Mikko Rajala

PILVIPALVELUN JA TEOLLISUUDEN OHJAUSJÄRJESTELMÄN VÄLISEN RAJAPINNAN TOTEUTUS

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2017

TIIVISTELMÄ

Mikko, Rajala
Pilvipalvelun ja teollisuuden ohjausjärjestelmän välisen monitorointirajapinnan toteutus
Jyväskylä: Jyväskylän yliopisto, 2017, 49 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja: Veijalainen, Jari

Teollisuuden ohjaus- ja valvontajärjestelmät ovat toimineet pitkään tehdasprosessien sisäisinä itsenäisinä ja suljettuina järjestelminä. Nykypäivän trendi on, että näistä automaatiojärjestelmistä (Process Automation System) ja niiden ohjaamista prosesseista haluttaisiin kerätä tietoa keskitetysti verkon yli. Aikaisemmin tämä tiedonkeruu on tapahtunut järjestelmän sisällä samassa toimitilassa.

Tiedonkeruun toteutukseen tarvittaisiin rajapinta (Interface), joka toimisi pilvipalvelun (Cloud service) ja teollisuudenjärjestelmän välillä. Rajapinta mahdollistaisi monenlaisten tietojen keruun teollisuudenjärjestelmistä keskitetysti pilvipalveluun. Tulevaisuudessa ja jopa nykypäivänä tiedon kerääminen on yhä tärkeämmässä roolissa. Teollisuuden ohjausjärjestelmien keräämistä prosessien historiatiedoista voidaan kaivaa ulos aivan uutta tietoa. Keräämällä useiden teollisuudenprosessien toimintahistoria yhteen paikkaan voidaan luoda aivan uutta tietoa näistä prosesseista ja jopa ennustaa prosessien toimintaa sekä ennakoida mahdollisia tulevia virhetiloja.

Automaatio- ja teollisuudenjärjestelmien muutokset ja nykyaikaistaminen herättivät kiinnostukseni tähän tutkimusaiheeseen. Tutkimus toteutettiin kirjallisuuskatsauksena ja konstruktiivisena tutkimuksena. Tutkimuksessa haettiin vastausta kysymykseen, miten teollisuuden ohjausjärjestelmän ja pilvipalvelun välinen tiedonkeruurajapinta voidaan toteuttaa TCP/IP- ja HTTP – protokollaa käyttäen. Kirjallisuuden avulla selvitettiin, että miten aihetta on tähän mennessä tutkittu ja minkälaisia tutkimustuloksia sekä artikkeleita aiheesta löytyy. Konstruktiivisessa osuudessa taas rakennettiin toteutus, jolla mahdollistettiin tiedonkeruun toteutus automaatiojärjestelmästä TCP/IP- ja HTTP –protokollaa käyttäen.

Rajapintatoteutuksen perusteella pystyttiin esittämään, että TCP/IP ja HTTP –protokollaa käyttäen voidaan toteuttaa rajapinta automaatiojärjestelmän ja pilvipalvelun välille.

Asiasanat: teollisuusautomaatio, teollisuuden ohjausjärjestelmät, pilvipalvelu, pilvilaskenta, rajapinta, tcp/ip, http

ABSTRACT

Mikko, Rajala
Realization of a remote monitoring interface between industrial controlling system and cloud service.
Jyväskylä: University of Jyväskylä, 2017, 49 p.
Information Systems, Master's Thesis
Supervisor: Veijalainen, Jari

Industrial control and monitoring systems have been operated independently and isolated in internal manufacturing processes for a long time. The current trend is that data is wanted to be collected centrally over the network from process automation systems and processes that they control. Previously this data has been collected within the process automation system in the same process space.

An interface is needed to implement the collection of data. This interface is operated between the cloud and the industrial system. Information could be collected to cloud service via interface. In future and even today data collection is playing more and more important role. Completely new information can be discovered from the collected history data of industrial control systems. By collecting many operation histories of industrial controlling systems in one place, completely new information of these processes can be created. Even the operation of processes or possible future error conditions can be predicted of the collected data.

Changes and modernization of automation and industrial systems got me interested in this research topic. The study was carried out by a literature review and constructive research. Research was designed to answer the question, how the data collection interface between industrial controlling system and cloud service can be implemented by using TCP/IP and HTTP protocols. By literature review it was investigated how the topic was studied so far and what kinds of researches and articles from this topic can be found. In constructive part realization of data collection interface was build. This realization enables the data collection interface to automation system by using TCP/IP and HTTP protocols.

The implementation makes utilization of TCP/IP and http protocols plausible in this context.

Keywords: process automation, industrial control systems, cloud computing, cloud service, interface, tcp/ip, http

KUVIOT

SISÄLLYS

Käytetty termistö

Asiakas-palvelin malli (Client-Server Model)

Amerikan yhdysvaltojen kauppaministeriön kansallinen standardoinnin ja teknologian laitos (US National Institute of Standards and Technology (NIST))

Automaatiojärjestelmä (Process Automation System)

DCOM – objektiprotokolla (Distributed Component Object Model Protocol)

Hajautettu tietojenkäsittely (Distributed Computing)

HTTP (Hyper Text Transfer Protocol)

Infrastruktuuri palveluna (Infrastructure as a Service, IaaS)

ISA (International Society of Automation)

ISA-95 ---standardi (mikä se nimi on suomeksi?)

I/O-laitteet (Input/Output Devices)

Kanava (Channel)

Kääreohjelma (Wrapper)

Langattomat anturiverkot (Wireless Sensor Networks,WSN)

Laite (Device)

Ohjausyksikkö/ohjelmoitavalogiikka (Programmable Logic Contoller)

Objektien linkittämis- ja upottamistoiminto (Object Linking and Embedding, OLE)

OPC (OLE for Process Control)

OPC DA (OLE for Process Control - Data Access)

OPC UA (OLE for Process Control -Unified Architecture)

Palvelin (Server)

Palvelukeskeisen arkkitehtuuri (Service-Oriented Architecture, SOA)

Pilvipalvelu (Cloud service)

Pilvilaskenta (Cloud computing)

Pyyntö (Request)

Rajapinta (Interface)

Robottiohjausyksiköt (Controllers of Industrial Robots)

SCADA (Supervisory Control and Data Acquisition)

Verkkosovelluspavelu (Software as a Service, SaaS)

Sovellusalusta palveluna (Platform as a service, PaaS)

Tarvelaskenta (Utility Computing)

TCP/IP (Transmission Control Protocol / Internet Protocol)

Internet-protokolla (Internet Protocol, IP)

TCP-yhteyskäytäntö

Teollisuudenjärjestelmä (Idustrial System)

Teollisuusverkko (Industrial network)

Toiminnanohjausjärjestelmä (Enterprise Resource Planning System, ERP system)

Tunniste (Tag)

Tyyppi (Type)

Tuotannonohjaus (Manufacturing Execution System, MES)

Vastaus (Response)

Verkkoasiakas (Client)

verkostoituminen (Networking)

www-sovelluspalvelu (Web service)

1 Johdanto

1.1 Tutkimuksen taustat ja motiivit

Teollisuuden ohjaus- ja valvontajärjestelmät ovat pitkään olleet täysin suljettuja, eikä niihin ole ollut ulkopuolisia yhteyksiä. Nämä automaatiojärjestelmät (Process Automation System) ovat siis olleet erilaisissa prosessien valvonta- ja ohjaustarkoituksessa täysin itsenäisiä ja suljettuja järjestelmiä, joille on luotu omat standardinsa. Nykypäivänä näistä järjestelmistä haluttaisiin kerätä tietoa keskitetysti verkon yli tai jopa ohjata tätä prosessia järjestelmän ulkopuolelta verkon yli. Aikaisemmin nämä asiat ovat tapahtuneet järjestelmän sisällä samassa toimitilassa, jossa järjestelmä sijaitsee ja tässä suljetussa ohjausverkossa on tapahtunut kaikki tiedonkeruu sekä järjestelmän ohjaus.

Jotta tietoa voidaan kerätä teollisuudenjärjestelmästä (Idustrial System), täytyy siihen luoda rajapinta, jonka kautta yhteys teollisuudenjärjestelmän sisälle pystytään luomaan. Näitä rajapintoja tarvitaan, oli tarkoitus sitten kerätä tietoa tai ohjata tätä järjestelmää pilvipalvelun kautta. Haasteita tähän luovat teollisuuden erilaiset standardit, jotka määrittelevät kuinka automaatiojärjestelmään ja sen osajärjestelmiin saa olla yhteydessä ja miten tieto järjestelmän sisällä siirtyy. Käytännössä järjestelmän toimintasäännöt määritellään näiden kunkin teollisuuden alan omien standardien mukaan. Koska tietoturvan ja monien standardien puolesta järjestelmän ohjauksien tuonti teollisuusjärjestelmään sen ulkopuolelta on liian vaarallista, keskitytään tässä tutkimuksessa tiedon keräämiseen teollisuudenjärjestelmästä ja sen prosesseista ulospäin keskitetysti verkon yli pilvipalveluun, jossa sitä voidaan jatkojalostaa eri tarkoituksiin.

Nykyään tiedon kerääminen on tullut yhä tärkeämpään rooliin. Erilaisten automaatiojärjestelmien ohjaamien prosessien tallennetuista historiatiedoista voidaan kaivaa ulos aivan uutta tietoa. Usean teollisuudenprosessien toimintahistorian yhdistäminen ja tämän tiedon käsittely voivat luoda aivan uutta tietoa näistä prosesseista. Samalla saadaan myös keskitettyä valvontaa ja voidaan vertailla näitä prosesseja. Tätä kautta voidaan myös ennustaa proses-

sien toimintaa ja ennakoida mahdollisia virhetiloja tai tulevia keskeytyksiä näissä prosesseissa sekä saada jatkuvuutta niihin. Prosessin jatkuvuus saataisiin näin kerätyn tiedon pohjalta optimoitua mahdollisimman hyväksi, laskettua virhemahdollisuuksia ja löytää prosessiin vaikuttavat uhat.

Käsiteltävä asia on tuore eikä se ole ollut vielä laajasti käytössä teollisuuden järjestelmissä, mutta siitä on kirjoitettu alan lehdissä. Automaatiojärjestelmään luotava rajapinta toteutus ja tiedonkeruu tämän kautta ei ole siis vielä laajemmin käytössä. Aikaisempaa tutkimusaineistoa löytyy langattomien anturiverkkojen (Wireless Sensor Networks) yhdistämisestä pilvipalveluihin (Cloud Services), palvelukeskeisen arkkitehtuurin (Service-Oriented Architecture) käyttämisestä automaatiojärjestelmän liittämisessä pilvipalveluun ja automaatiojärjestelmien ja niiden teollisuusverkkoihin (Industrial Networks) luotavien uusien yhteyksien tietoturvauhkista.

