitiedoston, jonka jälkeen ohjelma yhdistää asetusten mukaisesti MQTT Brokerille. Suurempi Case Structure ohjaa ohjelman toimintaa ja kun se on FALSE niin mitään ei tapahdu. Ohjelma käynnistyy, kun NI myRIO-1900:n painiketta painetaan 250ms ajan ja tätä ohjataan alimmalla While-loopilla. Sisempi Case Structure ei tee tällä hetkellä mitään ja on jäljelle jäänyt testiversiosta. Sisemmän While-loopin sisällä ohjelma lukee saapuvan viestin ja muodostaa viestin sisällöstä, kellonajasta ja kierrosluvusta Stringin-muuttujan ja kirjoittaa sen tekstitiedostoon. Ohjelmassa on myös vastaanotettavien viestin raja-arvon asettaminen, jotta tekstitiedoston koko ei kasva liian suureksi.

Näiden kahden ohjelman permutaatioiden avulla suoritetaan kaikki pilottia edeltävät mittaukset. Subscribe-mittauksissa käytetään lähettäjänä toisella NI myRIO-1900:lla operoitavaa Publish-mittausohjelmaa. Tämä mahdollistaa molempien toimintojen samanaikaista mittausta.

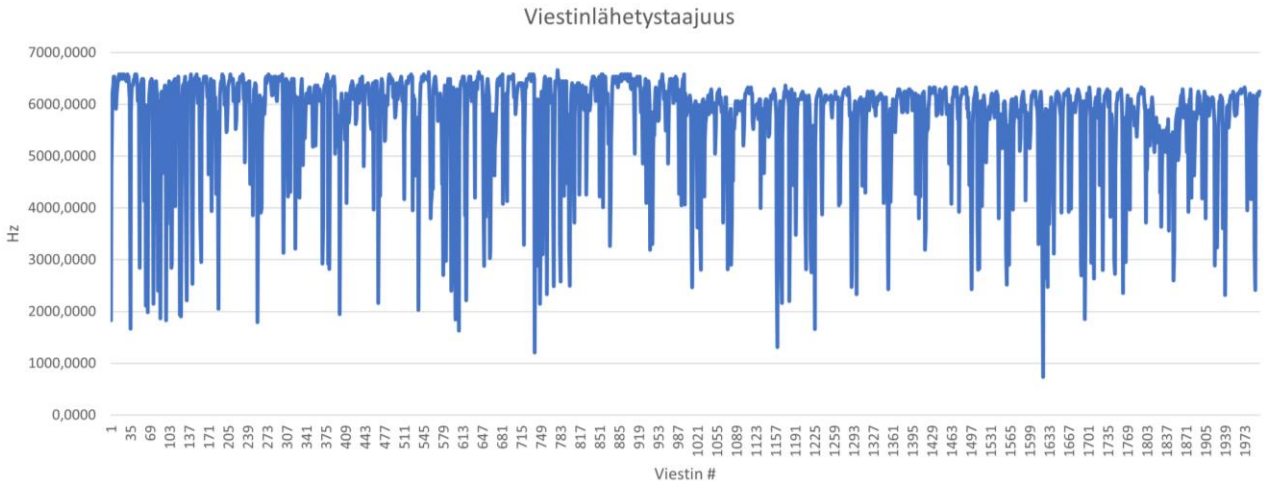
3.6 NI myRIO-1900 mittaustulokset

Ensimmäinen mittaus suoritetaan pelkästään Publish-clientin kanssa. Halutaan nähdä mihin mahdollisimman yksinkertainen Publish-ohjelma pystyy. Käytetään mittauksien tekemiseen kuviossa 16 esitettävää ohjelmaa. Ohjelma on muuten identtinen, mutta tekstitiedoston muokkaamisen toiminnot ovat poistettu.



KUVIO 16. Yksinkertaisin mahdollinen Publish client -ohjelma aikaleimankirjoituksella.

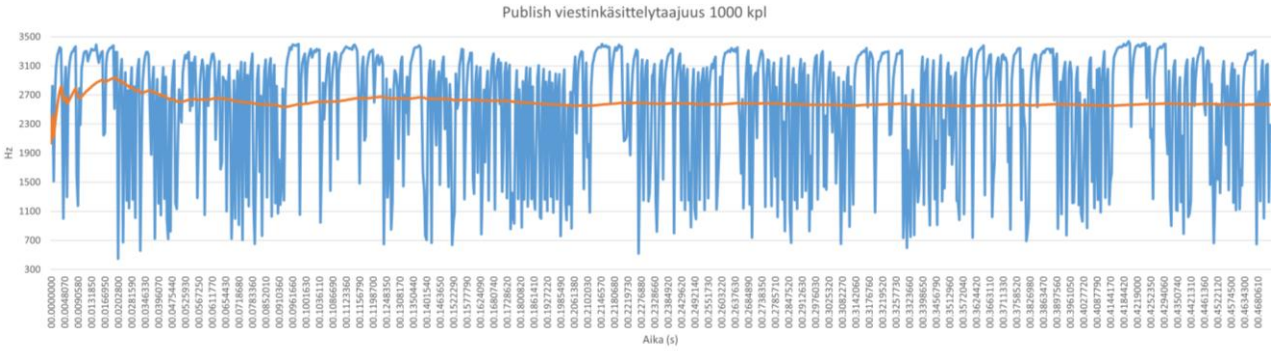
Koska ei luoda tekstitiedostoa, kerätään mittausdata PuTTY-terminaalin avulla. Tämä rajoittaa datapisteiden määrän kahteentuhanteen per mittaus. Suoritetaan kymmenen mittausta ohjelmalla ja kerättään jokaisessa mittauksessa kaksituhatta datapistettä. Yhden mittauksen viestinlähetystaajuudet ovat esitetty kuviossa 17.



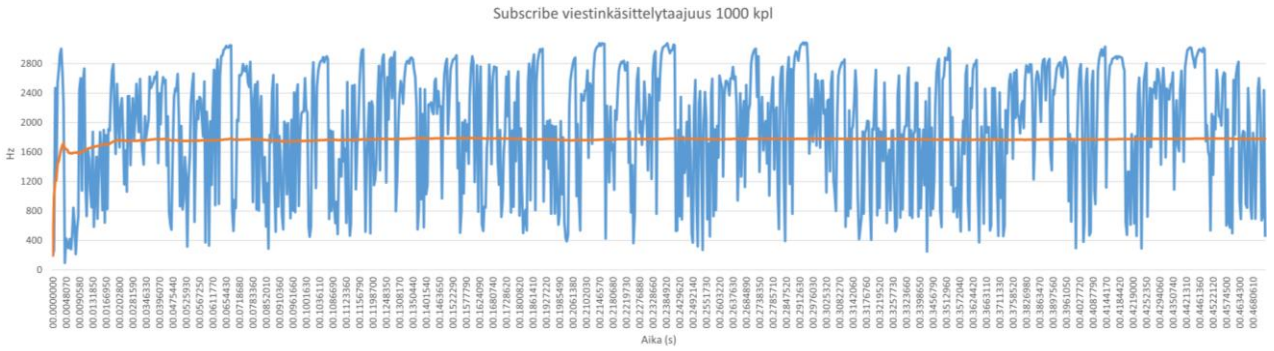
KUVIO 17. NI myRIO-1900 no load Publish testimittauksen viestinlähetystaajuus.

Kaikkien kymmenen mittauksen viestien viestinlähetystaajuuden keskiarvo oli 5,8 kHz (0,203 ms). Mittauksien viesteissä lyhyin viestinlähetysväli oli 0,147 ms ja pisin 2,71 ms. Taas kerran viestinlähetystaajuus on hyvin epätasaista ja amplitudi on suuri. Toisaalta NI myRIO-1900 pystyy ideaaliolosuhteissa erittäin huomattavasti nopeampaan viestintään verrattuna IoT Gateway -sovellukseen.

Seuraavassa mittauksessa palataan luvussa 3.5 esitettyyn Publish client -ohjelmaan. Siirrytään myös samanaikaisesti tutkimaan Publish- ja Subscribe-toimintoja. Ensiksi tutkitaan taas rajoittamatonta viestinlähetystä ja tutkitaan voiko tilaaja pysyä lähettäjän tahdissa. Mittauksessa Publish- ja Subscribe-tapahtuvat ovat molemmat QoS 1 -tasolla. Mittauksia suoritetaan vain yksi. Mittauksen kesto on 5 min 11,94 s ja lähetettyjä viestejä on 705683. Viestien suuresta määrästä johtuen muodostetaan kuviot vain tuhannen ensimmäisen viestin perusteella. Kuviossa 18 esitetään Publish-clientin viestinlähetystaajuus ja Kuviossa 19 esitetään Subscribe-clientin viestienkäsittelytaajuus.



KUVIO 18. Publish-client viestinlähetystaajuus 700k mittaukseen ensimmäisessä tuhannessa viestissä.



KUVIO 19. Subscribe-client viestinkäsittelytaajuus 700k mittaukseen ensimmäisessä tuhannessa viestissä.

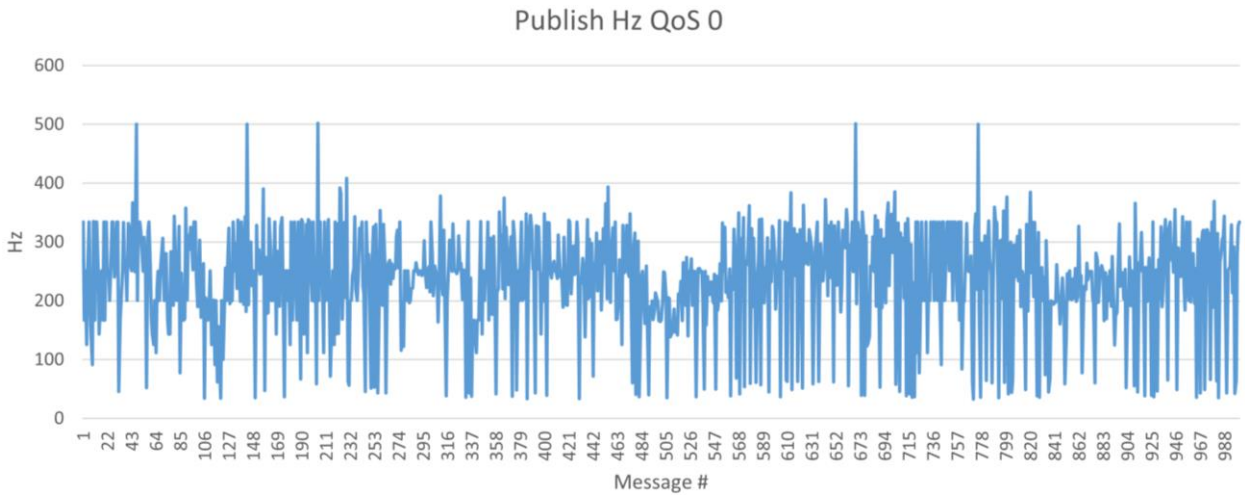
Mittauksen aikana Publish-clientin viestinlähetystaajuuden keskiarvo oli 2,26 kHz (0,442 ms). Mittauksen viesteissä lyhyin viestinlähetysväli oli 0,277 ms ja pisin 1,86 s. Subscribe-clientin viestinkäsittelytaajuuden keskiarvo oli 1,49 kHz (0,673 ms). Mittauksen viestien lyhyin viestinkäsittelyväli oli 0,312 ms ja pisin 20,1 ms. Tarkasteltiin lisäksi Subscribe-clientin desynkronointia ja saapuvien pakettien järjestystä. Kuviossa 20 esitetään PubSub-desynkronisaation käyrä.

KUVIO 20. Subscribe- ja Publish-aikaleimojen desynkronisaatio rajoittamattomassa viestinlähetysmittauksessa.