Kirjallisuuskatsauksessa käydään läpi näitä aikaisempia aiheesta tehtyjä tutkimuksia ja verrataan miten niitä voidaan yhdistää tähän tutkimukseen ja sen taustoihin. Kirjallisuuskatsauksessa vertaillaan sitä mitä yhteistä näissä tutkimuksissa on ja minkälaisia eroavaisuuksia niistä löytyy.

Kirjallisuuskatsaus on jaettu kolmeen osioon, jotka ovat langattomat anturiverkot, niiden tiedonkeruu ja palvelukeskeisen arkkitehtuurin yhdistäminen automaatiojärjestelmään sekä automaatiojärjestelmien tietoturva.

Tutkimuksen konstruktiivisessa osuudessa keskitytään automaatiojärjestelmän toteutettavan rajapinnan (Interface) luomiseen TCP/IP ja HTTP -protokollaa käyttäen. Tässä osuudessa on tarkoitus selvittää, että miten tämä rajapinta tulisi parhaiten toteuttaa.

1.2 Keskeisten käsitteiden määrittely

1.2.1 Automaatiojärjestelmä

Automaatiojärjestelmät ovat yleensä teollisuusprosessien valvonta- ja ohjaustarkoitukseen toteutettuja teollisuuden järjestelmiä. Automaatiojärjestelmillä tarkoitetaan järjestelmiä, jotka toimivat prosessikohtaisissa verkoissa, käyttävät prosessikohtaisia laitteita eivätkä ole yhteydessä Internetiin. (Byres, 2013). Nämä teollisuusverkot siis toimivat omina erillisinä ohjausjärjestelminään ilman yhteyksiä ulkoisiin tiedonsiirtoverkkoihin tai palvelimiin. Teollisuusverkko (Industrial Network) on datan siirtämiseen teollisuuslaitoksissa tarkoitettu langallinen tai langaton verkko. Se tukee isojen prosessikokonaisuuksien ohjauksia sekä yhdistää tiedonkulun prosessin osakokonaisuuksien välillä.

Prosessien valvonta- ja ohjausjärjestelmät pitävät sisällään suuren määrän erilaisia komponentteja ja osajärjestelmiä, jotka toimivat järjestelmän eri tasoilla.

Prosessi sisältää erilaisia hajautettuja valvonta- , prosessinohjaus- ja tuotantojärjestelmiä.(Jehertova, 2013.) Prosessilla tarkoitetaan teollisuuden prosessia, jonka ohjaamiseen automaatiojärjestelmä on kehitetty. Kokonaisuudessaan prosessissa voi olla monia aliprosesseja, joita ohjataan omien ohjausyksiköiden avulla ja nämä ohjausyksiköt taas keskustelevat toisten prosessien ohjausyksiköiden kanssa teollisuusverkkoa hyödyntäen.

1.2.2 Pilvipalvelu

Pilvipalvelut on toimintamalli, joka mahdollistaa pääsyn vapaasti konfiguroitaviin ja skaalautuviin tietotekniikkaresursseihin, jotka voidaan ottaa käyttöön tai poistaa käytöstä helposti ja nopeasti. Amerikan yhdysvaltojen kauppaministeriön kansallinen standardoinnin ja teknologian laitos (US National Institute of Standards and Technology (NIST)) määrittelee viisi ominaispiirrettä pilvipalvelulle: nopea joustavuus, resurssien yhteiskäyttö, itsepalvelullisuus, päätelaiteriippumattomuus ja tarkka resurssien käyttö ja valvonta. (Mell & Grance, 2011.)

NIST jaottelee pilvipalvelut kolmeen erilaiseen palvelumalliin, jotka ovat verkkosovelluspalvelu (SaaS), sovellusalusta palveluna (PaaS) ja infrastruktuuri palveluna (IaaS). Näiden lisäksi on myös neljä käyttöönottomallia (Deployment Model): yksityinen pilvi (Private Cloud), julkinen pilvi (Public Cloud), yhteisöllinen pilvi (Community Cloud) ja hybridipilvi (Hybrid Cloud). (Mell & Grance, 2011.)

Pilvipalvelut perustuvat virtuaalisiin resursseihin. Pilvipalvelu on suhteellisen uusi termi ja perustuu siis virtualisointiin, hajautettuun tietojenkäsittelyyn (Distributed Computing), tarvelaskentaan (Utility Computing.) ja viime aikoina esiin on tullut myös verkostoituminen (Networking).(Vouk, 2008.)

1.2.3 TCP ja IP–Protokollat

TCP (Transmission Control Protocol) ja IP (Internet Protocol) – protokollia käytetään internet-yhteyden luomiseen. Jokainen TCP ja IP -verkossa aktiivisesti osallistuva verkkolaite tarvitsee toimiakseen IP-osoitteen, aliverkon ja oletusyhdyskäytävän. TCP/IP -protokollan avulla verkkolaitteet voivat kommunikoida keskenään ja lähettää paketteja toisilleen. (Leidigh, 2000.) Protokollat vastaavat siis tiedonsiirtoyhteydestä. Laitteille määritellään IP-osoitteet, joiden perusteella ne tunnistetaan.

1.2.4 HTTP – Protokolla

HTTP (Hyper Text Transfer Protocol) – Protokolla toimii TCP protokollan päällä. HTTP -protokollan yhteysliikenne koostuu pyynnöistä (Request) ja vastauksista (Response). Jokainen tiedonsiirto alkaa siitä, kun verkkoasiakas (Client)

lähettää liikkeelle pyynnön. Tämän jälkeen palvelin (Server) valmistaa vastauksen pyynnön pohjalta ja lähettää sen takaisin verkkoasiakkaalle (Client). Asiakas odottaa yleensä vastausta ennen kuin voi jatkaa toimintaa. Mikäli vastaus ei tule sovelluksen kannalta riittävän nopeasti, aikakatkaisu lopettaa odottamisen ja pyyntö näin epäonnistuu. (Jestratjew, 2013.)

1.3 Tavoitteet ja menetelmät

Tutkielman tavoitteena on selvittää, voidaanko automaatiojärjestelmästä tehtävä tiedonkeruu pilvipalveluun toteuttaa TCP/IP- ja HTTP-protokollaa käyttäen. Lisäksi pyritään selvittämään, kuinka näiden protokollien käyttö mahdollistetaan ja voidaan toteuttaa sekä miten tietoa automaatiojärjestelmään luodaan rajapinta, jonka kautta tietoa voidaan kerätä pilvipalveluihin.

Seuraavissa alaluvuissa käydään läpi tutkimukseen vaikuttavat tekijät.

1.3.1 Aihepiirin rajaus

Tutkimuksen aihe on rajattu koskemaan tiedonkeruuta automaatiojärjestelmästä ja selvittämään kuinka tiedonkeruuseen tarvittava rajapinta voitaisiin toteuttaa TCP/IP- ja HTTP-protokollia käyttäen. Teollisuuden järjestelmien ja pilvipalveluiden välisestä rajapintatoteutuksesta kirjallisuuskatsaus pyrkii esittelemään tämän hetkisen tiedon, joka löytyy alan kirjallisuudesta.

1.3.2 Tutkimusongelma

Tutkimuksen varsinainen tarkoitus on vastata seuraavaan kysymykseen:

Miten teollisuuden ohjausjärjestelmän ja pilvipalvelun välinen tiedonkeruurajapinta voidaan toteuttaa itse TCP/IP- ja HTTP – protokollaan tukeutuen ja muokkaamalla valmiiden ratkaisujen lähdekoodia.

Pääkysymyksen lisäksi tutkimuksen tarkoituksena on vastata seuraavaan alakysymyksiin:

- Mitä valmiita ratkaisuja voidaan mahdollisesti hyödyntää rajapintatoteutuksessa?

1.3.3 Tutkimusmenetelmät

Tämä tutkimus toteutetaan kirjallisuuskatsauksena ja konstruktiivisena tutkimuksena. Kirjallisuuskatsauksen avulla määritellään teollisuuden ohjausjärjestelmät eli automaatiojärjestelmät, TCP/IP- ja HTTP protokollat sekä pilvipalvelut.

Kirjallisuuden avulla selvitetään, että miten tätä aihetta on tähän mennessä tutkittu ja minkälaisia tutkimustuloksia tutkimusartikkeleista löytyy. Kirjallisuuden avulla selvitetään myös tutkimuksen keskeiset käsitteet, sekä se miten teollisuuden standardit ja tietoturva tulisi ottaa huomioon toteutuksessa.

Konstruktiivisessa osuudessa on tarkoitus esittää, että miten tämä tiedonsiirto TCP/IP- ja HTTP-protokollaa käyttäen automaatiojärjestelmältä pilvipalvelulle voitaisiin toteuttaa käyttäen valmiita toteutuksia ja muokkaamalla valmiiden ratkaisujen lähdekoodia.

Konstruktiivisessa tutkimuksessa haluttu päämäärä on tiedossa, mutta sen saavuttaminen ei. Olemassa olevan tiedon pohjalta pyritään siis rakentamaan uusi toteutus käsiteltävästä asiasta tai ilmiöstä. Tätä rakentamista ja arviointia voidaankin luonnehtia konstruktiiviseksi tutkimukseksi. (Järvinen & Järvinen, 2004.) Tämän konstruktiivisen tutkimuksen tuloksena on tarkoitus toteuttaa uusi systeemi, jolla tutkimusongelmaan voidaan vastata.

2 Aikaisemmat tutkimukset aiheeseen liittyen

2.1 Langattomat anturiverkot ja pilvipalvelut

Langattomista anturiverkoista ja niiden tiedonsiirrosta pilvipalveluihin löytyi aikaisempaa tutkimusta, joka liittyy myös tähän tutkimusaiheeseen. Langattomat anturiverkot (Wireless Sensor Networks) koostuvat erilaisista solmuista joilla on valmiutena mitata, tunnistaa ja laskea erilaisia muutoksia mittauskohteessa sekä hoitaa viestintää langattomasti. Langattomat anturiverkot pitävät sisällään useita omia sovelluksia. Integroimalla tämän langattoman anturien paikallisverkon Internettiin voidaan edelleen tehostaa näitä sovelluksia, joita voidaan käyttää reaaliajassa pilvipalvelusta ja tallentaa anturiverkkojen keräämiä tuloksia sinne. (Shah & kumppanit, 2013.)
Myös automaatiojärjestelmät pitävät sisällään erilaisia antureita (Sensors) prosessitietojen tunnistukseen ja ohjausyksiköitä (Control Unit) näiden prosessitietojen laskentaan. Nämä automaatiojärjestelmät toimivat prosessikohtaisissa verkoissa, käyttävät prosessikohtaisia laitteita ja ne ovat eristetty pois internetistä. (Byres & Oulton, 2013). Myös automaatiojärjestelmässä tälle automaatiojärjestelmän ohjausyksikölle pitäisi rakentaa rajapinta, josta toteutettaisiin yhteys internettiin. Nämä langattomien anturiverkkojen paikallisverkot ovat siis idealtaan hyvin samanlaisia kuin automaatiojärjestelmien prosessikohtaiset verkot.