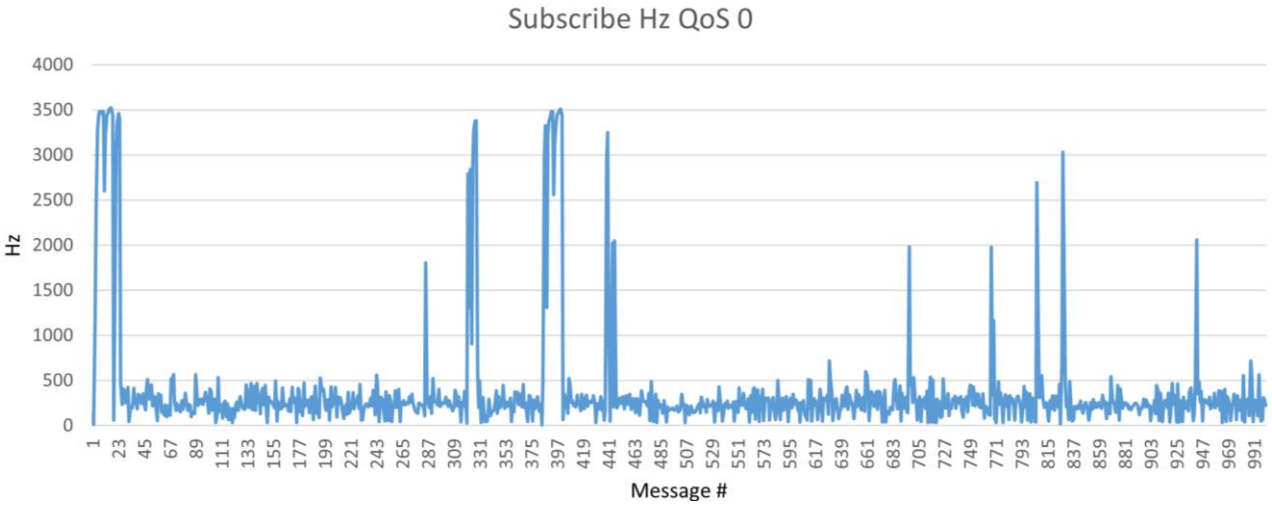
Tarkastellessa desynkronisaatiota se vaikuttaa olevan erittäin huomattavaa tällä viestinlähetystaajuudella. Viiden minuutin aikana viestin lähetystapahtuma ja vastaanottaminen siirtyvät yli kahden ja puolen minuutin päähän toisistaan. Nähdään myös desynkronisaation kasvamisen olevan suuressa skaalassa hyvin lineaarista muutamia poikkeuksia lukuun ottamatta. Tarkastellessa pakettien saapumisjärjestystä huomataan niiden saapuneen lähetysjärjestyksessä ja yhtään viestiä ei jäänyt saapumatta.

Seuraava NI myRIO-1900 suoritettava mittaus on äskeisen mittaustilanteen toistaminen tällä kertaa asettamalla loopin toimintaa ohjaamaan ajastin, joka siirtyy seuraavaan kierrokseen vasta, kun kello on 3 ms jaollinen. Lisäksi mittauksien välille asetetaan eri QoS-asetukset. Viestejä lähetetään vain kaksikymmentätu-

hatta ja mittauksia suoritetaan yksi Publisher QoS 0 -asetuksella ja toinen Publisher QoS 1 -asetuksilla. Kuviossa 21 esitetään Publish-clientin viestinlähetystaajuus QoS 0 -asetuksella ja Kuviossa 22 esitetään Subscribe-clientin viestinkäsittelytaajuus. Molemmat käyrät koostuvat tuhannesta ensimmäisestä viestistä.



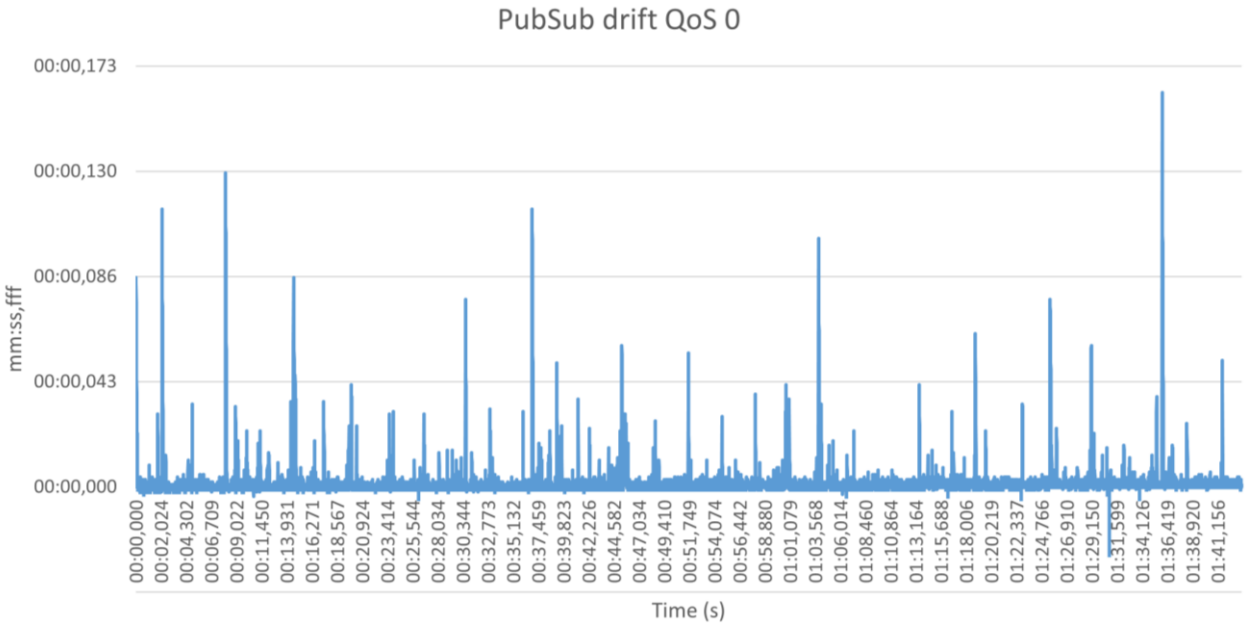
KUVIO 21. Publish-clientin viestinlähetystaajuus QoS 0 -asetuksella. 1000 viestiä.



KUVIO 22. Subscribe-clientin viestinkäsittelytaajuus Publisher QoS 0. 1000 viestiä.

Ohjelma ei pystynyt aina lähettämään viestiä 3 ms välein, vaikka aikaisemmissa mittauksissa lähetystiheys on pystynyt olemaan tätä mittausta korkeampi. Tämä luultavasti johtuu mittaustapahtumien ajoittumisesta ja mahdollisista muista toiminnallisista heittelyistä. Lähetystaajuus on kuitenkin huomattavasti tasaisempi-

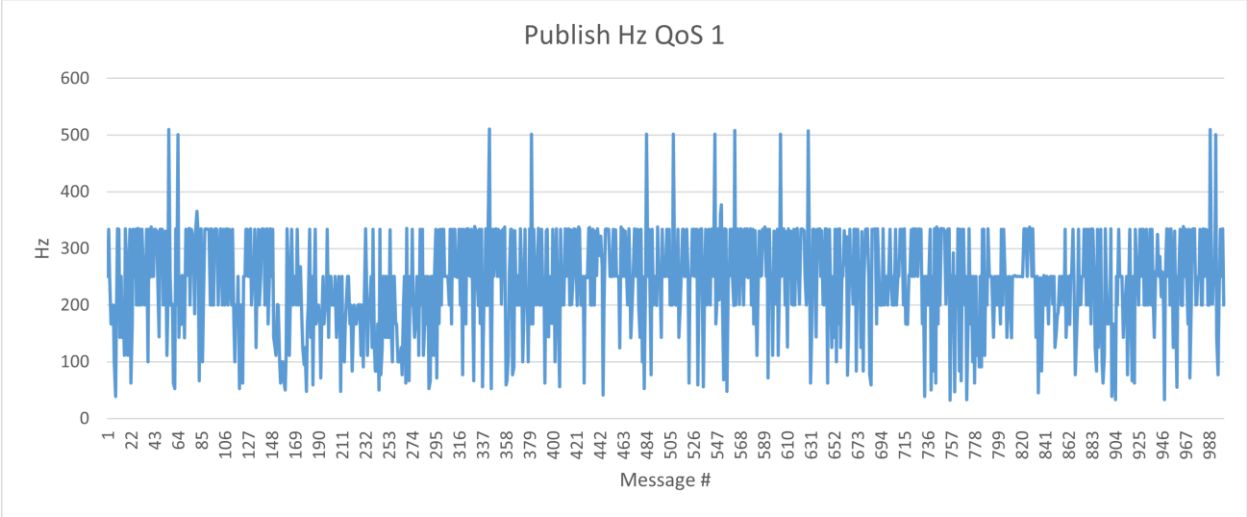
laatuista ja amplitudi on myös paljon alaisempi. Subscribe clientin -toimintaa arvioitaessa tutkitaan myös taas desynkronisaatiota. Kuviossa 23 esitetään PubSub-desynkronisaation käyrä QoS 0 -asetuksella.



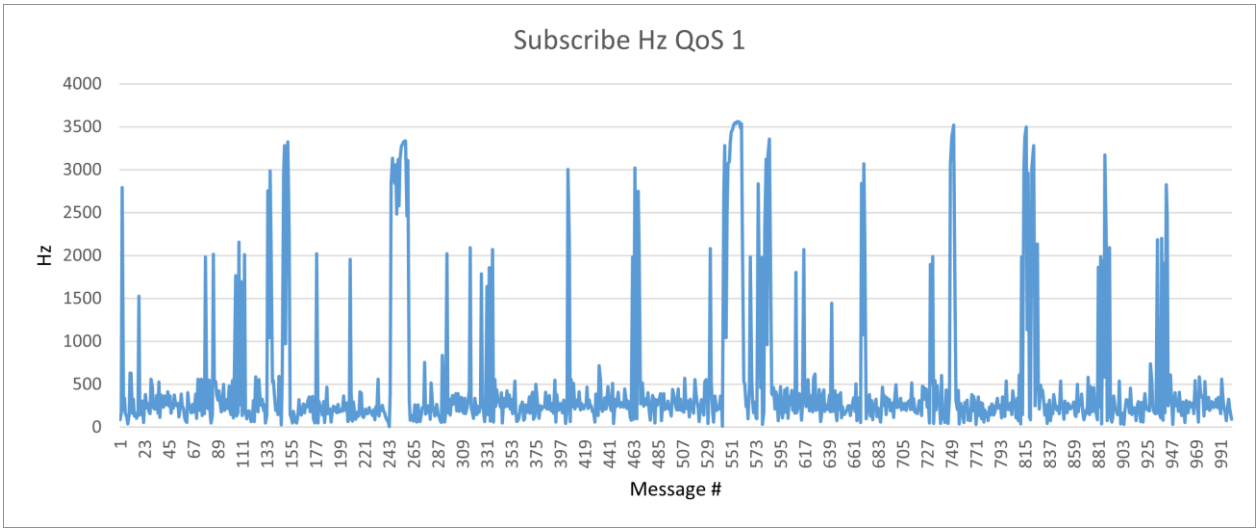
KUVIO 23. Subscribe ja Publish aikaleimojen desynkronisaatio QoS 0 -asetuksella. Koko mittauksen ajalta.

Rajoitetulla viestinlähetyksellä huomataan Subscribe-clientin pysyvän tahdissa mukana. Viestinkäsittelytaajuus on hyvin tasaista muutamia suuria nousuja lukuun ottamatta ja melkein koko mittauksen ajan pysytään hyvin lähellä 0 ms viivettä ja viive ei ehtinyt kertyä missään kohtaa. Muutamia korkeita piikkejä viiveessä syntyy, mutta nämä katoavat nopeasti. Mittauksesta myös saadaan varmistu siihen, että NI myRIO-1900 -laitteiden kellot eivät etene tasaisella tahdilla, koska muutamissa mittauspisteissä desynkronisaatio muuttuu negatiiviseksi, mikä on mahdotonta, koska ei viestiä voida käsitellä ennen sen lähettämistä. Mittauksessa myös menetettiin 11 viestiä lähettäjältä. Tämä on osittain odotettua, koska Subscribe Clientin QoS-asetus on tasolla 0.

Vertaillaan näitä tuloksia nyt Publisher QoS 1 -asetuksella suoritettavaan vastaavaan mittaukseen. Kaikki muu on täysin vastaavaa edelliseen mittaukseen paitsi QoS-asetus Publish-clienteillä. Kuviossa 24 esitetään Publish-clientin viestinlähetystaajuus mittauksessa ja Kuviossa 25 esitetään Subscribe-clientin viestinkäsittelytaajuus mittauksessa. Molemmat käyrät koostuvat tuhannesta ensimmäisestä viestistä.

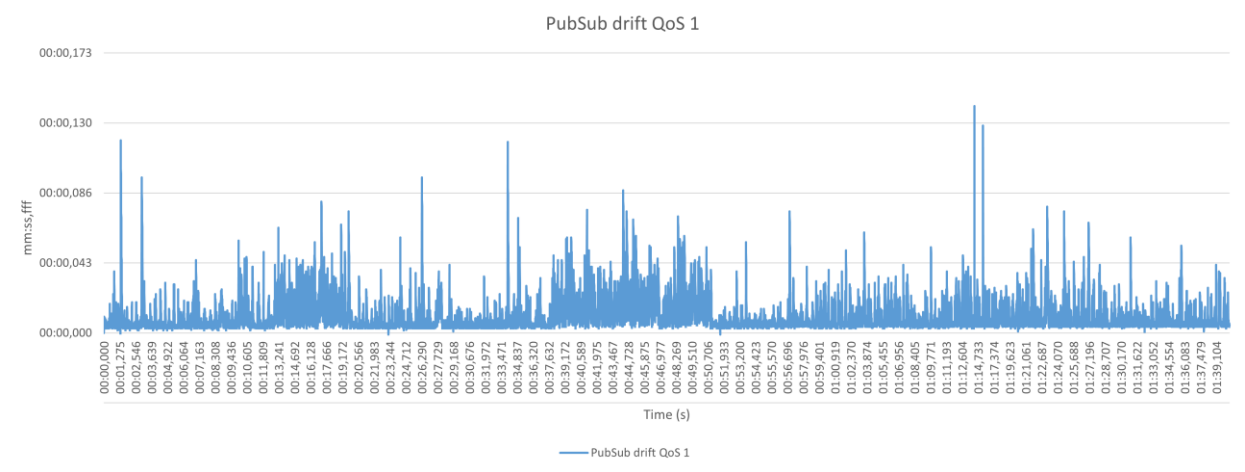


KUVIO 24. Publish-clientin viestinlähetystaajuus QoS 1 -asetuksella. 1000 viestiä.