Langattomat anturiverkot mahdollistavat uusia ratkaisuja tiedonkeruuseen erilaisissa kuljetusalan, yritystoiminnan, terveydenhuollon, teollisuusautomaation ja ympäristönseurannan sovelluksissa (Ahmed & Gregory, 2011). Langattomat anturiverkot ovat siis tarkoitettuja hyvin samantyylisiin ratkaisuihin kuin automaatiojärjestelmätkin. Niiden tarkoitus on pääasiallisesti kerätä tietoa, mikä eroaa automaatiojärjestelmistä, joissa on tarkoitus tämän lisäksi sekä ohjata että valvoa prosessia.

Langattomista anturiverkoista pilvipalveluihin johdettu informaatio on arvokas tietovara, josta tiedonnälkäiset sovellukset hyötyvät, kun langaton anturiverkko integroidaan pilvipalveluun ja liitetään se aiottuun pilvilaskennan palvelutyyppiin. Käyttäjien vaatimuksiin voidaan tarjota palvelua kolmella kerroksella, jotka ovat IaaS, PaaS ja Saas.(Ahmed & Gregory, 2011.) Saas eli verkkosovelluspalvelu tarkoittaa, että palveluntarjoaja tarjoaa pääsyn ohjelmiston ajonaikaiseen versioon ja dataan sekä ohjelmistotuotteisiin käyttöliittymäohjelmiston kautta. PaaS eli sovellusalusta palveluna tarkoittaa, että kuluttaja voi ajaa ennaltamääriteltyä sovellusta palveluntarjoajan virtuaalisella alustalla. Tämän tyyppisissä palveluissa kuluttajalla ei ole hallintaoikeuksia alustan infrastruktuuriin eikä sille asennettuihin sovelluksiin. IaaS eli Infrastruktuuri paveluna tarjoaa kuluttajille edun käyttää infrastruktuuria, joka sisältää mm. laskentatehon, tallennustilan ja verkon. Kuluttaja voi ajaa alustalla useita sovelluksia välittämättä taustalla olevan infrastruktuurin ylläpidosta. (Shah & kumppanit, 2013.) Langattomien anturiverkkojen keräämää tietoa voidaan siis jalostaa pilvilaskennan avulla hyvin laajasti erilaisia pilvilaskennan palvelutyyppejä käyttäen. Kuluttaja pystyy määrittelemään itselleen sopivan pilvilaskennan palvelutyypin ja rakentaa sen päälle tarpeidensa mukaan langattomien anturiverkkojen tiedon jatkojalostusta varten tarvittavan ohjelmiston tai infrastruktuurin.

Langattomat anturiverkot ovat suunniteltu keräämään tietoa todellisen maailman ympäristöstä, mutta mitä tehdä tiedoille, kun organisaatio, joka keräsi tiedot ei enää tarvitse niitä. On monia syitä pitää tieto tallessa kuten esimerkiksi historiallinen arvo, tulevaisuuden tutkimukset ja mahdolliset uusintaanalyysit tulevana ajankohtana. (Ahmed & Gregory, 2011.) Tätä samaa tiedon mahdollista jatkojalostusta voitaisiin käyttää hyväksi automaatiojärjestelmissä. Näin saataisiin pilvilaskennan avulla rikastettua tätä prosesseista kerättyä tietoa vielä tarkemmaksi, monikäyttöisemmäksi ja arvokkaammaksi sekä vertailla prosessien historiatietoja keskenään. Prosesseista kerättyä tietoa, jota ei tarvita prosessin reaaliaikaisen toiminnan selvittämiseen voitaisiin siis käyttää prosessien vertailuun ja tehostamiseen historiatietojen perusteella sekä selvittää mahdollisia tulevia virhetiloja ja prosessivirheitä. Kerätyn tiedon perusteella voitaisiin mahdollisesti ennakoida virhetiloja ja selvittää prosessien mahdolliset ihanne toimintatilat.

Langattomat anturiverkot eroavat automaatio- ja prosessinohjausjärjestelmistä siinä, että anturiverkoilla on yleisesti tarkoitus kerätä tietoa suoraan yksittäisten anturien tuottamista tiedoista tai yksittäisten kohteiden tiedoista. Automaatio- ja prosessinohjausjärjestelmistä kerätään tietoa tästä prosessista ja yhdistellään monen anturin ja mittalaitteen keräämää tietoa. Automaatiojärjestelmissä tieto voi siis olla monen mittalaitteen tuottaman tiedon summa, jonka avulla prosessin tilaa voidaan mitata ja selvittää. Automaatiojärjestelmän keräämä tieto voi siis tulla monesta yksittäisestä anturista.

2.2 Palvelukeskeisen arkkitehtuurin käyttö automaatiojärjestelmien ja pilvipalvelun yhdistämisessä

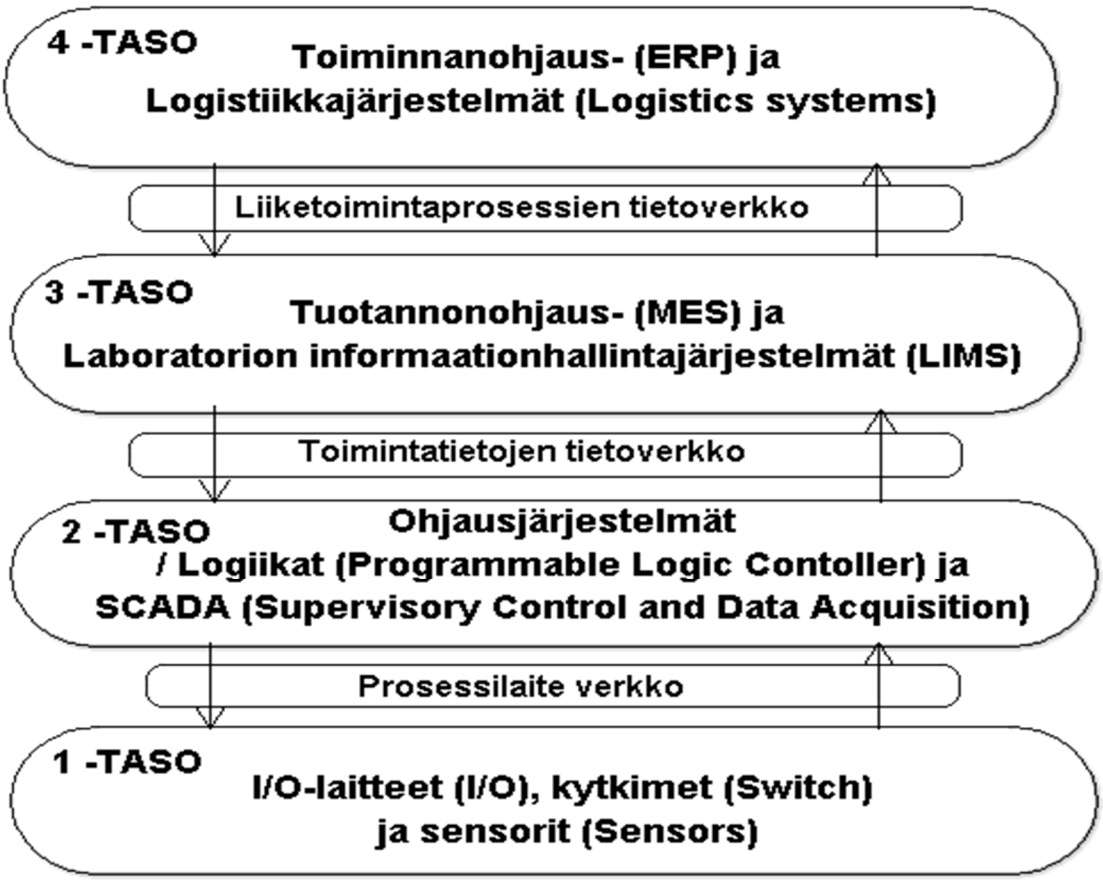
Palvelukeskeinen arkkitehtuuri (Service-oriented Architecture, SOA) lisää joustavuutta liiketoiminnan prosesseihin ja sovelluksiin, jotka ohjaavat niitä. Palvelukeskeinen arkkitehtuuri pilkkoo liiketoiminnan prosessit pienemmiksi toiminnallisiksi palasiksi, joita kutsutaan palveluiksi. Nämä palvelut asemoidaan omiksi sovelluksiksi, jonka jälkeen saadaan aikaan standardointi ja yhteen toimivuus. (Lager, 2005.) Ohjelmisto koostuu osista, joita voidaan kutsua moduuleiksi tai olioiksi tai aliohjelmiksi. Palvelu tuotetaan suorittamalla näitä ohjelman osia. SOA:n yksi keskeinen idea onkin se, että se on toteutettu hajautettuna laskentana. Kutsuva ohjelma kutsuu toista ohjelmaa verkon yli. Palvelukeskeinen arkkitehtuuri sisältää monia pieniä paketteja koodia, joista jokaiselle on määritelty tietty tehtävä, eli se tuottaa tiettyä (ali)palvelua. Palvelukeskeisellä arkkitehtuurilla tarkoitetaan siis sitä, että ohjelmisto jaotellaan tarjottaviin osiin, joita kutsutaan palveluiksi. Ohjelmisto koostuu osista eli palveluista, joita voidaan valita ohjelmaan sen mukaan, kuinka niihin on asiakkaalla tai käyttäjällä tarvetta.
Ketjuttamalla palveluita yhteen saadaan aikaan modulaarinen lähestymistapa, joka mahdollistaa joustavuutta ohjelmiston toteuttamiseen (Lager, 2005).

Pilvilaskenta (Cloud Computing) viime aikoina tullut uusi tietojenkäsittelyn paradigma monilla sovellusalueilla, käsittäen niin toimisto- kuin toiminnanohjausjärjestelmät. Pilvilaskenta tarjoaa dynaamiseen ja joustavan infrastruktuurin isännöimään tietokoneen resursseja sekä toimittaa ne palveluna asiakkaan tai kuluttajan tarpeiden mukaan. Tulevaisuuden teollisuuden automaatiojärjestelmien on oltava joustavia ja ketteriä, joten pilvilaskentaa pidetään lupaavana ratkaisuna tälle alalle. (Givehchi & kumppanit, 2013.) Pilvilaskentaa voitaisiin hyödyntää oleellisesti tulevaisuuden automaatio- ja teollisuuden järjestelmissä. Tietojenkäsittelyä voitaisiin keskittää joustavasti ja dynaamisesti tälle pilven tarjoamalle infrastruktuurille. Näin tietojenkäsittelyyn voitaisiin tarpeiden mukaan määritellä sen verran laskentatehoa ja muistikapasiteettia, kuin kulloisessakin prosessin tiedoille sitä tarvittaisiin. Hyötynä tästä saataisiin myös se, että kaikki tieto pystyttäisiin keräämään talteen paikkaan, jossa sen jatkokäsittely olisi helpompaa ja se olisi helpommin saatavilla.