KUVIO 25. Subscribe-clientin viestinlähetystaajuus Publisher QoS 1 -asetuksella. 1000 viestiä.

Publish client -toiminta muuttuu vähän QoS 1 -asetuksen seurauksesta. Viestinlähetystaajuuden amplitudi pysyy samana, mutta viestinlähetystaajuus pysyy tasaisemmin 200–330 Hz välillä. Silti piikkejä tapahtuu edelliseen mittaukseen verrattavasti ja taajuus heittelee paljon. Vaikuttaa QoS 1 -toiminta kuitenkin vakaammalta. Subscribe-toiminnon arvioimiseen kuviossa 26 esitetään PubSub-desynkronisaation käyrä Publisher QoS 1 -asetuksella.



KUVIO 26. Subscribe- ja Publish-aikaleimojen desynkronisaatio Publisher QoS 1 -asetuksella. Koko mittauksen ajalta.

Viestinkäsittelytaajuus on vertailtaessa edelliseen mittaukseen huomattavasti epätasaisempaa, mutta se voi vain johtua suoraan korkeammista viestinlähetystaajuuksista. Suurin havainto on Desynkronisaation käyttäytyminen viestinkäsittelyssä. Desynkronisaatio on noussut ja on huomattavasti korkeammalla QoS 0 -mittaukseen verrattuna. Desynkronisaatio silti on hyvin alhaista ja ei missään kohtaa ala kasvamaan. Lisäksi tässä mittauksessa menetettiin 13 viestiä. Numero on suurempi kuin QoS 0 mittauksessa, mutta vain hyvin vähän. Syynä voi olla esimerkiksi suurempi viestinlähetystaajuus, tietoverkon ongelmat, ohjelma vaatinut erinäisistä syistä enemmän aikaa suorittaa ja ei ehtinyt tilata ennen uuden viestin saapumista. Syitä on monia ja ei voida tämän mittauksen avulla selvittää oikeaa syytä tai muodostaa syy-seuraussuhdetta menetettyjen viestien ja nousseeseen viestinlähetysnopeuteen.

3.7 Lopputulokset

Suoritettujen mittauksien perusteella saadaan hyvä korkean tason kuva IoT Gateway -ohjelman sekä LVMQTT-kirjaston LabVIEW-ohjelmien toiminnasta. Joitakin aukkoja toiminnan selvittämisessä jäi, mutta niihin ei ole työn tilaaja vaatinut selvitystä ja on tyytyväinen kerätyn tiedon kanssa. Nyt voidaan käyttää saatuja tietoja pilottijärjestelmän suunnittelemiseen ja järjestelmän toiminnallisuuden pohjana.

IoT Gateway -sovelluksen MQTT-toiminnot tyhjiössä pystyvät reagoimaan RapidVariable Trigger -tyypillä noin 350–400 ms välisiin tapahtumiin luotettavasti pienellä viestikoolla. Parhaimmillaan reaktioaika voi olla lähempänä 300 ms, mutta se ei luotettavasti voi toimia tällä tasolla. Timer Trigger pystyy noin 100 ms väliseen viestintään pienellä viestillä ja vähäisillä etsittävillä Handlebars Placeholdereilla. Monien samanaikaisten Trigger-tiedostojen käyttämisessä pitää olla hyvin tarkka, koska Trigger-tapahtuvat voivat helposti estää toisien Trigger-tapahtuman tapahtumisen. QoS 1 -asetus myös huomattavasti pahentaa tätä ongelmaa. Pitää nämä seikat pitää mielessä, kun suunnitellaan ohjelman käyttöä kommunikaatiojärjestelmässä. Hitaahkon toiminnan takia ei sovellusta voida käyttää sovelluksissa, jotka vaativat lähetystapahtumien olevan alle 100 ms välein. Timer Triggerin kanssa on myös se ongelma, että viesti lähetetään aina, vaikka robotti olisi poissa käytöstä, kunhan ohjelma on tietokoneella päällä ja on yhteys robottiohjaimeen.

LVMQTT-kirjaston käyttäminen NI myRIO-1900:n kanssa onnistui hyvin ja viestinnän alaraja on huomattavasti nopeampi verrattuna IoT Gateway -ohjelmaan. Publish client -mittauksissa huomattiin se, että ohjelmalla on suuri merkitys viestinnän nopeuteen. Pitää tämän takia testata valmistettuja ohjelmia myös myöhemmin, jotta niiden toiminnallisuudesta voidaan olla varmoja. Kaikissa mittauksissa viestintä oli hyvin vaihtelevaa ja lähetystaajuus ei pysynyt tasaisena lähetyksien välillä. Huomattiin korkealla tasolla joitakin tasaisuuteen viittaavia käytöksiä, mutta yksittäisten viestien välisestä ajasta ei ole mitään varmistuksia. Pystytään NI myRIO-1900:lla silti pääsemään kHz lähetystaajuuksiin, jos sitä tarvitaan. QoS-asetuksilla on vaikutusta toimintaan mutta se ei ole erityisen huomattavaa ja toiminnallisuus ei kärsi ainakaan 300 Hz toimintatasolla.

Subscribe clientin -toiminta on raskaampaa kuin Publish-clientin. Subscriber-client ei pysty niin korkeisiin viestinkäsittelytaajuuksiin, mutta käsittely on silti mahdollista suorittaa kHz taajuusalueella. QoS 1 -asetus mahdollistaa kuitenkin kaikkien viestien vastaanottamisen, vaikka se aiheuttaisi desynkronisaatiota lähettäjän kanssa. Mittauksien mukaan tilaaja pystyy pysymään viestinlähetyksen perässä ongelmitta vähintään muutaman sadan Hz alueella, tarkkaa katkaisupistettä ei kuitenkaan vielä tiedetä.

Mittaustuloksissa on aukkoja eri huomioimattomien muuttujien, epävarmuuksien tai oletuksien muodossa. Kaikkia muuttujia ei voida tarkkailla ja kaikkien ohjattavien muuttujien vaikutuksia ei voida aikataulun sisällä tutkia, joten jouduttiin keskittymään vain helpoiten ja yleisesti muuttuvien muuttujien vaikutuksiin. Lisäksi vaikutuksia tutkittiin vain muutamien valittujen testiarvojen avulla. Lisäksi joidenkin mittauksien datasetit ovat vähän määrällisesti alhaisia, joka voi piilottaa harvinaisia tilanteita.

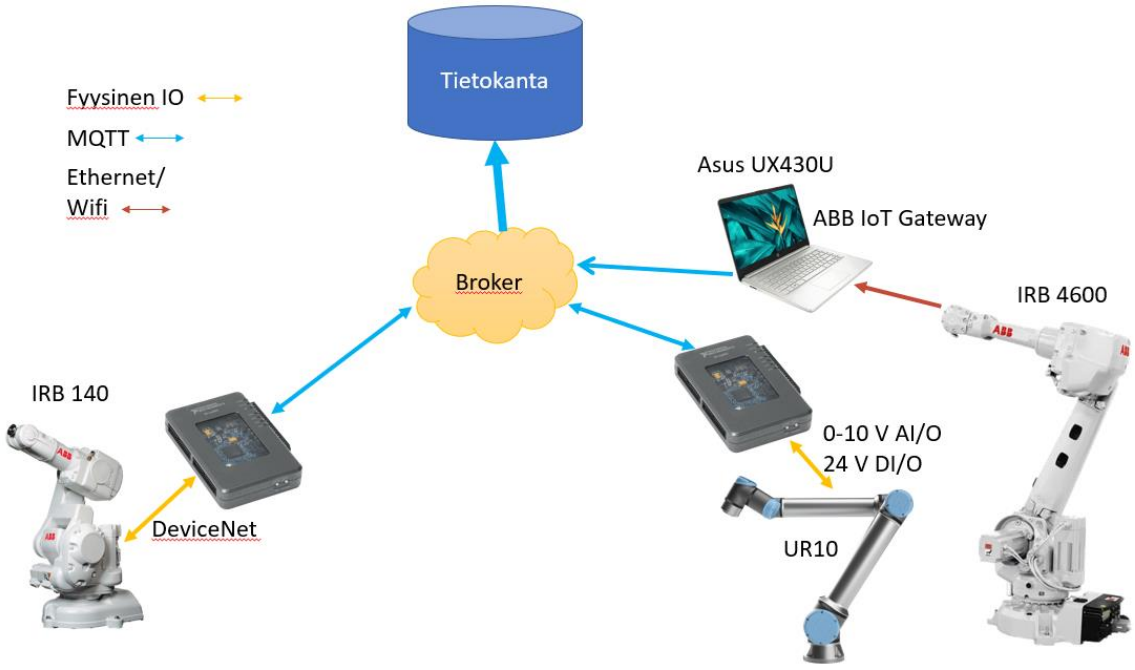
Oletukset esiintyvät IoT Gateway -ohjelman mittauksissa siinä, että joudutaan luottamaan täysin Log-tiedoston toimintaan. Kaikissa NI myRIO-1900 -laitteilla suoritetuissa mittauksissa luotetaan kellojen toimivan yhdenmukaisesti, joka mittauksissa on todistettu olevan virheellinen oletus. Kellojen toiminta on hyvin vaikeaa määrittää ja mitata, joten niihin luottava data on vähintään lievästi väärää. Vääryyden tasoa ei voida sanoa, mutta siitä syntyvät virheet eivät näin lyhyissä mittauksissa pitäisi olla liian suurta kellon toiminnan tarkastelun perusteella. Lisäksi mittauksissa kerätty tieto on aina vähän harhaanjohtavaa, koska itse ohjelmalla on niin suuri vaikutus toimintaan.

Jatkona tälle tutkimukselle olisi syytä tutkia tarkemmin IoT Gateway:n avulla monien samanaikaisten Trigger-tapahtumien interaktioita. Näissä tilanteissa on hyvin paljon muuttujia, jotka vaikuttavat toimintaan ja olisi hyvä saada selville, miten voidaan monien viestin kanssa toimia paremmin. RapidVariable- ja Timer Trigger -toiminta on aika selkeää, mutta muitakin Trigger-toimintoja voisi tutkia. NI myRIO-1900 -testeissä voitaisiin keskittyä tilaukseen ja löytää tarkka käsittelytaajuus, jossa QoS 1 -asetus alkaa kasvattamaan desynkronisaatiota. Lisäksi monien Topicien tilaamista voisi tutkia.

4 JÄRJESTELMÄN SUUNNITTELU JA PILOTTI

4.1 Pilotin laitteisto ja käyttötapaus

Nykyisellä laitteistolla valmistettava järjestelmä koostuu IRB 4600 -robotista, johon kytketään Asus UX430U -tietokoneella operoitava ABB IoT Gateway, UR10-robotista, joka kytketään luvun 3.1 mukaisesti NI myRIO-1900 -laitteeseen ja IRB 140, joka myös kytketään luvun 3.1 mukaisesti NI myRIO-1900:n. Tämä tarkoittaa UR10-robotin kommunikaation koostuvan ulospäin Digitaalisignaaleista sekä analogisignaaleista, mutta voidaan vastaanottaa vain yksi analogisignaali. IRB 140 taas pystyy käsittelemään ulospäin seitsemää analogisignaalia ja vastaanottaa neljä analogisignaalia. Syntyvän järjestelmän topologinen kuvaus on esitetty kuviossa 27.

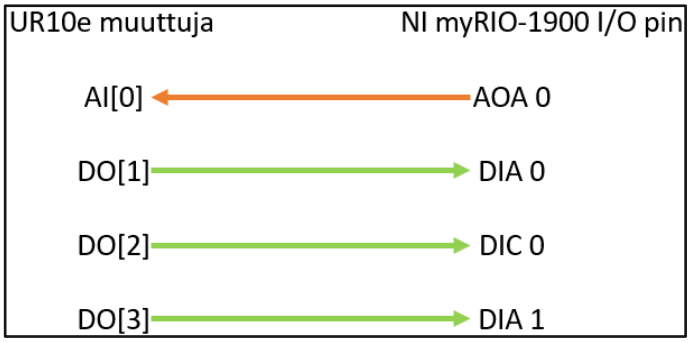


KUVIO 27. Projektin pilottijärjestelmän topologinen kuvaaminen

Projektin pilottijärjestelmä rakennetaan työn tilaajan kanssa päätetyn käyttötapauksen ympärille. TAMK:lla ei ole valmista prosessia johon järjestelmää tarvitaan, joten tapaus on keksitty työtä varten ja tarkoituksellisesti yksinkertainen. Käyttötapaus on suunniteltu simuloimaan valmistuslinjamaista toimintoa, jossa kappale kulkee robotilta robotille.