Nykyään automaatiojärjestelmät joutuvat mukautumaan nopeasti kasvavien markkinoiden vaatimuksiin, jolloin haetaan ketteryyttä ja joustavuutta tuotantoyksikköihin. Teollisuuden neljännen vallankumouksen kasvu perustuu älykkääseen tuotantoon. Pääpaino neljännessä teollisuuden vallankumouksessa on älykkäissä esineissä, autonomisissa tuotteissa ja uusien tietotekniikan menetelmien käyttämisessä päätöksientekoprosesseissa. Pilvilaskenta, joka on uusi kehityssuunta tietotekniikan alueella saattaa toimia tämän suuntauksen mahdollistajana tulevaisuuden automaatiojärjestelmissä. Pilvilaskenta onkin jo viime aikoina vaikuttanut monilla aloilla muokaten niitä, kuten esimerkiksi toimisto ja toiminnanohjausjärjestelmissä. (Givehchi & kumppanit, 2013.) Älyk-

käillä mittalaitteilla ja antureilla tulee siis olemaan iso vaikutus tässä neljännessä teollisuuden vallankumouksessa. Tietotekniikan menetelmiä voidaan hyödyntää pilvilaskennan pohjalta ja käyttämällä erilaisia älykkäitä laitteita. Prosessien ohjauksiin, eli niin sanottuihin päätöksentekovaiheisiin, saadaan näin hoidettua ketteryyttä sekä keskitettyä prosessien valvontaa ja tiedonkeräystä hyödyntäen tietotekniikan uusia menetelmiä. Näiden tietotekniikan menetelmien hyötyjä voidaan huomata erilaisista jo tuotetuista tuotannonohjaus- (Manufacturing Execution System, MES) ja toiminnanohjausjärjestelmistä (Enterprise Resource Planning Systems, ERP Systems).

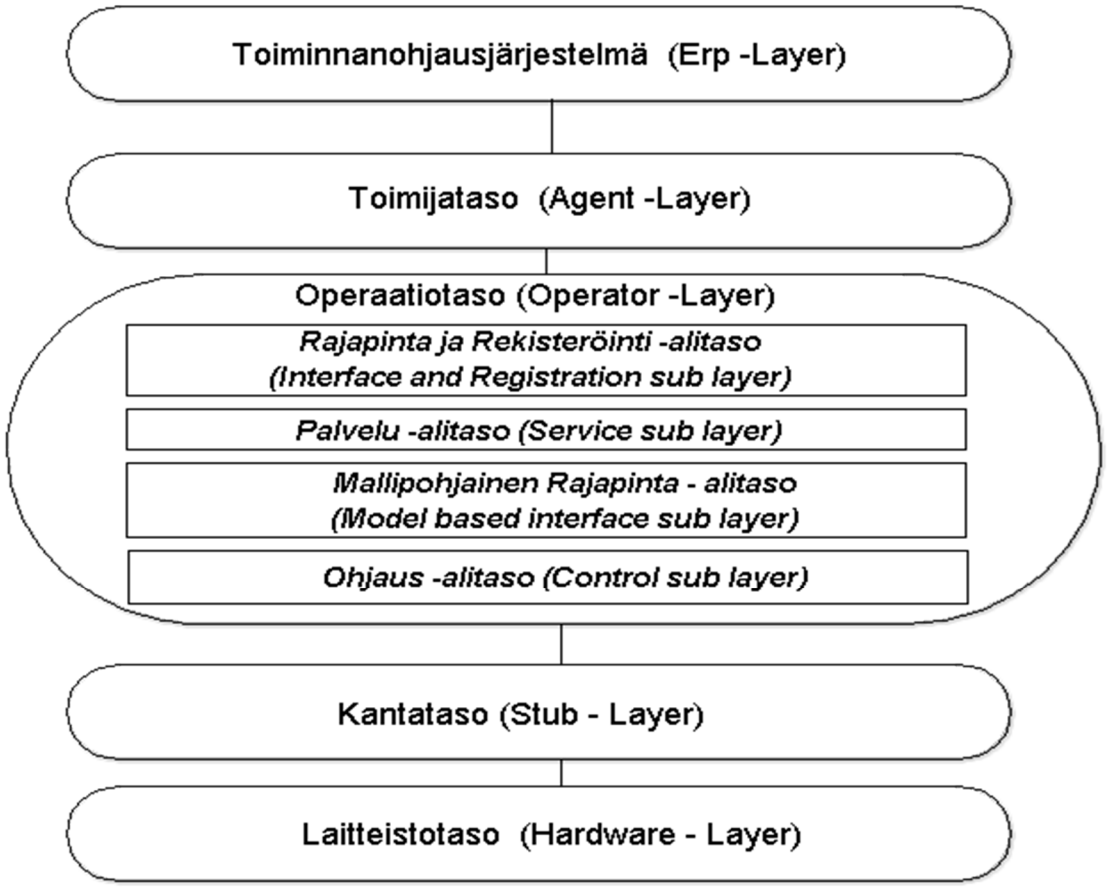
Pilvipalveluita onkin esitetty uudeksi liiketoimintamalliksi muuttamaan perinteisen tehdasteollisuuden liiketoimintamallia ja luomaan älykkäitä tehdasverkkoja, jotka edistävät tehtaiden tehokkuutta. Tätä valmistustekniikkaa on ehdotettu kutsuttavaksi ketteräksi pilviteollisuudeksi (Cloud Agile Manufacturing). Tärkeimpänä tavoitteena on tarjota teollisuuden automaation toiminnot palveluina, jotta automaatiojärjestelmän ulkopuolisille eli korkeamman tason käyttäjille saataisiin pääsy automaatiojärjestelmän toiminallisuuksiin mutkattomasti. ISA -standardissa tarkoitetaan tasoja 3 ja 4.(Givehchi & kumppanit, 2013.) Nagorny & kumppanit (2012) ehdottavat palvelukeskeisen arkkitehtuurin käyttöä helpottamaan älykkäiden automaatioverkkokomponenttien hallintaa ja valvontaa hajautetussa tuotantoympäristössä. Kuviossa 1 esitetään miten palvelut syntyvät ja esiintyvät automaatiokomponenteille, jotka sijaitsevat palvelukeskeisen arkkitehtuurin tasoilla 1 ja 2 ISA-95 -standardissa. Tasot 1 ja 2 ovat olennainen osa automaation palvelupilveä, joka on seurausta fyysisen tuotantoympäristön virtualisoinnista. (Nagorny & kumppanit, 2012.) Näitä tasoilla 1 ja 2 olevia automaatiokomponentteja virtualisoitaisiin pilvipalveluun, jonka avulla saataisiin helpotettua niiden valvontaa ja hallintaa, koska pystyttäisiin seuraamaan niiden tilaa etänä järjestelmien ulkopuolelta. Niitä voitaisiin sitten tarjota tuotantoyksikköön palveluna pilvipalvelusta käsin. Tämä Nagonyn & kumppanien (2012) ehdottama lähestymistapa on siis hyvin samanlainen kuin Givenchi & kumppanien (2013) pilvilaskennan hyödyntäminen automaatiojärjestelmissä. Nagornyn & kumppanien (2012) tapauksessa mennään vain tarkemmin yksityiskohtaisempiin seikkoihin automaatiojärjestelmän muuttamisessa palvelukeskeisen arkkitehtuurin mukaiseksi. Artikkelissa kuvataan lähestymistapa, jolla voidaan toteuttaa palvelu ja monitoimisesti suuntautunut? teollisuuden valmistuslaitosten automaatioarkkitehtuuri hyödyntäen ISA-95 –standardin muunnos. (ks. kuvio 1). (Nagorny & kumppanit, 2012.)



KUVIO 1 ISA-95 standardi (Nagorny & kumppanit, 2012)

Palvelukeskeinen automaatioarkkitehtuuri seuraa ISA-95 standardin peruserittelyä ja se on hierarkkisesti rakennettu, mutta sen on helpotettava jokaisen tason sisäisiä sekä tasojen välisiä rakenteellisia yhdistettävyyksiä ja yhteistoiminnallisuuksia komponenttien välillä (ks. kuvio 1). Nagornyn & kumppanien (2012) ehdotetun arkkitehtuurin tasot on jaettu viidelle tasolle toisin kuin perinteisen ISA-95 -standardin tasot, joita on neljä. ISA-95 -standardissa ensimmäisellä tasolla ovat I/O-laitteet, kytkimet ja sensorit. Toisella tasolla ovat ohjausjärjestelmät, logiikat ja SCADA (Supervisory Control and Data Acquisition). Kolmannella tasolla ovat tuotannonohjausjärjestelmät (MES), laboratorion informaationhallintajärjestelmät (LIMS) ja vastaavat. Neljännellä tasolla sijaitsevat toiminnanohjausjärjestelmät. Nagornyn & kumppanien (2012) ehdotettu Operaatio 2.0 (Operator 2.0) -arkkitehtuuri (ks. kuvio 2) on jaettu viidelle fyysiselle tasolle, jotka ovat suoraan kartoitettu tai koostettu ISA-95 -standardin tasojen fyysisten komponenttien ja toimintojen mukaan. Tärkein ero näiden kahden arkkitehtuurin välillä on, että Operaatio -2.0 käyttää ainutlaatuista viestintäverkkoa, joka kattaa koko arkkitehtuurin, alkaen ensimmäiseltä tasolta arkkitehtuurin ylimmälle tasolle asti. Tämä on toteutettu käyttämällä yleistä Ethernet -tekniikkaa. (Nagorny & kumppanit, 2012.)

Operator 2.0 -arkkitehtuurin ensimmäinen taso, eli laitteistotaso (Hardware Layer), käsittää laitteistokomponentit, joissa on omat rajapinnat (Interfaces), joita voidaan käyttää valmistusprosessin ohjausjärjestelmissä. Toinen taso, eli kantataso (Stub Layer), on laitteistotason yläpuolella oleva taso, joka muuntaa laitekohtaiset rajapinnat yhtenäiseksi protokollaksi ohjausjärjestelmän ja laitteen välille. Esimerkiksi teollisuuden robotin ohjausjärjestelmää liitettäessä verkkopalveluun voi esiintyä protokollien välisiä ristiriitoja, koska robotin ohjausjärjestelmissä käytetään omia protokollia viestintään. Tähän väliin voidaan asentaa kantataso, jonka avulla saadaan eri protokollat toimimaan yhtenäisesti. Kantataso kommunikoi laitetasolle tarvittavien protokollien avulla ja muuntaa operaattoritasolle päin kommunikoinnin yhtenäiseksi. Tämän takia kantatasolle pitää luoda kanta jokaiselle komponentille, joka lisätään valmistusjärjestelmään. Kolmas taso, eli operaattoritaso (Operator Layer), on palvelupohjainen monikerros-rakenteellinen taso, joka vastaa laitteiston osien virtualisoinnista ja altistaa niiden toiminnot palvelukeskeisiksi. Tämä tarkoittaa sitä, että operaattorit mahdollistavat toisen tason, eli kantatason, tuoman laitetason komponenttien palvelukeskeisen käytön informaatioinfrastruktuurissa. Operaattoritaso tarjoaa mahdollisuuden eri automaatio – ja valvontajärjestelmien toiminnoille. Operaattoritaso mahdollistaa automaatiopalvelupilven käytön, eli automaation toimintojen käytön verkkopalveluissa. Neljäs taso, eli toimijataso (Agent Layer) on taso, jolta toimijat ja operaattorit pääsevät käsiksi palveluihin paikallisen operaattoritason kautta. Viidennellä tasolla (ERP Layer) taas on toiminnanohjausjärjestelmä, joka kerää näiden toimijatason tiedot yhteen ja tarjoaa rajapinnan pilvipalvelulle ja ihmisille. (Nagorny & kumppanit, 2012.)