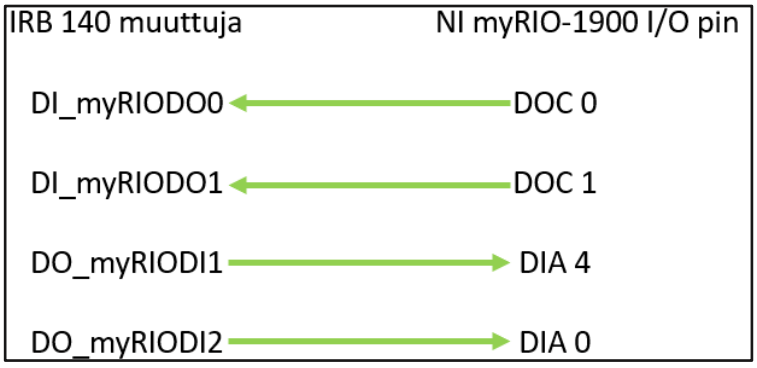
Koska IoT gateway ei tue Subscribe-ominaisuuksia sitä käytetään prosessin alkupisteenä. IRB 4600 suorittaa lyhyen ohjelman, jonka päätyttyä lähetetään satunnaisesti yksi kahdesta viestistä UR10-robotille. IRB 4600:lle asetetaan myös ajastettu datankeruu tietokantaan.

UR10:n toiminta riippuu siitä, kumpi kahdesta viestistä lähetetään. Jos lähetetään alle 4 V analogisignaali, UR10 suorittaa ohjelman A. Jos lähetetty signaali on yli 4 V analogisignaali UR10 suorittaa ohjelman B. molempien ohjelmien päätöksenä UR10 lähettää viestin IRB 140:lle. Viestisisältö riippuu siitä kumpi UR10 ohjelmista suoritettiin. Ohjelman A viesti sisältää vain yhden boolean-muuttujan ja ohjelman B viesti sisältää kaksi boolean-muuttujaa. Pilotissa käytettävät UR10 ja NI myRIO-1900 väliset kytkennät ovat kuviossa 28 mukaisia.



KUVIO 28. UR10 muuttujat ja NI myRIO-1900 I/O pinnien yhteydet

IRB 140 suorittaa pilotissa viimeisen osan. Se myös suorittaa yhden kahdesta ohjelmasta riippuen UR10 lähettämän viestin sisällöstä. Jos on kaksi boolean-muuttuja, suoritetaan ohjelma C, jos on yksi boolean-muuttuja, suoritetaan ohjelma D. Pilotissa käytettävät IRB 140 ja NI myRIO-1900 väliset kytkennät ovat kuvion 29 mukaisia.



KUVIO 29. IRB 140 muuttujat ja NI myRIO-1900 I/O pinnien yhteydet

4.2 Pilotti järjestelmän suunnitteluperiaatteet

Järjestelmän ja sen osien ohjelmien suunnittelussa pohjustetaan luvussa 2.1 esitettyihin suunnitteluperiaatteisiin. Halutaan järjestelmän olevan mahdollisimman yleispätevä ja käytettävissä mahdollisimman monen laitteen kanssa. Halutaan vähentää valmistajariippuvuus ja yksittäisien laitteiden ympärille rakentaminen. Järjestelmässä voi laitteet muuttua, mutta toiminta ei saa häiriintyä. Tämä myös johtaa siihen, että järjestelmä tulee olemaan modulaarinen ja helposti muokattavissa. Pitää uuden laitteen yhdistäminen olla kivutonta ja nopeaa. Muokkaamisen vaativuus riippuu laitteesta, mutta se pyritään laitteen tai ohjelman sisällä tehdä mahdollisimman helpoksi.

Järjestelmä tulee myös pitää erossa siihen yhdistettyjen robottisolujen toiminnasta. Tietysti järjestelmän tarvitsee olla toiminnassa, jotta sitä hyödyntävät työtoimenpiteet voidaan suorittaa automatisoidusti, mutta järjestelmän toimimattomuus ei saa estää solun käyttämistä muihin käyttötarkoituksiin.

Pyritään myös kaikessa mahdollisimman korkeaan suoritustehoon. Suoritusteholla tarkoitetaan lähetys-/vastaanottamiskapasiteettia ja luotettavuutta. Mitä korkeampi suoritusteho sitä käytettävämpi järjestelmä on ja se soveltuu suurempaan määrään käyttötarkoituksia.

IoT Gateway -sovelluksen ohjelmien suunnittelussa joudutaan huomioimaan Trigger-tapahtumien määrä ja niiden tyypit. Niiden sovellutus robottiohjelmaan pitää myös miettiä. Itse tiedostojen muotoilu on täysin standardoitu, paitsi payload-tiedostot kohdalla, joten seurataan valmistajan ohjeistusta.

LabVIEW-ohjelmoinnissa pyritään yksinkertaisuuteen, ja laajennettavuuteen. Yksinkertaista ohjelmaa on helppo muokata ja ymmärtää sekä suoritusteho on parempi. Laajennettavuudessa huomioidaan se, että tehdään valmiiksi kaikki IO-portteihin liittyvät toimet ja asetuksien muokkaamisesta tehdään mahdollisimman helppoa.

4.3 IoT Gateway ja IRB 4600 ohjelma

IoT Gatewayn -ohjelmisto on hyvin robottiohjelmaan sidonnaista, joten selitetään pilotissa IRB 4600:n robottiohjelmat ensin. Liikeohjelma on esitetty kuvassa 18.

```
PROC MQTT_Pilotti()
    !liiku kotipisteeseen
    MoveJ testHome, v100, z50, tCEAD_Extr;
    !aseta komennot pois päältä
    UR10_KomentoA := FALSE;
    UR10_KomentoB := FALSE;
    !liiku aloituspisteeseen
    MoveL testHome10, v100, z50, tCEAD_Extr;
    MoveL testHome20, v100, z50, tCEAD_Extr;
    FOR i FROM 0 TO randomNumber() DO !liiku satunnaismäärä kierroksia
        MoveL offs(testHome30,0,0,0+(i*20)), v100, z50, tCEAD_Extr;
        MoveL offs(testHome40,0,0,0+(i*20)), v100, z50, tCEAD_Extr;
        MoveL offs(testHome50,0,0,0+(i*20)), v100, z50, tCEAD_Extr;
        MoveL offs(testHome60,0,0,0+(i*20)), v100, z50, tCEAD_Extr;
        MoveL offs(testHome30,0,0,0+(i*20)), v100, z50, tCEAD_Extr;
    ENDFOR
    !jos satunnaisluku on tasaluku lähetä 3.5 V analogisignaali ja jos ei ole niin 4.5 V
    IF randomNumber() MOD 2 = 1 THEN
        UR10_KomentoA := TRUE;
    ELSE
        UR10_KomentoB := TRUE;
    ENDIF
    MoveJ testHome, v100, z50, tCEAD_Extr;
    WaitDI diExampleInput, 1;
ENDPROC

FUNC num randomNumber() !funktio generoi luvun 1-6
    Kierrosmaara := Rand()/65535*5 +1; !Rand()/65535= (0...1), eli kierrosmaara = (1...6)
    kierrosmaara := round(kierrosmaara); !pyöristää luvun kokonaisluvuksu
    RETURN kierrosmaara;
ENDFUNC
        VAR num kierrosmaara;
        PERS Bool UR10_KomentoB;
        PERS Bool UR10_KomentoA;
```

KUVA 18. IRB 4600 pilotin liikeohjelma.

Liikeohjelmassa kuljetaan aluksi kotipisteeseen ja resetoidaan viestintään käytettävät ohjattavat Boolean-muuttujat. Kuljettuaan liikeradan alkupäähän robotti generoi satunnaisesti luvun väliltä 1–6 ja suorittaa luvun mukaisen määrän kierroksia. Kierrosten jälkeen generoidaan uusi luku, jota käytetään päättämään lähetettävä viesti. IRB 4600:n pilottiosuuteen kuuluu myös mainittu tietokantaviesti. Tietokanta vaatii aikaleiman formaatissa mitä IoT Gateway ei voi ulosantaa itsenäisesti. Tämä korjataan generoimalla ohjelmallisesti aikaleima, joka on muotoiltu oikealla formaatilla. Tähän käytettävä ohjelma esitellään kuvassa 19.

```
PROC main()
    ClkStart timestampClock;

    second := GetTime(\Sec);


    IF lastsec <> second THEN
        ClkReset timestampClock;
        MillisTimeStamp := TimestampGenerator();
        lastsec := second;
    ELSE
        MillisTimeStamp := TimestampGenerator();
        lastsec := second;
    ENDIF
ENDPROC

Func string TimestampGenerator()
    time := CTime();
    date := cDate();
    nmillis := ClkRead(timestampClock);
    sMillis :=StrPart(NumToStr(nMillis,3),2,4);
    TimeStamp := date + " " + time + sMillis;
    return TimeStamp;
ENDFUNC
PERS string MillisTimeStamp;
VAR clock timestampClock;
VAR num second;
VAR num lastsec:= 100;
```

KUVA 19. IRB 4600:n tietokantaviestin aikaleiman generointi.

Ohjelma toimii robotilla liikeohjelman ulkopuolella. Ohjelmassa ensiksi aloitetaan kello. Käsky ei tee mitään, jos kello on jo aloitettu, joten se ei vaadi erikoiskäsittelyä. Haetaan kellosta sekunti ja verrataan, onko se edellisestä aikaleimasta eroava. Jos sekunti eroaa, käynnistetään kello uudelleen. Tämä tehdään, jotta kellossa on vain millisekunteja. Itse aikaleima luodaan funktiossa ”Timestamp-Generator”. Kerätään robotin muistista aika ja päivämäärä ja luetaan millisekuntikellosta millisekunnit. Muut tiedot ovat string-muuttujia, joten muutetaan kellosta luettu data string-muotoiseksi. Viimeiseksi muodostetaan vaaditusti formatoitu aikaleima.

Näiden ohjelmien ympärille on muodostettu kaikki vaadittavat IoT Gateway -tiedostot. Controller- ja Trigger-tiedostot ovat esitetty kuvassa 20. Trigger-tiedostot ovat leikattu lyhyeksi, mutta jokaisessa tiedostossa tämä arvo on sama.

KUVA 20. Pilotin IRB 4600 IoT Gatewayn Controller-, Trigger- ja UR viestien Payload-tiedostot.

Tämä asetelma on hyvinkin vastaava luvussa 3.4 suoritettujen mittauksien kanssa. Controller tiedostossa määritellään kaikki kolme Trigger-tiedostoa käytettäväksi. UR10 Trigger -tiedostoissa on määritelty liikeohjelmassa kommunikaatiotarkoituksiin käytettävät muuttujat. Lisäksi UR10-kommunikaatiosta vastaavat Triggerit ovat asetettu QoS 1 -tasolle, jotta ne varmasti lähetetään. DBsender Trigger on asetettu toimimaan 150 ms ajoituksella. Tämän Triggerin Payload-tiedosto on liian suuri näytettäväksi, mutta se on liitteessä 6 esitetyn JSON-formaatin mukainen.

4.4 NI myRIO-1900 ohjelmisto

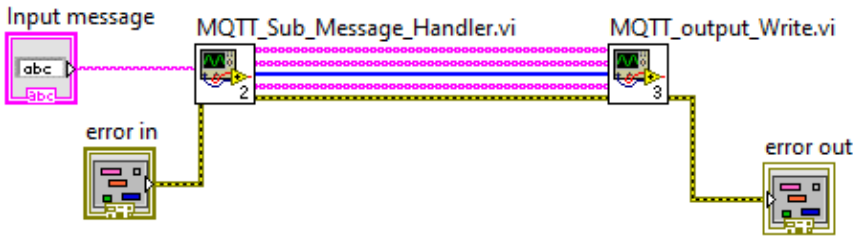
Pilottia varten kehitetty ohjelma on rakennettu testeissä käytettyjen LabVIEW-ohjelmien päälle. Ohjelman runko koostuu pääasiallisesti luvun 3.5 Subscribe client -ohjelmasta. Tähän on muokattu viestien käsitteleminen tekstitiedostopohjaisesta käsittelemisestä I/O:ta ohjaavaksi. Viestien lähetys perustuu NI myRIO-1900 IRQ-toimintoihin. Viestinlähetystapahtumat sidotaan tiettyjen DI/O-pinnien arvojen tarkkailuun ja nousureunalla lähetetään viesti. Passiivisesti ohjelma toimii Subscribe-clienttinä, kunnes tiettyihin DI/O-pinniehin tulee positiivinen signaali, jolloin ohjelma lähettää tähän pinniin konfiguroidun viestin.

Toteutustavan taustaideana on se, että Subscribe-toiminnot ovat passiivisia ja tilaajan pitää olla aina valmiina vastaanottamaan viestiä. Täten suoritetaan Subscribe-toimintoja jatkuvasti. Viestin lähettämin taas on täysin tietoinen valinta, joten se voidaan suorittaa taustalla haluttaessa. IRQ on erinomainen tapa toteuttaa lähetys, koska se ei ole sidottu tiettyihin kohtiin ohjelmaa. IRQ-toteutustavan haittapuolena on tapahtumien rajattu lukumäärä. NI myRIO-1900 tukee maksimissaan kahdeksaa samanaikaista IRQ-tapahtumaa. IRQ-tapahtumat voivat myös olla vain ajastettuja, AI/O-portin DIO0–DIO3 -pinnejä tai rungossa olevaa painiketta tarkastelevia. Pinnejä tarkastelevat IRQ-tapahtumat voivat olla joko nousevaa-, laskevaa- tai molempia reunoja tarkasteleva. Painiketta käytetään jo ohjelman käynnistämiseen sekä sitä ei voida sähköisesti automatisoida, joten sitä ei voida käyttää.

Toinen suunnittelupäätös liittyy NI myRIO-1900:n kohdistuvien ja lähtevien viestien formatoinnista. Koska laitteisiin kohdistuvalla viestinnän päämäärä on laitteen ulkopuolella, joten jokainen viesti kohdistuu I/O-portteihin. Tämä helpottaa viestien formatointia, koska saapuvia viestejä voi olla vain kahdenlaisia, analogisignaaleja tai digitaalisignaaleja. Signaalien formatoinniksi päätettiin seuraava muoto (xIm # | xIm # …).

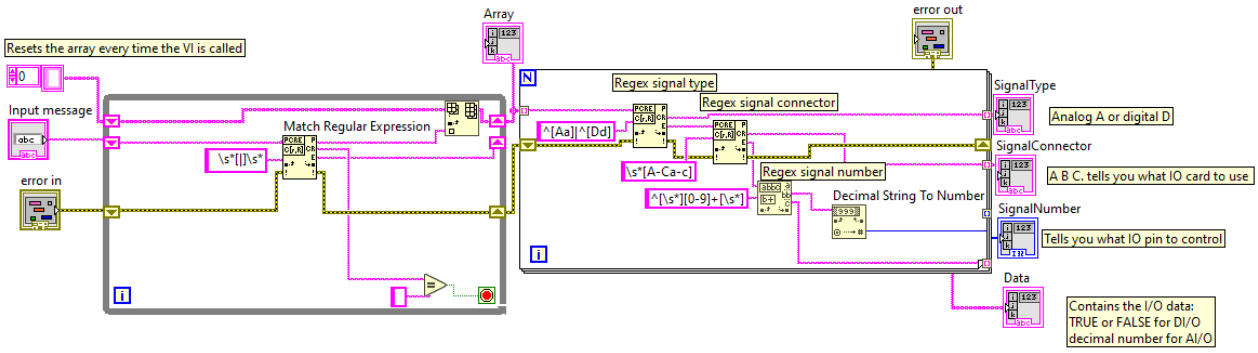
xIm # viestin formatoinnin selitys

Kaikki merkit ovat isoilla tai pienillä kirjaimilla. x tarkoittaa vaikutettavan signaalin tyyppi, digitaalisignaali D tai analogisignaali A. I on vain välimerkkinä, sitä ei tarvita mutta sen tilalla voi olla mitä tahansa, paitsi kirjaimet a, b tai c, myös pituudella ei ole merkitystä. m tarkoittaa I/O-porttia, johon halutaan signaalin vaikuttavan eli se on A B tai C. # tarkoittaa m-portilla olevan I/O-pinnin numeroa, tarkat pinnien numerot ja niiden positiot löytyvät luvusta 2.6. Pystyviivamerkki | toimii signaalien erottimena. Viestien formaatti on valittu sillä perusteella, että se on helppo ymmärtää mihin vaikuttaa. Viesti toimii itse omana kommenttinaan. Saapuvien viestin käsittelyyn käytetään ”SubReader.VI”-ohjelmaa, joka on esitetty kuviossa 30. Minkään ohjelman Front Panel ei ole toiminnallisesti merkittävä, joten niitä ei käsitellä.



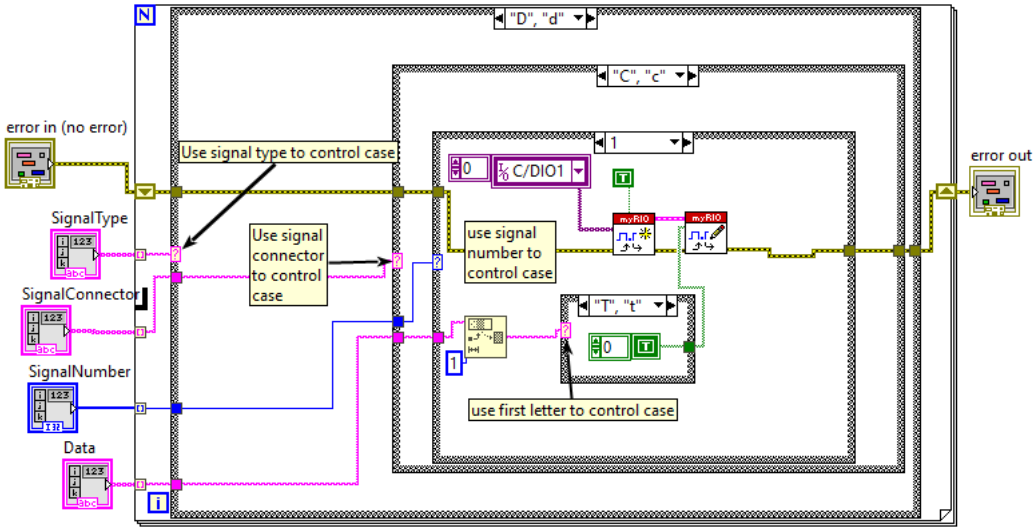
KUVIO 30. SubReader.VI-ohjelma.

Ohjelman tarkoitus on yhdistää oikeat viestienkäsittelyohjelmat ”MQTT_Sub_Message_Handler.vi” ja ”MQTT_output_Write.vi”. ”MQTT_Sub_Message_Handler.vi” käsittelee vastaanotettavat viestit ja ulosantaa niiden sisältävän I/O-informaation ”MQTT_output_Write.vi”:lle, joka kirjoittaa informaation mukaisesti arvot I/O pinneille. ”MQTT_Sub_Message_Handler.vi” on esitetty kuviossa 31.



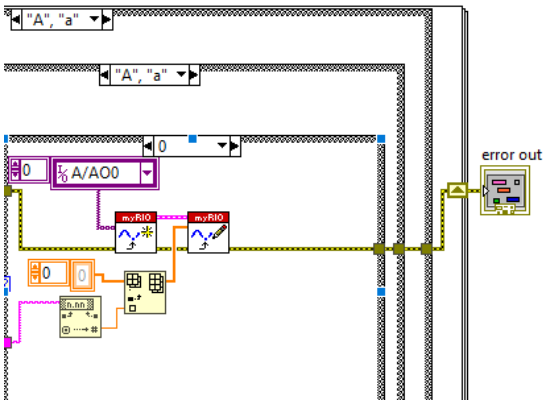
KUVIO 31. “MQTT_Sub_Message_Handler.vi”-ohjelman Block Diagram

VI ottaa vastaan saapuneen viestin ja jakaa sen sisällä olevat signaalit listaan regex-funktion avulla. Erotteluun käytetään aikaisemmin mainittua | merkkiä. Eroteltujen signaalien lista kuljetetaan For-looppiin, jossa signaaleista yksi kerrallaan kerätään I/O-informaatio eroteltuihin listoihin. Kerättävät tiedot ovat aikaisemman formaatin mukaista ja kommentoituja kuviossa. ”MQTT_output_Write.vi” ohjelma on esitetty kuviossa 32.



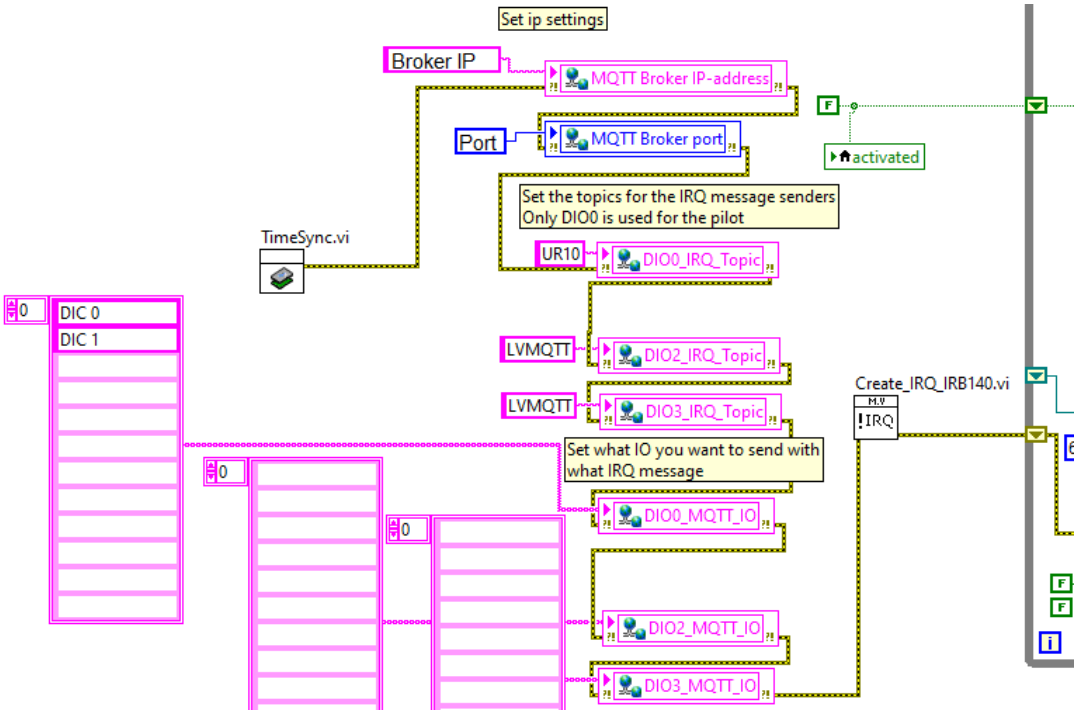
KUVIO 32. “MQTT_output_Write.vi”-ohjelman Block Diagram, Digital signal case

Viesteistä eroteltuja osia käytetään nyt ohjaamaan monia sisäänrakennettuja Case Structureja. SignaalinType-listalla päätetään Analogin ja digitalisignaalin välillä. SignalConnector-listalla päätetään mihin I/O-porttiin vaikutetaan. SignalNumber-listalla ohjataan mihin I/O-pinniin vaikutetaan. Data-lista sisältää viestien pinnitiedot, DI/O True tai False ja AI/O desimaaliluvun. Analogisignaalin case on esitetty kuviossa 33. Ainoa ero on siinä, että data syötetään suoraan string to double muuntamisfunktioon, eikä käytetä Case Structurea.



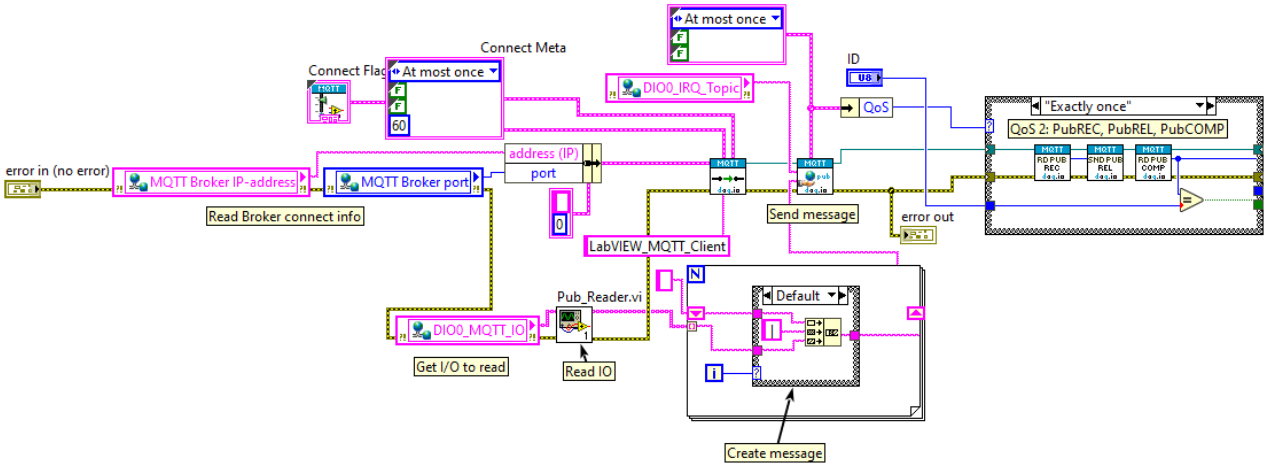
KUVIO 33. “MQTT_output_Write.vi”-ohjelman Block Diagram, Analog signal case

Ohjelman runko on liian suuri selkeästi esitettäväksi yhdessä kuviossa, joten käsitellään se osissa. Kuviossa 34 esitetään ohjelman aloituksessa suoritettava konfigurointi. Molemmille käytettäville NI myRIO-1900 -laitteille pitää tehdä omat ohjelmat, koska ne vaativat erilaiset konfiguraatiot ja projektissa ei ollut aikaa valmistaa tiedostopohjaista konfiguraatiota. Muuten ohjelmat ovat identtisiä.