KUVIO 2 Nagornyn & kumppanien (2012) Operator 2.0 – arkkitehtuuri

Ensimmäinen taso on teknologisesti heterogeeninen ja koostuu pääasiassa teollisuuden automaatio-, mekatroniikka- ja tuotanto-osista. Näitä osia ovat ohjelmoitavat logiikat (PLC), I/O-laitteet (I/O Devices) ja robottiohjausyksiköt (Controllers of Industrial Robots), jotka kaikki sisältävä omat kommunikointiprotokollansa ja eivät näin ollen ole yhteen toimivia toistensa kanssa. Toisella tasolla tämä yhteentoimivuusongelma hoidetaan luomalla jokaiselle laitteelle oma kanta, joka yhdistää ne yhtenäiseen protokollaan. Teknisesti tämä tarkoittaa, että jokainen komponentti tarvitsee kannan tältä kantatasolta yhtenäisen protokollan käyttöön. Seuraava taso, eli kolmostaso, käyttäisi tätä yhteistä yhdessä määriteltyä protokollaa ohjatakseen ensimmäisellä laitteistotasolla olevia laitteita. (Nagorny & kumppanit , 2012.) Arkkitehtuurin kaksi ensimmäistä tasoa siis luovat yhteydet laitteisiin, joilta operaattoritason ohjausyksiköt ja ohjausjärjestelmät sitten lukevat tietoa tai lähettävät ohjauskäskyjä.

Tässä artikkelissa oli paljon yhtäläisyyksiä tutkimusongelmaani, mutta artikkeli käsitteli enemmän kokonaisarkkitehtuurin luontia ja sitä, miten arkkitehtuurillisesti viestintäprotokolla automaatiolaitteiden, komponenttien, ohjausjärjestelmien ja toiminnanohjausjärjestelmien välille tulisi toteuttaa, jotta sinne saataisiin yhtenäinen protokolla viestintään laitteiden välille. Artikkeli ei ottanut kantaa siihen, miten itse laitteelle tai ohjausjärjestelmälle tulisi luoda raja-

pinta, miten sen tulisi toimia ja mitä protokollia tämän käyttöön tarvittaisiin. Tutkimusongelmassani sijoittuu kantatasolle ja operaattoritasolle.

Karnouskos (2011) toteuttaisi www-sovelluspalvelutoteutuksen (Web Service) avulla tulevaisuuden tehtaiden laitteet palveluperustaisiksi. Liiketoiminnan sovelluksilla tulisi olla aina pääsy laitteiden tietoihin ja tiloihin www-sovelluspalvelun avulla. Vaikka muita yhteystoimintatapoja on olemassa, vain www-sovelluspalveluabstraktio mahdollistaa asynkronisen viestintämenetelmän aidosti riippumatta taustalla olevasta käyttöjärjestelmästä tai ohjelmointikielestä. (Karnouskos, 2011.) Rajapintaa toteutettaessa www-sovelluspalvelun avulla on viestinnässä käytössä yhteinen protokolla. Tätä protokollaa voidaan käyttää käyttöjärjestelmästä tai ohjelmistokielestä riippumatta eri kohteissa asynkronisesti eli ajasta riippumattomasti.

Karnouskos:n (2011) toteutuksessa ei pitäisi ottaa kantaa vain lähettämiseen, joka tapahtuisi järjestelmässä pelkästään alhaalta ylöspäin vaan ennen kaikkea ylhäältä alaspäin automaatiolaitteille. Liiketoimintajärjestelmä tarvitsee suoran pääsyn reaaliaikaisesti asiayhteyden mukaisiin tietoihin automaatiojärjestelmässä, jotta tiedon siirto olisi reaaliaikaista. Tässä tapauksessa tarpeettomat yksityiskohdat pitäisi saada piiloon. Palvelun pitäisi toteuttaa vain haluttu toiminnallisuus. Tämä on yleensä mahdollista toteuttaa koostettuna muista yleisistä palveluista. (Karnouskos, 2011.)

Toteutusvaihtoehto tälle on evolutionäärinen lähestymistapa, jossa laitteet, jotka eivät ole www-sovelluspalvelulle yhteensopivia, ohjelmistopäivitetään ja sisällytetään niiden ohjelmistoon www-sovelluspalvelupino tai korvataan ne toinen toisensa jälkeen uusilla. Näihin uusiin laitteisiin tämä toiminta on sisällytetty ja niillä voitaisiin korvata vanhat laitteet, jotka muutenkin ovat eläneet aikansa ja vaativat toiminnallisia ja fyysisen tason päivityksiä. Kaikilla laitteilla on kyky liittyä suoraan palvelukeskeiseen arkkitehtuuriin implementoimalla niihin www-sovelluspalvelut valmistusvaiheessa ja tarjota toiminnallisuutta palveluna muille. (Karnouskos, 2011.) Tämä laitteiden palveluperustaistaminen lähtisi siis liikkeelle siitä, että laitteisiin luotaisiin jo aluksi mahdollisuus ottaa niihin viestintäyhteys käyttäen www-sovelluspalvelua, joka karsisi Nagornyn & kumppanien (2012) arkkitehtuuritoteutuksessa olevan kantatason (ks. kuvio 2) kokonaan pois. Tämän kantatason ideanahan oli toteuttaa näiden laitteiden ja ohjausjärjestelmien välinen rajapinta niin, että se voitaisiin toteuttaa yleiselle protokollalle. (Nagorny & kumppanit , 2012.) Karnouskos (2011) toteuttaisi tämän rajapinnan suoraan laitteelle ja päivittäisi kaikkiin laitteisiin jo tuotantovaiheessa tämän www-sovelluspalvelun, jonka jälkeen siihen saataisiin yleisiä protokollia käyttäen luotua viestintä. Karnouksen (2011) laitteisiin lisättävän www-sovelluspalvelun sekä Nargornyn ja kumppanien (2012) arkkitehtuurisen toteutuksen yhdistämistä voitaisiin hyödyntää tässä tutkielmassa käsiteltävässä tutkimusongelmassa.

Nykyään monet prosessiautomaatioon sisällytetyt viestintäverkot ovat yhä sovelluskohtaisia ja suunniteltu keräämään I/O –tietoja. Tämän lisäksi käytössä on avoimia standardoituja protokollia, kuten Modbus, Profibus ja erilaiset Ethernet vaihtoedot. (Cucinotta, 2009.) Avoimen verkon infrastruktuurin

adoptoiminen, jossa on kyky tarjota plug & play palveluita sekä kyky piilottaa laitteiden monimutkaisuus, tarjoaa yksinkertaisemman ja luonnollisemman työnkulun mekaniikkainsinöörille ja valvontainsinöörille. Avoin verkkoinfrastruktuuri mahdollistaa myös saman alustan tunnistamaan esineitä ja niiden ominaisuuksia. Myös teollisuuslaitoksen uudelleen järjestely kärsii näistä nykyaikaisista sovelluskohtaisista viestintäverkoista ja siitä, että riittävän tason älykkyyttä ei ole lisätty suoraan laitteisiin. Laitteita ohjataan keskitetysti ohjausyksiköstä, johon täytyy jokaisen muutoksen ja uudelleenjärjestelyn jälkeen tehdä muutoksia sen ohjelmakoodiin. Tämä rajoittaa modulaarisointia ja yhteentoimivuutta. (Cucinotta, 2009.)

Ethernet-pohjainen viestintä, johon sisältyy lähetyksen ohjaus protokolla/ internet -protokolla (TCP/IP) ja verkko-ominaisuudet reaaliaikaisen liikenteen hallintaan voidaan toteuttaa palvelukeskeisen arkkitehtuurin avulla. Palvelukeskeinen arkkitehtuuri helpottaa järjestelmän ongelmien tunnistuksessa, laitteiden välisessä tunnistuksessa ja viestinnässä, kun www-sovelluspalvelu on laajennettu tukemaan reaaliaikaista yksittäistä toimia. (Cucinotta, 2009.)

2.2.1 OPC (OLE for Process Control) Protokolla

Tutkimuksen tutkimusongelma pyritään ratkaisemaan käyttämällä OPC – protokollaa (OLE for Process Controll). OPC-protokollaa voitaisiin käyttää palveluperusteisten arkkitehtuurien ISA-95 -standardin 2-tasolla sekä Operaatio 2.0 arkkitehtuurin laitetason ja operaattoritason väliselle kantatasolle. OPC -protokolla määrittelee koko automaatiotoimialan hyväksymät standardit tietojen hakuun teollisuuden järjestelmistä (Goldschmidt & Mahnke, 2012.). OPC -standardi on alunperin suunniteltu rajapinnaksi ohjaamaan automaatiolaitteiden kommunikaatiota, jossa niille voidaan kirjoittaa ja niiltä lukea tietoa. OPC -protokolla on implementoitu tuhansiin tuotteisiin ja se on standardisoitu kommunikointirajapinnaksi automaatiojärjestelmän eri tasojen välille. (Girbeal, 2010.) OPC -protokollalla voidaan toteuttaa hyvin samoin toteutus kuin Karnouskosin palveluperustainen idea oli, jossa laitteisiin integroidaan www-sovelluspalvelulle rajapinta. Karnouskosin esittämä protokollaratkaisu on siis hyvin samanlainen kuin olemassa oleva OPC-protokolla. Markkinoilta löytyy yli 22 000 OPC -standardia tukevaa tuotetta ja valmistajia tuotteille on yli 3500 (Uslar & Kumppanit ,2012). Valmistajien joukossa ovat kaikki automaatioalan tärkeimmät toimijat. Tämän seurauksena lähes jokaisessa automaatioteollisuuden tuotteessa on implementoituna OPC -protokolla.(Uslar & Kumppanit ,2012.) OPC -protokolla näyttää olevan suoraan ratkaisu tähän Karnoukos:n esittämään ongelmaan.