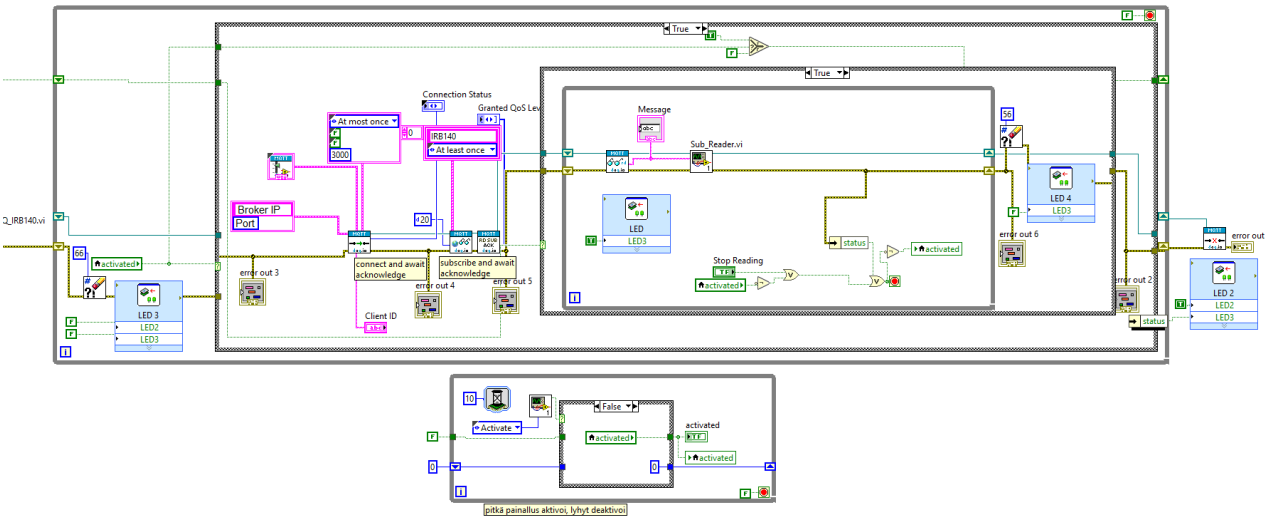
KUVIO 34. LabVIEW ohjelman alkukonfigurointi, Block Diagram.

Ensiksi ohjelmassa suoritetaan ”TimeSync.vi”, joka synkronoi NI myRIO-1900:n kellon NIST-serverin aikaan. Seuraavaksi tallennetaan Brokerin-yhdistystiedot, IRQ-tapahtuma spesifit Topic-tieto ja lähetettävät I/O-pinnit. DIO1 ei annetta Topic-tietoa, koska pinnin IRQ-tapahtuma on varattu ulostulojen nollaamiseen. ”Create_IRQ_IRB140.vi” muodostaa IRQ-tapahtumat. Kaikki IRQ-tapahtumat ovat nousureunaan reagoivia ja tapahtumien suorittavia ohjelmia on kaksi. Toinen lähettää viestin ja toinen nollaa ulostulot. Ulostulojen nollaamisen ohjelma on ”MQTT_output_Write.vi” ohjelmaa vastaava, mutta kirjoittaa DI/O pinneihin FALSE ja AI/O pinneihin 0 V. Lähetys tapahtuman IRQ-ohjelma on esitetty kuviossa 35.



KUVIO 35. “IRQ_SendMessage_Callback.vi”-ohjelman Block Diagram.

Ohjelmassa luetaan konfiguraatiossa asetettu Broker- ja I/O-tiedot. ”Pub_Reader.VI” on muuten ”Sub_Reader.vi” vastaava, mutta ”MQTT_output_Write.vi” vastaavassa ohjelmassa korvataan I/O-kirjoittaminen lukemisella. Luetut tiedot lähetetään For-looppiin, jossa muodostetaan formaatin mukainen viesti toiselle NI myRIO_1900:lle. Loput pääohjelmasta on esitettynä kuviossa 36. Ohjelma on pitkälti verrattava luvussa 3.5 esitettyyn Subscriber client -ohjelmaan. Ainoa huomattava ero ohjelmien välillä on viestin käsittely ja virheidenhallinta. Uudessa ohjelmassa viestin käsittely suoritetaan jo esitellyllä ”Sub_Reader.vi”:llä. Looppiin on lisätty hieman virheidenkäsittelyä siinä muodossa, että poistetaan toiminnassa mahdollisesti syntyviä virhesignaaleja, jotka eivät oikeasti haittaa ohjelman toimintaa.



KUVIO 36. Pääohjelman rungon Block Diagram.

4.5 IRB 140 ja UR10 pilotin robottiohjelmat

IRB 140:n pilotin robottiohjelma koostuu vain yhdestä liikeohjelmasta. Pilotti käyttötilanteessa ei myöskään IRB 140:n tarvitse suorittaa kommunikaatiota ulospäin. Ohjelmassa silti lähetetään mahdollisesti viesti toiminnallisuuden testaamiseksi. IRB 140:n robottiohjelma on esitetty kuvassa 21.

```
PROC PilottiMain ()
    WaitDI DI_myRIOD00, 1;
    WaitTime 1;
     MoveJ pHome, v1000, z50, tool1\WObj:=pTreduVino1;
    IF DI_myRIOD01 = 1 THEN
         MoveL testA10, v1000, z50, tool1\WObj:=pTreduVino1;
         MoveL testA20, v1000, z50, tool1\WObj:=pTreduVino1;
         MoveL testA30, v1000, z50, tool1\WObj:=pTreduVino1;
         MoveL testA40, v1000, z50, tool1\WObj:=pTreduVino1;
         MoveL testA50, v1000, z50, tool1\WObj:=pTreduVino1;
    ELSE
         MoveL testB10, v1000, z50, tool1\WObj:=pTreduVino1;
         MoveL testB20, v1000, z50, tool1\WObj:=pTreduVino1;
         MoveL testB30, v1000, z50, tool1\WObj:=pTreduVino1;
         MoveL testB40, v1000, z50, tool1\WObj:=pTreduVino1;
         SetDO DO_myRIODI2, 1;
    ENDIF
    setDO DO_myRIODI1, 1;
    MoveL testB10, v1000, z50, tool1\WObj:=pTreduVino1;
    MoveL Conv10, v1000, z50, tool1\WObj:=pTreduVino1;
    SetDO DO_myRIODI2, 0;
    setDO DO_myRIODI1, 0;
    MoveL Conv20, v1000, z50, tool1\WObj:=pTreduVino1;
    MoveL Conv30, v1000, z50, tool1\WObj:=pTreduVino1;
    MoveJ pHome, v1000, z50, tool1\WObj:=pTreduVino1;
ENDPROC
```

KUVA 21. IRB 140 pilotin robottiohjelma.

Robottiohjelma on hyvin yksinkertainen. Robotti odottaa, kunnes saa NI myRIO-1900:n yhdistetyn TRUE-signaalin ”DI_myRIODO0” I/O-porttiin. Signaalin saatua tarkastetaan tuliko viestin mukana TRUE-signaali myös DI_myRIODO1 I/O-porttiin. Ohjelmalla suoritetaan kappaleen nostamista simuloiva ohjelma. Nostotapahtuma valitaan signaalien perusteella. Signaalien perusteella myös ohjataan testiviestin lähettämistä. Loput ohjelmasta ensin tyhjentää NI myRIO-1900:n ulostulot ja suorittaa loput liikeohjelmasta.

UR10 pilotin robottiohjelma koostuu myös vain yhdestä liikeohjelmasta. robottiohjelma on esitetty kuvassa 22.

```
Program
  Robot Program
    Set DO[1]=Off
    Set DO[2]=Off
    Wait AI[0]>3.0
    Set DO[3]=On
    If analog_in[0]<4
      MoveJ Waypoint_1
      MoveJ Waypoint_2
      MoveJ Waypoint_3
      MoveJ Waypoint_4
      MoveJ Waypoint_5
      MoveJ Waypoint_6
      Set DO[3]=Off
    ElseIf analog_in[0]>4
      MoveJ Waypoint_1
      MoveJ Waypoint_7
      MoveJ Waypoint_8
      Set DO[2]=On
      MoveJ Waypoint_9
      MoveJ Waypoint_10
      Set DO[1]=On
      Set DO[3]=Off
```

KUVA 22. UR10 pilotin robottiohjelma.

Robottiohjelma on toiminnaltaan hyvin vastaava IRB 140:n toiminnallisuutta. Aluksi asetetaan viestinnän ohjaussignaalit FALSE arvoihin ja odotetaan vietin saapumista. Viestin saavuttua sisällön mukaisesti suoritetaan liikkeet ja toisen liikkeen jälkeen lähetetään viesti IRB 140:lle.

4.6 Pilotin tulokset

Pilotissa ei päästy luvun 4.1 mukaiseen tilanteeseen asti. Kaikki käyttötapauksen osat ovat rakennettu, eli luvun 4.1 mukaisen järjestelmän osat ovat valmiita. LabVIEW-ohjelmistossa kuitenkin esiintyvien ongelmien takia koko järjestelmää ei saatu toimimaan. Osia saatiin testailtua yksittäin, mutta luvun 3.6 tyylisiä mittauksia ei saatu ongelmien ratkomiseen kuluneen ajan takia myöskään suoritettua.

IoT Gateway -osuutta testattiin luvun 4.3 mukaisilla ohjelmilla. IoT Gateway viestintä toimii ja kyseisessä käyttötapauksessa viestine menettämine QOS 1 -tason RapdiTrigger-tapahtumista oli erittäin pieni, muutamia Timer Trigger -tapahtumia katosi testin aikana, mutta ei liikaa datankeräystarkoituksiin. Syy vähäiselle viestien menettämiselle on Trigger-tapahtumien välisen ajan olevan niin suuri. Muodostettua järjestelmää pystyttäisiin käyttämään tällaista käyttötarkoitusta vastaavissa tilanteissa ilman ongelmia.

LabVIEW-ohjelmassa syntyi paljon ongelmia. Kaikki ohjelman osat toimivat yksittäisinä. Viestien käsittely toimii ongelmitta ja pystyy käsittelemään kaikkia formaatin mukaisia käskyjä. Tietysti ohjelmasta puuttuu virheellisten käskyjen käsittely lukemisvaiheessa, mutta väärin luettujen viestien vastaanottaminen on huomioitu ”MQTT_output_Write.vi”-ohjelman Case Structureiden avulla. Ongelma syntyy siinä, että asetetut NI myRIO-1900:n I/O-arvot eivät pysy viestissä asetetuissa arvoissa. Arvot kuitenkin asetetaan onnistuneesti, koska robottiohjelmat reagoivat saapuviin viesteihin, mutta ylimääräiset ohjauksen ulkopuoliset muutokset eivät pysy päällä tarpeeksi kauaa, että ne tunnistettaisiin. Ongelma voi johtua mahdollisesti asettamisessa tapahtuvista virheistä, taikka ongelmista ulostulojen nollaamiseen käytettävän IRQ-toiminnon kanssa.

IRQ-pohjaiset toiminnot myös toimivat yksittäisinä tapauksina. Manuaalisesti asetettujen ulostulojen nollaaminen onnistuu ongelmitta siihen tehdyn ohjelman avulla. Publisher-toiminnallisuus toimii funktionaalisesti. Asetetut I/O-pinnit luetaan ja viesti muodostetaan oikein. Ongelma syntyy silloin, kun IRQ-tapahtumia suoritetaan luvussa 4.4 esitetyn ohjelman rungossa. Viestejä voidaan lähettää vain yksi, joka johtaa ohjelman virhetilaan.

Subscribe-toiminnot käyttäytyvät hyvin samanlailla, kuin Publish-toiminto siinä, että viestejä voidaan onnistuneesti vastaanottaa vain yksi, jonka jälkeen ohjelma joudutaan uudelleenkäynnistämään. Lisäksi ohjelma ei pysy yhdistyneenä MQTT Brokkerille asetetun keep alive time -arvon mukaisesti. Keep alive time tarkoitaa sitä aikaa, jonka laite pitää yhteyden Brokeriin toiminnassa ilman viestine lähetys-/vastaanottamistapahtumaa. Tämä aika oli testeissä asetettu 3000 sekuntiin eli 50 minuuttiin. Todellisuudessa yhteys pysyy vain vähän yli minuutin toiminnassa, kunnes ohjelmassa syntyy Error virhekoodilta 66. Tämä virhekoodi kuvaa ”Connection was closed by peer” virhettä. Tästä syystä epäiltävästi ongelma voi johtua käytetystä yhteydestä Brokkerille. Broker sijaitsee TAMK:n tietoturvallisesti tiukemmassa TiTe-verkossa, ja aikaisemmin on ollut ongelmia rajoitettujen yhteystapahtumien kanssa. Tämän takia voi olla mahdollista, että tietoturva-asetukset estävät ohjelman kunnollisen toiminnan.