OPC -protokolla on teollisuuden ohjaus- ja valvontajärjestelmien prosessitietojen hakuun rakennettu palvelin ja asiakasohjelmiston välinen kommunikointistandardi. Perinteinen OPC DA (Data Access) -palvelin sijaitsee tyypillisesti teollisuuden verkkojen ja ylemmän tason teollisuuden tietojärjestelmien välisellä tietokoneella. Tämä perinteinen OPC – protokolla on rajattu toimimaan Mircosoftin DCOM (Distributed Component Object Model) objektiprotokol-

lan mukaan. Tämän johdosta perinteistä OPC -palvelinta ei voida sisällyttää teollisuuden järjestelmiin ja laitteisiin, jotka eivät tue Microsoft:n DCOM -objektiprotokollaa. (Cupec, 2013.)

Perinteisen OPC:n ongelma on siis se, että laitteista täytyy löytyä tuki DCOM -objektiprotokollalle, joka taas rajaa pois kaikki muut kuin Microsoftin ohjelmistoalustat kuten unix -pohjaiset laitealustat. Perinteinen OPC eli DCOM pohjaiseen teknologiaan perustuva on Microsoft:n tuote ja se toimii ainoastaan microsoftin alustoilla. OPC ei toimi siis Linux, VxWork:lla tai muilla alustoilla.(Rinaldi, 2016.) OPC -protokollan nopea menestys johtuikin Windows koneiden käytön nopeasta kasvusta sekä DCOM -objektiprotokollan valinnasta, koska se löytyi kaikista windows -koneista. Samaan aikaan kritiikkiä ilmeni siitä, että OPC oli liian keskittynyt Microsoft-yhtiön tuotteisiin, alustariippuvainen eikä palomuuriyhteensopiva. Tämän johdosta se ei soveltunut käytettäväksi Internetistä. Kun XML ja www-sovelluspalvelu rajapinnat(Web Service interface) tulivat on OPC -säätiö soveltanut niitä saadakseen DCOM puutteita poistettua. Vuodesta 2003 OPC XML Data Access (OPC DA) määrittely on tarjonnut ensimmäisen palvelukeskeisen arkkitehtuuripohjaisen lähestymistavan klassisen DCOM- pohjaisen OPC -teknologia lisäksi. OPC DA www-sovelluspalvelupohjainen ratkaisu mahdollistaa kommunikoinnin alusta- ja valmistajariippumattomasti. (Cavalier, 2012.)

OPC -säätiön tuorein julkaisu on OPC UA (Unified Architecture) -standardi (OPC Foundation, 2008), joka perustuu palveluperusteiseen ja alustariippumattomaan lähestymistapaan luomalla uusia ja helppoja mahdollisuuksia toimia alustariippumattomasti mm. Unix -järjestelmissä tai toteuttaa sulautettua valvontaa muilla alustoilla ja toteuttaa OPC -yhteydet internetin ylitse. OPC UA luo siis uusia mahdollisuuksia käyttää OPC komponentteja muissakin kuin Windows -ympäristöissä. (Cavalier, 2012.) Tutkimuksessa rajapintatoteutus on tehty tällä OPC UA standardilla alustariippumattomasti.

OPC UA perustuu paljon käytettyyn vertikaaliseen kommunikaatiostandardiin OPC DA:han. OPC DA perustuu Microsoft DCOM:iin ja sitä on vaikea käyttää suoraan arkkitehtuurin laitetasolla. OPC DA palvelimet on yleensä asennettu pc-tietokoneisiin ja erilaisiin teollisuuden verkkoihin kommunikoimaan suoraan laitetason elektronisten laitteiden kanssa. Tämä toiminta johtaa suuriin viiveisiin ja voi johtaa ristiriitaisen tiedon syntyyn. Uusin määritys OPC UA eli yhdistynyt arkkitehtuuri määrittelee uuden alustariippumattoman mallin viestintään, jota voidaan soveltaa eri laitteissa ja laitteistoalustoissa. (Fokert & Kumppanit, 2011.) OPC UA:n avulla voidaan siis saada luotua yhteys automaatiojärjestelmän ohjausyksiköille alustariippumattomasti. Tämä mahdollistaa UNIX -järjestelmien käytön.

2.2.2 OPC UA (OLE for Process Control -Unified Architecture) -protokolla

Klassinen OPC (OLE for Process Control) määrittelee standardin koko toimialan hyväksymän normiston tiedonsiirron automaatiojärjestelmiin ja sieltä pois. Uudempi ratkaisu OPC UA tuo mukanaan uusia toiminnallisuuksia kuten yhtenäisen osoiteavaruusmallin, palveluperustaisen rajapinnan sekä laajennettavan metamallin. Tämä mahdollistaa sen, että OPC UA:ta voidaan käyttää pienten sulautettujen teollisuuden ohjausjärjestelmien ja hajautettujen ohjausjärjestelmien (Distributed Control System , DCS) lisäksi valmistuksenojaus- ja toiminnanohjausjärjestelmissä (Goldschmidt & Mahnke, 2012.) OPC UA:n tuomien toiminnallisuuksien avulla teollisuuden ohjausjärjestelemistä voidaan kerätä tietoa suoraan korkeamman tason tuotannon- ja toiminnanohjausjärjestelmiin.

Rinaldin (2016) 10 kohdan luonnehdinta OPC UA:sta

1. OPC UA ei ole protokolla

Tietokoneprotokolla on paketti/setti sääntöjä, jotka hallitsevat tiedon siirtymistä tietokoneelta toiselle. OPC UA määrittelee myös säännöt tietokoneiden väliselle tiedonsiirrolle, mutta sen tarkoitus on paljon suurempi kuin vain tiedon siirto tietokoneelta toiselle. OPC UA on kokonainen yhteentoimivuus. Se on arkkitehtuuri, joka järjestää tiedon, systeemin, laitteiden ja koko koneiston.

2. OPC UA on OPC:n seuraaja

OPC UA ratkaisee OPC:n puutteet. Nykypäivänä tietoa pitää siirtää erilaisten sulautettujen laitteiden välillä. Perinteistä OPC:ta ei ollut suunniteltu tähän. Perinteinen OPC on Microsoft-riippuvainen, eikä siinä ole tukea nykypäiväisille tietomalleille ja COM sekä DCOM, joita se käyttää ovat haavoittuvaisia erilaisille hyökkäyksille ja viruksille.

3. OPC UA tukee asiakas-palvelin arkkitehtuuria

OPC UA palvelin voidaan konfiguroida vastaanottamaan viestejä monilta asiakkailta, joten se ei ole yhden asiakkaan ja palvelimen välistä tiedonsiirtoa. OPC UA palvelin on paljon kehittyneempi järjestelmä tai komponentti kuin monet muut teknologiat, joita tavallisesti on käytetty.

4. OPC UA on alustariippumaton ja erittäin skaalautuva teknologia

Toisin kuin OPC on OPC UA rakennettu täysin alustariippumattomaksi. Kaikki OPC UA -komponentit on suunniteltu skaalautuviksi, myös OPC UA tietoturva, lähetys ja tietomalli ovat suunniteltu skaalautuviksi.

5. OPC UA voidaan yhdistää helposti IT järjestelmiin

Valtion säännöstely on pakottanut tiedonkeruun lisääntymiseen. Kaikkien näiden tekijöiden johdosta yhteyksiä toiminnanohjausjärjestelmiin ja pilvipalveluihin tulee kasvavissa määrin tehtaiden ohjausjärjestelmistä. OPC UA on kehitetty näiden yhteyksien toteuttamiseen. Palvelimet tukevat tiedonsiirtoa monien nykyaikaisten IT-järjestelmien kanssa. Palvelin voidaan yhdistää näihin järjestelmiin käyttämällä SOAP tai HTTP. OPC UA palvelimet voidaan myös konfiguroida käyttämään XML koodausta.

6. Hienostunut osoiteavaruusmalli

OPC UA:n osoiteavaruusmalli on huomattavasti hienostuneempi kuin EtherNet/IP Profinet IO, Modbus tai missä tahansa muussa teollisuuden automaatioprotokollassa esiintyvä. OPC UA:n osoiteavaruuden peruskomponenttia kutsutaan solmuksi (node). Solmu koostuu ominaisuuksista eli atribuuteista ja siitä miten se on yhdistetty toisiin solmuihin viittauksien ja erilaisten suhteiden kautta. Solmut jakavat yhteiset ominaisuudet toistensa kanssa.

7. OPC UA tarjoaa oikean tietomallin

. Koska oliosolmut (Object Node) luovat viittauksia toisiin olio solmuihin, jotka taas viittaavat toisiin solmuihin rajoittamattomissa määrin, voidaan muodostaa hierarkisia yhteysrakenteita. Tämän pohjalta voidaan määritellä systeemi, prosessi taitietomallin. Tietomalli on looginen esitystapa fyysisestä prosessista. Tietomalli on yksinkertaisesti hyvin esitetty tietorakenne vailla yksityiskohtia siitä,miten käyttää prosessin muuttujia, metadataa tai jotain muuta prosessin sisällä.

8. OPC UA ei ole tehdastason protokolla

OPC UA voidaan kutsua automaatiojärjestelmän www-sovelluspalveluksi tai, että se on automaatiojärjestelmien palvelukeskeinen arkkitehtuuri. Automaatiojärjestelmien maailmassa ohjausyksiköiden yhteysparadigma on, että järjestelmässä on hallintaohjausyksikkö (Master PLC) joka siirtää tietoa sisään ja ulos lapsiohjausyksiköistä (Slave

PLC). Tiedonsiirrossa käytetään erittäin yksinkertaista tapahtumapohjaista viestittelyä tai jonkinlaista yhdistettyä viestittelyä. Palvelimilla on lapsiohjausyksiköille ja solmuille sisään ja ulos menevälle tiedolle puskurit johon tiedot tallentuvat. Sisään tulevasta puskurista tieto siirtyy ohjausyksikölle itselleen ja ulospäin menevästä puskurista muille laitteille. OPC UA sijoittuu oikeastaan paradigman ulkopuolelle tai tarkemmin ottaen sijoittuu sen rinnalle. Se ei korvaa tätä paradigmaa vaan se laajentaa sitä. Se tuo uusia toiminnallisuuksia sekä luo uusia käyttötapauksia ja ohjaa uusia sovelluksia.

9. OPC UA on sertifioitu starndardi IEC 62541 (OPC Foundation, 2009)

Kullekin OPC UA -komponentille on sertifikaattiasiakirja, jonka avulla sertifioidaan, että laite läpäisee sertifikaatin testisarjan. OPC UA:n skaalautuvasta luonteesta johtuen ei ole mahdollista luoda yhtä testiä, joka kaikkien laitteiden on läpäistävä. Laitteet jotka tukevat vakioprofiilimääritelmää tukevat monia lisäominaisuuksia ja palveluita, jotka sopivat sulautettujen järjestelmien profiiliin. Skaalautuvien järjestelmien sertifiointi OPC UA:ssa ei ole ongelma. Jokainen OPC UA palvelinlaite raportoi tunnistetietonsa, jotka sisältävät tuotetut kuljetusmuodot, tietoturvaprofiilit, tuotetut palvelut ja tuotetut profiilit.