5 POHDINTA

Työn tutkimustehtävässä suoritettujen mittauksien perusteella saatiin Timer- ja RapidVariable-triggereiden toiminnallisuus ja yhteisvaikutuksista hyvä kuva. Timer Trigger pystyy noin 9 Hz (110 ms viestien lähetysväli) luotettavaan tiedonsiirtoon ja RapidVariable trigger vaatii enemmän aikaa lähetystapahtumien välillä, noin 2,5–2,9 Hz (n.350–400 ms viestien lähetysväli) luotettavaan toimintaan. Ohjelman suurimmaksi ongelmaksi paljastui päällekkäisten viestien kanssa toimiminen. Ohjelmassa aiheutuu suuri määrä lähettämättömiä viestejä, jos viestitapahtumat ovat yleisiä. Hitaahkon viestinnän takia IoT Gateway -ohjelman käytettävyys nopeaa viestintää vaativissa käyttötarkoituksissa heikkenee tai on jopa mahdotonta.

NI myRIO-1900 -laitteen ja LVMQTT-kirjaston toimintaan kohdistuneista mittauksista saatiin myös positiivisia tuloksia. Saatiin selkeitä havaintoja siitä, miten ohjelma voi vaikuttaa toimintaan. Ilman rajoitteita viestinlähetystaajuudeksi saatiin 2,26 kHz (0,442 ms) ja viestinvastaanottotaajuudeksi 1,49 kHz (0,673 ms). Hitaammalla ohjelmalla saatiin selville, että käytetyllä ohjelmalla pystytään ainakin noin 100–300 Hz alueella toimimaan ongelmitta.

IoT Gatewayllä tehdyt mittaukset sisältävät vähemmän oletuksia ja siten niiden voidaan sanoa olevan luotettavampia, kuin NI myRIO-1900 -mittaukset. Toisaalta osissa mittauksissa IoT Gateway -tarkastelu oli huomattavasti pienempi otanta, verrattuna NI myRIO-1900 -mittauksiin. NI myRIO-1900 -mittauksissa jouduttiin tekemään enemmän oletuksia ja ne sisälsivät enemmän tuntemattomia muuttujia, esimerkiksi kellon toiminta on vieläkin odottamatonta ja itse laitteen toiminnan tasaisuudesta ei voida saada varmuutta. Näistä seikoista huolimatta, kerättyä dataa pystytään käyttämään vähintään vertailukohteena, ja tämä olikin työn tarkoitus. Vaikka kerätyt arvot eivät ole todistettavasti absoluuttisia ja sisältävät mahdollisia virheitä, on työn osa silti onnistunut.

LabVIEW-ohjelmistokehitys LVMQTT-kirjastolla on osittain onnistunut. On saatu luotua todistettavasti toimivat osat, jotka mahdollistavat halutut toiminnot ja mit-

tauksien mukaisesti pitäisi täyttää hyvin tarvittavat toiminnallisuuden kriteerit. Toisaalta valmista kokonaisuutta ei saatu toimintakuntoon. Ohjelmassa ja mahdollisesti toimiympäristössä on esteitä, joita projektin aikataulun sisällä ei voitu ratkoa. Tämä osio voidaan silti laskea ainakin osittain onnistuneeksi. Alkuperäisesti valmis järjestelmä oli aina ideaalinen tavoite, ja odotettu tuotos oli tarkoitettu pohjaksi, jota jatkokehitettäisiin. Tuotoksena on pohja, joka on todettu osiltaan toimivaksi ja siinä on selkeät kohdat, joissa suorittaa jatkokehitystä. Jos ohjelma saataisiin toimimaan, sen käyttömahdollisuudet ovat suuremmat, kuin IoT Gatewayn. Pääasiallisesti nopeamman viestintäkapasiteetin seurauksena. Alhaiset jännitetasot signaaleissa voivat kyllä tuottaa lisää ongelmia.

NI myRIO-1900:n käyttämistä viestintälaitteena ei tämän työn perusteella voida suoraan kumota tai hyväksyä, johtuen siitä, että valmista järjestelmää ei koskaan saatu valmiiksi. Laitteessa on omat hyvät ja huonot puolensa. Hyvinä puolina on se, että ohjelmallisesti lähetettävälle datalle voitaisiin tehdä moniakin operaatioita, esimerkiksi kääntää viestintänumeroita suoraan volteiksi jne. Huonoina puolina on kuitenkin laitteen rajatut I/O-liitännät ja liitäntöjen alhaiset jännitetasot, jotka eivät suoraan sovi kaikkiin laitteisiin.

Järjestelmässä on sanotusti selkeitä jatkokehityskohteita. IoT Gateway -ohjelmassa on vielä tutkimattomia toimintoja ja samanaikaisten viestien interaktion hallitseminen on vielä suurimmaksi osaksi tuntematonta. Eniten jatkokehitystä vaativa osa valmiissa laitteistossa on kuitenkin LabVIEW-ohjelma. Ohjelmassa ensimmäiset jatkokehitysaskeleet ovat tutkia, mikä aiheuttaa yhteyden katkeamisen ja miksi viestitapahtuma katkaisee yhteyden. Nämä voivat mahdollisesti olla saman esteen aiheuttamia. Lisäksi ohjelman konfiguraatiotiedot olisi hyvä siirtää tiedostosta luettavaksi. On paljon nopeampaa luoda uusi esimerkiksi XML-tiedosto, kuin muokata ja luoda uusi versio ohjelmasta.

Tietysti toinen jatkokehityssuunta on muiden robottien integroiminen järjestelmään. FieldLab-tilassa on esimerkiksi kaksi Omron LD-90 -mobiilirobottia, joille on olemassa kaupallinen valmis MQTT-toimintopaketti. Lisäksi TAMK:lla on kehitetty python ohjelma, jolla kerätään dataa näiltä roboteilta ja lähetetään se tietokantaan MQTT-protokollalta. Tämä voitaisiin luultavasti kehittää myös roboteille suuntautuvaksi ohjelmaksi.

LÄHTEET

Gilchrist, A. 2016. Industry 4.0 : The Industrial Internet of Things. E-kirja. Uud. painos. New York: Apress. Viitattu 25.04.2022. Vaatii käyttöoikeuden. https://ebookcentral.proquest.com/lib/tampere/detail.action?docID=4573237.

MQTT Specifications. n.d. MQTT.org. Verkkosivu. Viitattu 23.04.2022 https://mqtt.org/mqtt-specification/

Banks, A., Gupta, R., MQTT Version 3.1.1. 2014. Pdf-dokumentti. Viitattu 23.04.2022. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf

Banks, A., Briggs, E., Borgendale, K., Gupta, R., MQTT Version 5.0. 2019. Pdf-dokumentti. Viitattu 23.04.2022. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf

IRB 140. 2021. ABB. Pdf-dokumentti. Viitattu 23.04.2022. https://search.abb.com/library/Download.aspx?DocumentID=PR10031EN_R15_HR&LanguageCode=en&DocumentPartId=&Action=Launch

Bus Terminals. n.d. Beckhoff. Verkkosivu. Viitattu 23.04.2022. https://www.beckhoff.com/en-en/products/i-o/bus-terminals/tabular-product-overview/

DeviceNet®. n.d. ODVA. Verkkosivu. Viitattu 23.04.2022. https://www.odva.org/technology-standards/key-technologies/devicenet/

IRB 4600. 2021. ABB. Pdf-dokumentti. Viitattu 23.04.2022. https://search.abb.com/library/Download.aspx?DocumentID=ROB0109EN_G&LanguageCode=en&DocumentPartId=&Action=Launch

Application manual - IoT Gateway. 2022. ABB. Pdf-dokumentti. Viitattu 23.04.2022. https://robotapps.blob.core.windows.net/apps/ApplicationManual%20IoT%20Gateway%20v1.2.pdf

UR10e. n.d . Universal Robots. Verkkosivu. Viitattu 23.04.2022 https://www.universal-robots.com/products/ur10-robot/

User manual UR10/CB3. n.d. Universal Robots. Verkkosivu. Viitattu 23.04.2022. https://automationdistribution.com/content/Universal-Robots-UR10-User-Manual.pdf

National Instruments. 2022. From Student to Engineer: Preparing Future Innovators With the NI LabVIEW RIO Architecture. Verkkosivu. Viitattu 21.04.2022. https://www.ni.com/fi-fi/innovations/white-papers/14/from-student-to-engineer--preparing-future-innovators-with-the-n.html

User guide and specifications NI myRIO-1900. n.d. National Instruments. Pdf-dokumentti. Viitattu 21.04.2022. https://www.ni.com/pdf/manuals/376047c.pdf

LabVIEW Environment Basics. n.d. National Instruments. Verkkosivu. Viitattu 29.04.2022 https://www.ni.com/getting-started/labview-basics/environment

Meaning of Different Wire Colors in LabVIEW. 2019. National Instruments. Verkkosivu. Viitattu 29.04.2022. https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019LsVSAU&l=fi-FI

Graphical Programming. n.d. National Instruments. Verkkosivu. Viitattu 29.04.2022 https://www.ni.com/getting-started/labview-basics/dataflow

Execution Structures in LabVIEW. n.d. National Instruments. Verkkosivu. Viitattu 29.04.2022 https://www.ni.com/getting-started/labview-basics/execution-structures

MQTT Protocol in LabVIEW. 2020. National Instruments. Verkkosivu. Viitattu 29.04.2022. https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000kKAcSAM&l=fi-FI

LVMQTT. 2017. DAQIO. Verkkosivu. Viitattu 29.04.2022. https://github.com/DAQIO/LVMQTT

Silva, D., Carvalho, L., Soares, J., Sofia, R. A performance analysis of internet of things networking protocols: Evaluating MQTT, CoAP, OPC UA. Applied sciences 2021-06-01, Vol.11(11), p.4879. Viitattu 30.04.2022. https://www-proquest-com.libproxy.tuni.fi/docview/2635419621?pq-origsite=primo&accountid=14242

Pub/Sub: A Google-Scale Messaging Service. Google Cloud. n.d. Viitattu 30.04.2022. https://cloud.google.com/pubsub/architecture#judging_performance_of_a_messaging_service

LIITTEET

Liite 1. MQTT Control Packet types

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgement
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish Received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

“Represented as a 4-bit unsigned value.” (Banks & Gupta 2020, 16-17)

Liite 2. MQTT Flag Bits

“The remaining bits [3-0] of byte 1 in the fixed header contain flags specific to each MQTT Control Packet 243 type”. (Banks & Gupta 2020, 17-18)

Name	Fixed header flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	
PUBLISH	Used in MQTT 3.1.1	DUP	QoS	QoS	Retain
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0

Liite 3. IoT Gateway Default tiedostot

Seuraavassa kuvassa on IoT Gatewayn ohjaimelle automaattisesti luotu Controller tyypin tiedosto. Tiedosto on avattu VScode ohjelmalla. Tiedosto on pakko olla tallennettuna ”%ProgramData%\ABB\IoT Gateway\MQTT\Controllers” tiedostopolun mukaiseen kansioon.

MQTT > Controllers > default.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RobotController xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <TriggerFiles>
    <string>ExampleTrigger1.xml</string>
  </TriggerFiles>
  <UserData>
    <item>
      <key>AssetID</key>
      <value>Robot123456</value>
    </item>
    <item>
      <key>PurchaseDate</key>
      <value>2021.04.15</value>
    </item>
  </UserData>
</RobotController>
```

<TriggerFiles> XML tagin väliin voidaan lisätä haluttu määrä Trigger tiedostojen nimiä samalla tavalla, kuin siinä jo oleva tiedostonimi. <UserData> tagiin voidaan laittaa ei lähetettävää tietoa kyseisestä Aliaksesta.

Seuraavassa kuvassa on automaattisesti luotu Trigger tyypin tiedosto. Tiedosto on pakko olla tallennettuna "%ProgramData%\ABB\IoT Gateway\MQTT\Triggers” tiedostopolun mukaiseen kansioon.

MQTT > Triggers > ExampleTrigger1.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<Trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Type>Timer</Type>
  <Parameters>1</Parameters>
  <GuardCondition>true</GuardCondition>
  <Topic>SmallTest</Topic>
  <Payload>ExamplePayload.hbs</Payload>
  <QoS>0</QoS>
  <Retain>true</Retain>
</Trigger>

```

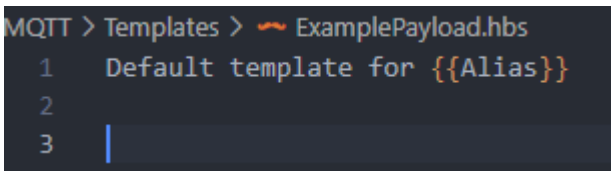
<Type> XML tagin sisällä määritetään lähetystapahtuman tyyppi, jotka esitetään seuraavassa kuvassa. <Parameter> tagiin kirjoitetaan lähetystapariippuvaisesti tarkasteltava parametri esimerkiksi ajastimen pituus sekunneissa, Rapid muuttuja, IO signaali. <GuardCondition> on lisätarkastelu, jonka pitää olla boolean TRUE, jotta viesti lähetetään.