10. OPC UA on yhä kehittyvä teknologia

Käynnissä on erilaisia teknologioita omaksuva ja käyttöön ottava elinkaari, jota OPC UA myös seuraa. Ensimmäiset järjestelmät tulivat käyttöön vasta muutamia vuosia sitten. OPC UA on yhä käyttöönottovaiheessa oleva teknologia, koska innovaattorit ja varhaiset käyttäjät ovat sen pääasiallisina käyttäjinä.

OPC UA -standardi on seuraavan sukupolven teknologiaa tiedonsiirtoon tuotannonsuunnittelu- ja toiminnanohjausjärjestelmiin. Se takaa turvallisen ja luotettavan datan sekä prosessikerroksen esikäsiteltyjen tietojen kuljetukseen näihin ylemmän tason järjestelmiin. Standardi perustuu XML-syntaksiin, www-sovelluspalveluihin ja Palvelukeskeisen arkkitehtuurin, jonka kautta jaetaan tietoa monimutkaisempiin ylemmän tason järjestelmiin. (Tan & Kumppanit, 2009.)

OPC UA tarjoaa turvallisen, luotettavan ja tehokkaan viestintäinfrastruktuurin tiedonvaihtoon teollisuuden ohjausjärjestelmissä. Tiedonvaihto sisältää reaaliaikaista dataa kuten mittauksia, prosessin tapahtumia ja hälytyksiä. Lisäksi se tarjoaa mahdollisuuden historian keruuseen nykyisen datan ja tapahtumien lisäksi. (Goldschmidt & Mahnke, 2012.)

OPC UA komponenttien kehittämistä tukee tällä hetkellä erityisten sovelluskehitinten (Software Development Kit) käyttö. Nämä sovelluskehittimet tarjoavat välineen kehittää koodia, joka käsittelee luomista ja navigointia OPC

UA -tietomallissa eli arvojen muutoksien monitorointia, metodikutsujen määrittelyä sekä yhteyksien ja sessioiden hallintaa.(Goldschmidt & Mahnke, 2012.) OPC UA -komponentteja on siis mahdollista toteuttaa sovelluskehitinten avulla. Sovelluskehittimillä voidaan toteuttaa komponentteja, jotka mahdollistavat tiedon haun ja kirjoittamisen teollisuuden ohjausjärjestelmiin.

OPC UA:n toiminta perustuu asiakas-palvelin malliin (Client-Server Model), jossa asiakas pyytää dataa ja palvelin toimittaa datan. Asiakas voi lukea ja kirjoittaa dataa, mutta sillä on mahdollisuus myös tilata datamuutokset tai tapahtumailmoitukset. Tilaaminen tarkoittaa, että mahdolliset datamuutokset ja tapahtumailmoitukset toimitetaan asiakkaalle hänen pyyntönsä vastauksena. Lisäksi asiakas voi selailla palvelimen osoiteavaruutta ja lukea meta-dataa. Suurissa ja monimutkaisissa osoiteavaruuksissa asiakkaalla on myös mahdollisuus tehdä pyyntöjä, joilla kerätään tietoa osoiteavaruudesta. Esimerkiksi voidaan tehdä pyyntö, jolla kysytään, ovatko kaikki lämpötilasensorit mitanneet yli 25 C lämpötilan. (Goldschmidt & Mahnke, 2012.) Asiakas saa siis reaaliaikaista tietoa prosessista, jonka mittaustietoja on tallentunut ohjausyksikön osoiteavaruuteen. Osoiteavaruus voidaan lukea reaaliaikaisesti palvelimen kautta. Näiden pyyntöjen avulla taas voidaan kerätä laajemmin tietoa järjestelmän toiminnasta, esim. ovatko lämpötilat koko prosessissa oikeat tai ovatko kaikki anturit varmasti käytössä. Kyselyjen kautta voidaan myös kehittää aivan uutta tietoa prosessista.

Asiakkaat voivat tilata näitä tapahtumatietoja tilaamalla tietyn olion, joka on merkattu tapahtumailmoittajaksi. Riippuen siitä missä tämä tapahtujailmoittaja sijaitsee osoiteavaruudessa, suodattaa se automaattisesti ylimääräisen tiedon pois halutun tapahtuman tiedoista, joita ei tämän tapahtumanilmoittaja olion tietojen lisäksi tarvita. Lisäksi tapahtumailmoittajat voivat suodattaa tietoa niin, että tapahtumakentästä haetaan vain ne tiedot, jotka ovat kiinnostavia. Antaessaan tietoja tapahtumarakenteesta, palvelin tarjoaa tapahtumatyyppihierarkian. Se käyttää abstrakteja olioita määritellessään tapahtumatyyppihierarkiaa ja muuttujia määritellessään tapahtuman tietokenttiä. Hälytyksissä olioita käytetään määrittelemään hälytykset osoiteavaruudesta, jotta hälytykset voidaan konfiguroida näyttämään oikeanlainen hälytys, esimerkiksi määrittelemällä raja-arvo hälytykselle. (Goldschmidt & Mahnke, 2012.) Integroitu OPC UA palvelu mahdollistaa pääsyn ohjausyksikön koko osoiteavaruuteen, joka sisältää kaikki tiedot eri käyttötapauksia varten(Cupec & kumppanit, 2013).

OPC UA kommunikaatio perustuu TCP/IP tai HTTP/SOAP -protokolliin ja sitä voidaan kutsua palvelukeskeisen arkkitehtuuriksi (SOA). OPC UA:n protokollapinon toteutus perustuu TCP tai SOAP/HTTP -protokollaan. (Cupec & kumppanit, 2013.) OPC UA tiedonsiirtoliityntä yksinkertaistaa automaatiolaitteisiin käytettävän yhteyden luontiin tarvittavat protokollat ja lisää joustavuutta prosessidatan tulkintaan. Prosessidatan keruun helpottuminen on seurausta siitä, että sulautetuissa laitteissa on mahdollista käyttää OPC UA palvelimia.

Helppo siirtyminen klassisesta OPC -standardista uuteen OPC UA -standardiin on ollut yksi suurimmista huolen aiheista. Tämä on ollut erittäin

isossa roolissa, koska klassinen OPC-rajapinta on implementoitu yli 15 000 tuotteeseen ja on tärkeää hyödyntää tätä OPC UA -standardin käytössä. On tärkeää mahdollistaa OPC UA:n hyödyt klassista OPC -standardia käyttäen ja taata klassisen OPC:n toimittajille helppo tapa siirtyä tähän uuteen OPC UA -standardiin. (Girbeal & Kumppanit, 2010.)

KUVIO 3 Kääreohjelma (wrapper)

Kääreohjelma (Wrapper): Tässä tapauksessa COM -palvelimelle saadaan luotua yhteys OPC UA –asiakasohjelman (OPC UA Client) kautta. Kääreohjelma on siis todellisuudessa OPC –asiakas (OPC Client) niin kuin klassisessa OPC standardissa. Kääreohjelmalla on pääsy klassisen OPC:n palvelimeen ja samalla kääreohjelma itsessään on myös OPC UA palvelin (OPC UA Server), joka paljastaa UA:n tapahtumat, hallinnoi salausta ja purkamista, tietoturvaa ja lähetystä. Välityspalvelin: Kuten kääreohjelma, myös välityspalvelin tarjoaa nopean tavan sovittaa UA standardi klassisen OPC -standardin kanssa toimivaksi. Välityspalvelin on vastine kääreohjelmalle ja se mahdollistaa klassisen OPC -asiakasohjelman yhdistämisen OPC UA palvelimelle. Kuviossa 3 näkyy rakenne, jossa välityspalvelinkomponentti on OPC UA asiakas, jolla on pääsy OPC UA palvelimelle. Välityspalvelinkomponentti toimii siis samalla myös klassisena OPC –palvelimena, joka paljastaa COM-liitännän asiakasohjelmalle. Näin mahdollistetaan asiakkaille, jotka käyttävät klassista OPC standardia mahdollisuus käyttää OPC UA -palvelimia. (Girbeal & Kumppanit, 2010.) OPC UA -standardin implementoinnissa on siis otettu huomioon kummatkin standardit ja niitä voidaan käyttää helposti ristiin. Tämä mahdollistaa sen, että jos käytössä on klassinen OPC-standardi voidaan silti ottaa käyttöön laite, johon on implementoitu OPC UA -standardin mukainen rajapinta.

Sovelluskehittimet ja OPC UA pinot, joita on kehitetty eri ohjelmointikielillä mahdollistavat natiivien UA -asiakasohjelmien ja -palvelimien kehityksen. Tällä tavoin koko UA -standardin kaikki toiminnallisuudet voidaan toteuttaa sovelluksissa. Erilaiset käyttötapaukset ja tietomallit voidaan implementoida osoiteavaruuteen. (Girbeal & Kumppanit, 2010.) Sovelluskehitintyökalut mahdollistavat siis UA -palvelimien ja -asiakasohjelmien kehityksen erilaisilla ohjelmointikielillä ja näin mahdollistavat sen käytön erilaisilla alustoilla sekä monimutkaisempien ohjausjärjestelmien toteutuksen.

2.3 Tietoturva

Siitä lähtien, kun Stuxnet -haittaohjelma löydettiin 2010, on teollisuuden infrastruktuuri joutunut mitä monimutkaisimpien kyberhyökkäyksien kohteeksi kasvavissa määrin. Vaikka liiketoiminta ei olisi keskittynyt kriittisiin teollisuuden infrastruktuureihin kuten energiaan, veteen tai kuljetukseen, monet yritykset käyttävät SCADA tai teollisuudenohjausverkkoja joissain organisaationsa rakenteissa. Nämä verkot ovat kohteena hyökkäyksille, joita on odotettu ennemmin finanssi- ja hallinnollisiin instituutteihin. (Byres & Oulton, 2013.)

Aikoinaan teolliset tietoverkot toimivat eristettyinä omina tietoverkkoinaan ja käyttivät omia laitteitaan eristettyinä liiketoimintaverkoista ja Internetistä. Nykyisten vuosikymmenten aikana teollisuuden kenttäväylät (Fieldbus) ovat siirtyneet suljetuista verkoista kaupallisiin teknologioihin. Pysyäkseen moderneina, teollisuuden järjestelmät tarvitsevat nykyään jatkuvaa päivittämistä ja tietoa ulkopuolisista järjestelmistä ja Internetistä. Tämä on johtanut siihen, että nämä aiemmin eristetyt ohjausverkot eivät ole enää eristyksissä, vaan niihin on tullut laajasti yhteyksiä järjestelmän ulkopuolelta.(Byres & Oulton, 2013.)