(jatkuu)

<Topic> on viestin määränpään Topic. <Payload> kertoo minkä tiedoston sisältö lähetetään viestin rakenteen Payload osassa. <QoS> kertoo viestin QoS asetuksen tason. <Retain> kertoo, säilytetäänkö viesti Brokerilla.

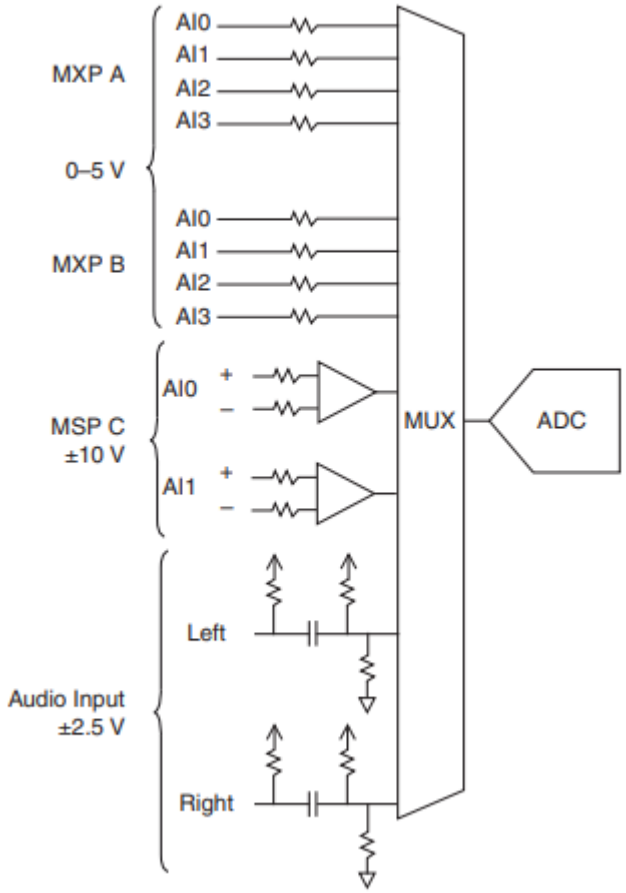
Polled
Timer
ControllerState
ControllerConnected
ControllerDisconnected
OperatingMode
ProgramState
IOSignal
RapidVariable
EventLog

Seuraavassa kuvassa on esiteltynä automaattisesti luotu Templates tyypin tiedosto. Viesti sisältö on itse määritettävissä. Tiedosto on pakko olla tallennettuna ”%ProgramData%\ABB\IoT Gateway\MQTT\Templates” tiedostopolun mukaiseen kansioon.



Liite 4. NI myRIO analogi sisääntulon piirikaavio

Figure 5. NI myRIO-1900 Analog Input Circuitry



(User guide and specifications NI myRIO-1900 n.d, 8)

Liite 4. NI myRIO digitaali I/O piirikaavio

Figure 7. DIO Lines <13..0> on MXP Connector A or B

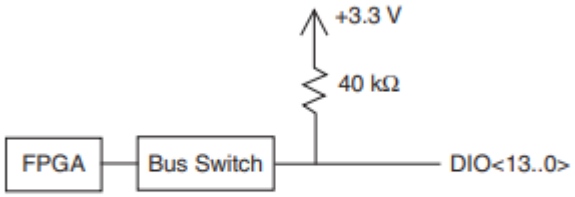


Figure 8. DIO Lines <15..14> on MXP Connector A or B

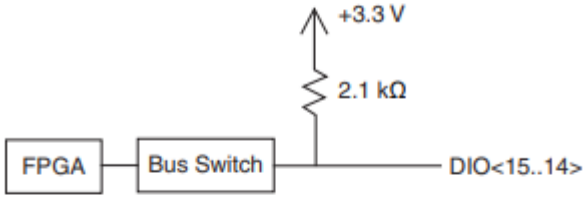
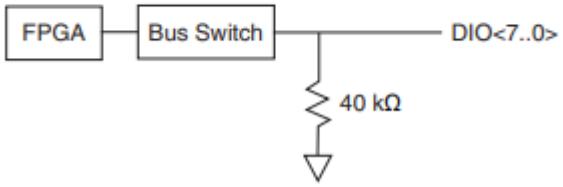


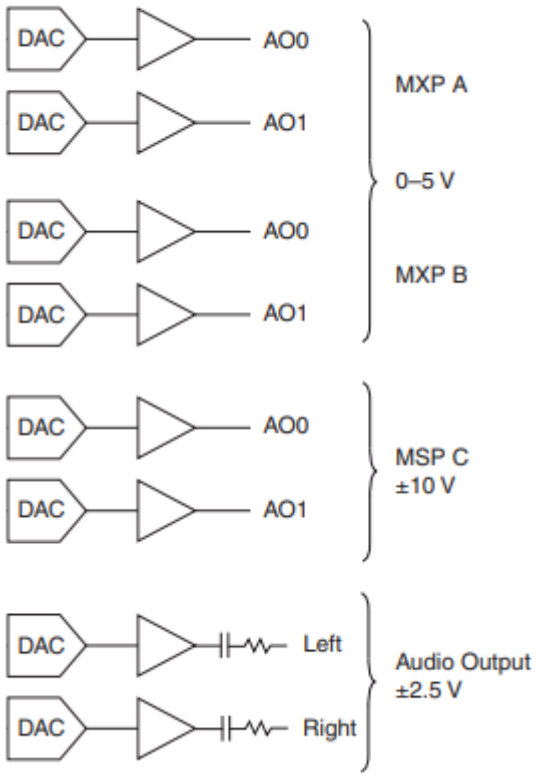
Figure 9. DIO Lines <7..0> on MSP Connector C



(User guide and specifications NI myRIO-1900 n.d, 11)

Liite 5. NI myRIO analogi ulostulon piirikaavio

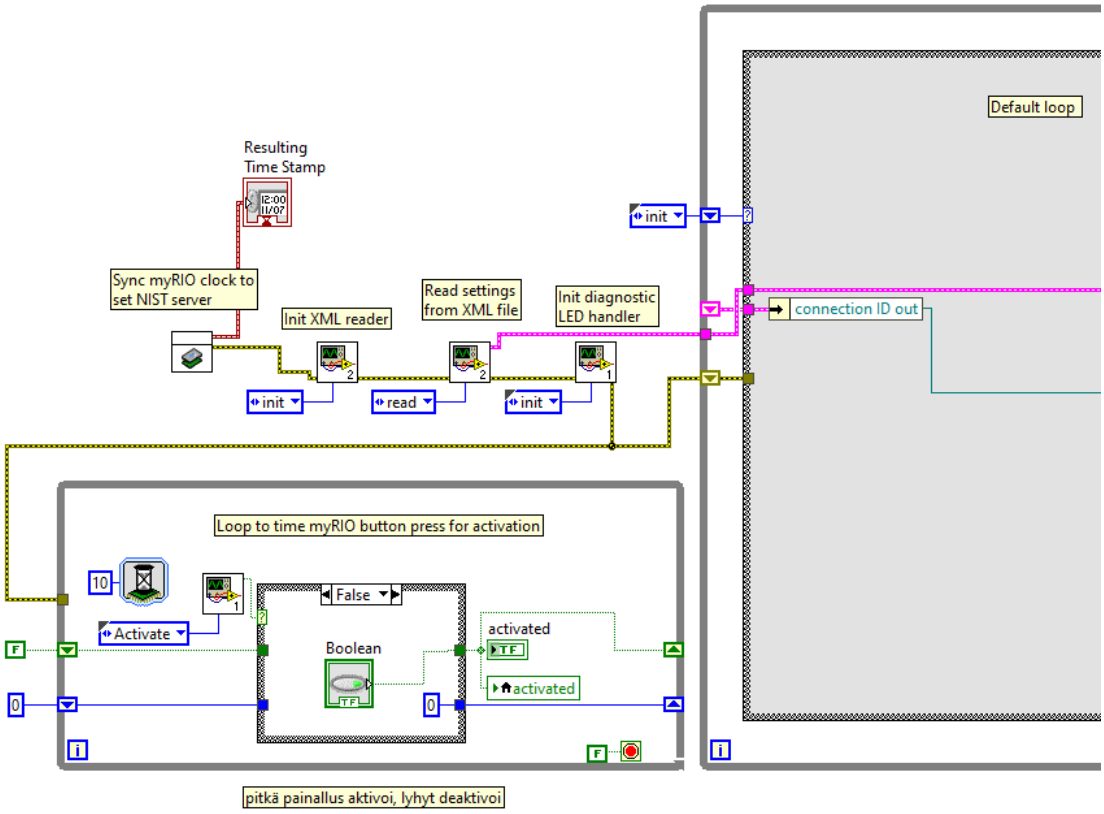
Figure 6. NI myRIO-1900 Analog Output Circuitry



(User guide and specifications NI myRIO-1900 n.d, 9)

Liite 6. Datankeruujärjestelmän LabVIEW ohjelma

Ensimmäinen osa ohjelmaa



Tietokantaan ohjattavien viestien JSON formaatin esittäminen

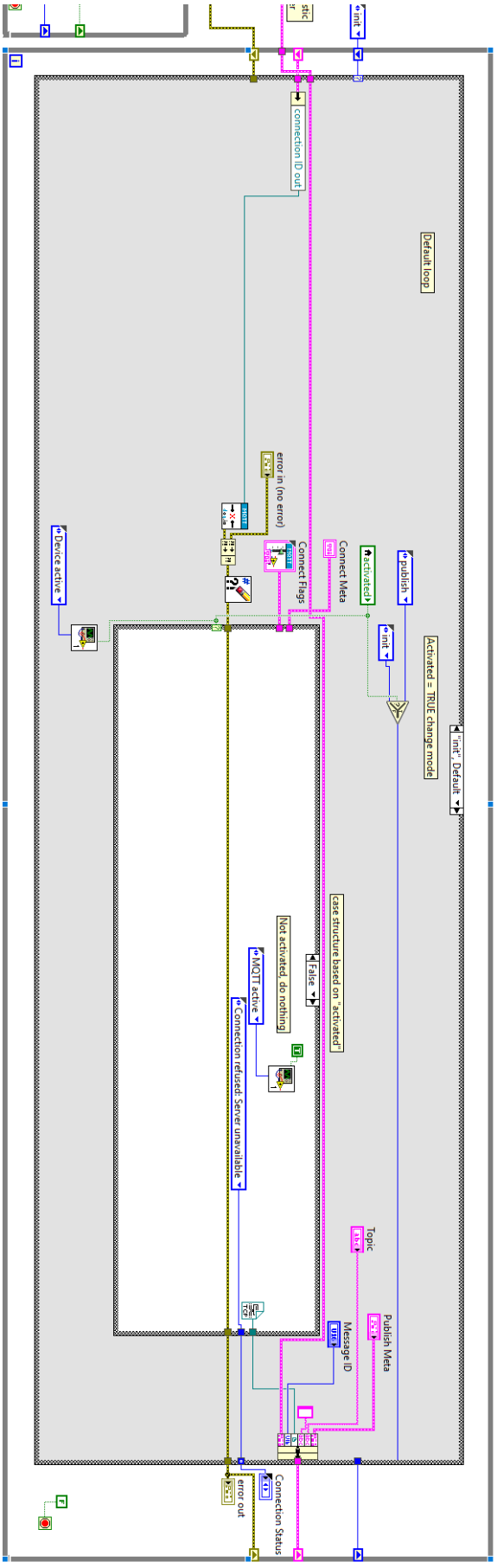
Fieldlab Database data structure 14.12.2020 / KN

Data type: Data format Postgres
City: ASCII text city / text
Place: ASCII text place/text
Machine: ASCII text machine/text
Machine Type ASCII text machine_type/text
Machine order number Integer machine_number/numeric
Machine module: ASCII text machine_module/text
Measurement name: ASCII text meas_name/text
Signal name: ASCII text signal_name/text
Signal order number: integer signal_number/numeric
Signal value: ASCII signal_value/text
Signal unit ASCII text signal_unit/text
Signal value type NUM / ASC signal_value_type/text
Time Stamp Time with ms time_stamp/timestamp without time zone

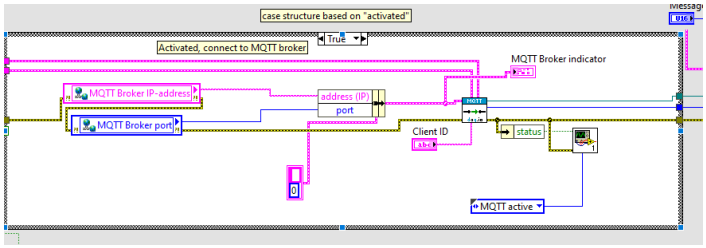
Example1:
City Tampere
Place TAMK F0-29
Machine ABBrobot
MachineTyp IRP2001
MachineNum 2
MachineMod Arm1
Measurement name Printing test 3
SignalName torque
SignalNum 3
SignalVal 24.78
SignalUnit Nm
TimeStamp 2020-02-24 15:12:34.524
SignalTyp NUM

(jatkuu)

Ohjelman primäärinen, jos case = Init ja activated = False



Primäärinen loop, jos activated = True



Primäärinen loop, jos case = Publish