Teollisuusjärjestelmät toimivat vaarallisissa ympäristöissä taukoamatta tai niiden on odotettu toimivan yhtäjaksoisesti pitkiä aikoja ilman, että niihin on suunniteltu tietoturvasäännöksiä. Operatiivisten määräysten käyttö ja turvallisuusmääräykset ovat kumonneet tietoturvasäännösten käyttöönoton aikaisemmin. Jopa tavalliset IT tietoturvastrategiat ovat olleet mahdottomia, koska ne ovat olleet konfliktissa teollisuuden standardien kanssa. Teollisuusjärjestelmien suojautumisessa on ajateltu ennen lähinnä tahattomista tietoverkon ongelmista johtuvia vikoja tai sisältäpäin tulevien hyökkäysten torjumista. (Byres & Oulton, 2013.) Suojautumisessa on ennen selkeästi suuntauduttu järjestelmien laitteiden vikaantumisesta johtuvien ongelmien ratkaisuun sekä teollisuusympäristön kulunvalvontaan, ettei ohjausverkkoon päästä käsiksi tai vaikuttamaan paikan päältä. Ulkopuolisen kyber-hyökkäyksen riski oli laskettu minimaaliseksi. Näin on ajateltu erityisesti voimalaitosteollisuuden alalla, kunnes 2010 Stuxnet haittaohjelma onnistui Iranissa häiritsemään uraanin rikastuttamisessa tarvittavaa linkousjärjestelmää. Se pystyttiin levittämään tähän suljettuun järjestelmään USB-tikun avulla. (Byres & Oulton, 2013.) Näin ollen suljetussa järjestelmässä, joissa ei oletettu tulevan ulkopuolista hyökkäystä, ei ollut minkäänlaista tietoturvaratkaisua, joka olisi tunnistanut uhan.

3 Unified Architecture toteutus

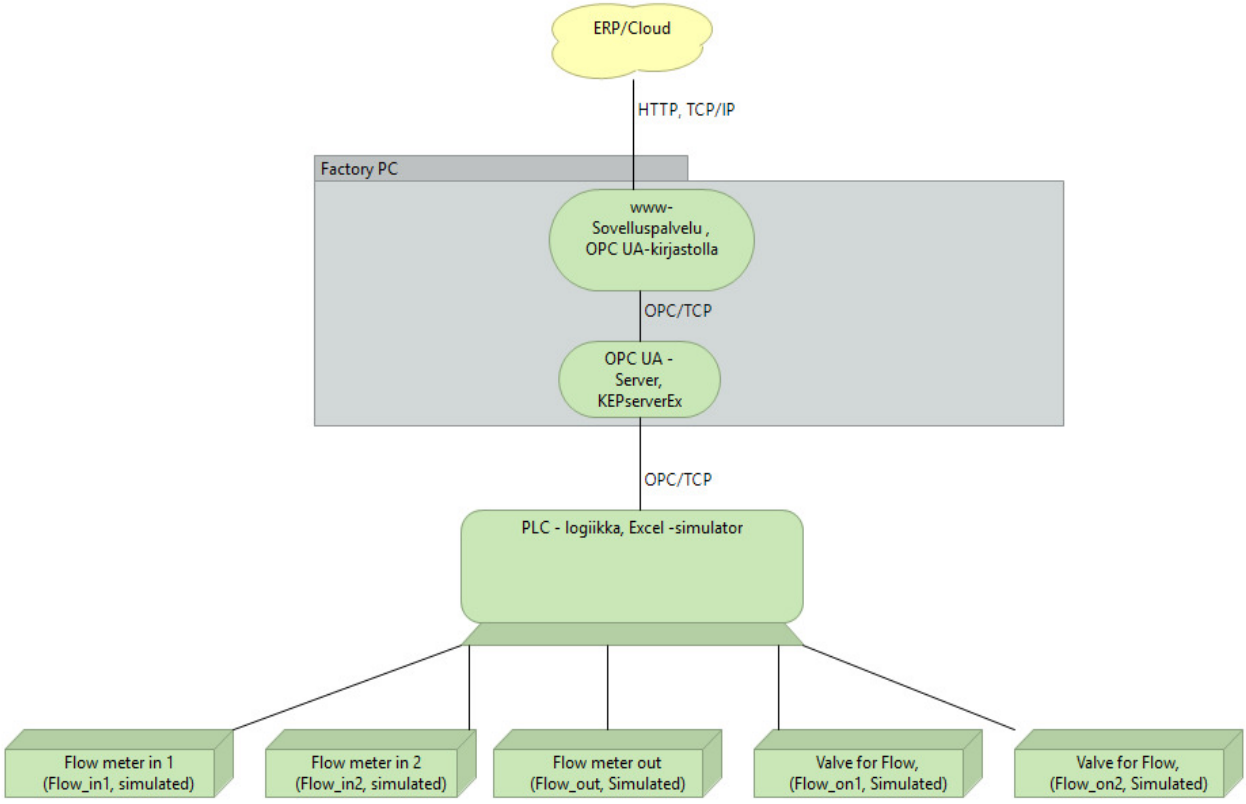
Tutkimusongelmaan haettiin ratkaisua myös käyttämällä konstruktiivista lähestymistapaa, jossa luodaan uusi toteutus käsiteltävästä asiasta olemassa olevien tietojen perusteella (Järvinen & Järvinen, 2004.). Tutkimuksessa pyrittiin vastaamaan, että miten teollisuuden ohjausjärjestelmän ja pilvipalvelun välinen tiedonkeruurajapinta voidaan toteuttaa itse TCP/IP- ja HTTP – protokollaan tukeutuen ja muokkaamalla valmiiden ratkaisujen lähdekoodia. Tutkimusongelmaa käsiteltiin paitsi käymällä läpi kirjallisuutta myös tekemällä toteutus, jossa luodaan Karnouksen (2011) esittämän www-sovelluspalveluratkaisun avulla rajapinta automaatiojärjestelmään, jota käyttäen pystytään hakemaan tietoa automaatiojärjestelmästä pilvipalveluun tai toiminnanohjausjärjestelmään.

Järjestelmän arkkitehtuurillinen toteutuksessa oltiin hyvin lähellä Nagorny & kumppanien (2012) ISA-95 standardin arkkitehtuuria. Toteutuksessa keskitytään automaatiojärjestelmän ohjausyksikköön luotavan rajapinnan toteutukseen ja pilvipalvelun toteutus jätetään pois. Pilvipalvelulle luodaan siis mahdollisuus hakea tietoa rajapinnan kautta automaatiojärjestelmästä, mutta varsinainen toteutus jää tulevaisuuteen. Toteutus perustuu OPC UA -standardiin sekä TCP/IP ja HTTP -protokollaan. Toteutuksessa mallinnetaan hypoteettinen automaatiojärjestelmä, josta OPC UA -palvelimen kautta tarjotaan yhteys OPC UA –asiakasohjelmalle, joka toteutettiin www-sovelluspalvelun yhteyteen. Tämän www-sovelluspalvelun kautta pystyttään HTTP-protokollaa käyttäen hakemaan hypoteettisen automaatiojärjestelmän keräämä tieto eli automaatiojärjestelmän muistiavaruus.

Konstruktiivisen osuuden toteutus aloitettiin etsimällä sopivat ratkaisut, joilla järjestelmä saataisiin toteutettua. Aluksi etsittiin ohjelmisto, joka tarjosi mahdollisuuden virtuaalisenjärjestelmän tai simulaattorin toteutuksesta. Tämän toteutuksen avulla pystyttiin mallintamaan teollisuuden automaatiojärjestelmän prosessia. Seuraavaksi selvitettiin miten OPC UA -palvelinohjelmisto toteutettaisiin, käytettäisiinkö siinä valmista ratkaisua vai muokattaisiinko jotain avoimen lähdekoodin ratkaisua. Tämän palvelinohjelmiston piti myös toimia yhteen aikaisemmin valitun virtuaalisenjärjestelmän/ simulaattorin kanssa.

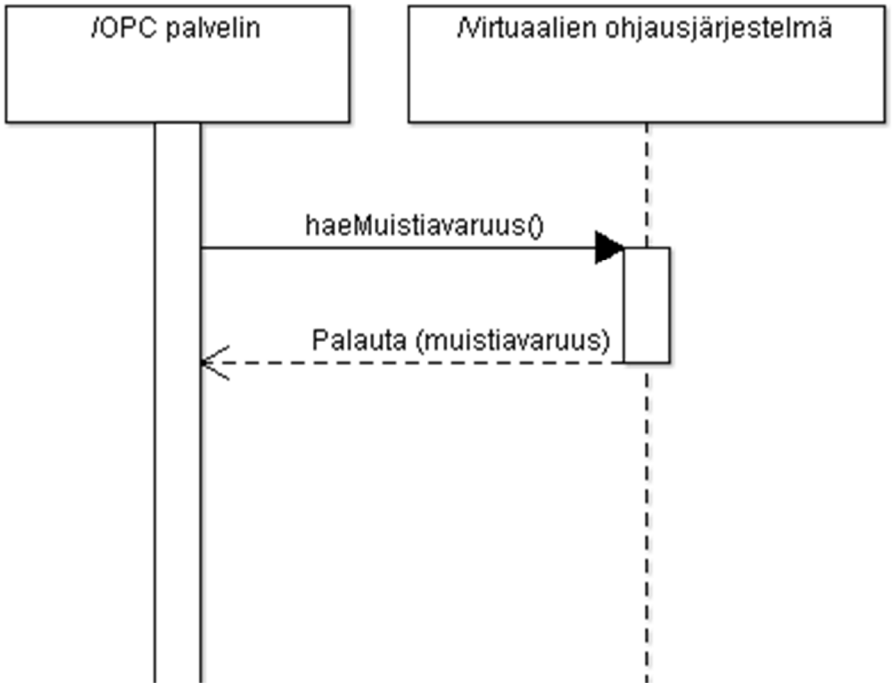
OPC UA -palvelinohjelmiston toteutuksessa piti löytää avoimenlähdekoodin ratkaisu, joka pystyttiin ottamaan käyttöön yhdessä www-sovelluspalvelun kanssa tai sen rinnalle. Valintaprosessin jälkeen pystyttiin keskittymään järjestelmän toteutukseen.

3.1 Toteutetun järjestelmän arkkitehtuuri



KUVIO 4 Hypoteettisen UA osatoteutuksen Komponenttikaavio

Toteutuksessa lähdettiin etenemään automaatiojärjestelmän mallintamisesta, jonka avulla pystyttiin demonstroimaan oikean automaatiojärjestelmän toimintaa ks. KUVIO 4. Aluksi määritettiin malli automaatiojärjestelmästä, johon on kerätty virtausmittarien ja venttiilien tietoja. Virtausmittarien kautta pystyttiin saamaan virtausnopeusdataa sisään ja ulos järjestelmästä sekä kytkimien avulla tieto virtauksen tilasta, eli onko virtauskytkin on- vai off -tilassa. Tämän jälkeen valittiin valmis OPC UA -palvelin toteutus, jonka kautta automaatiojärjestelmästä pystyttiin keräämään osoiteavaruudessa olevat virtaustiedot OPC UA -standardiin tukeutuen.

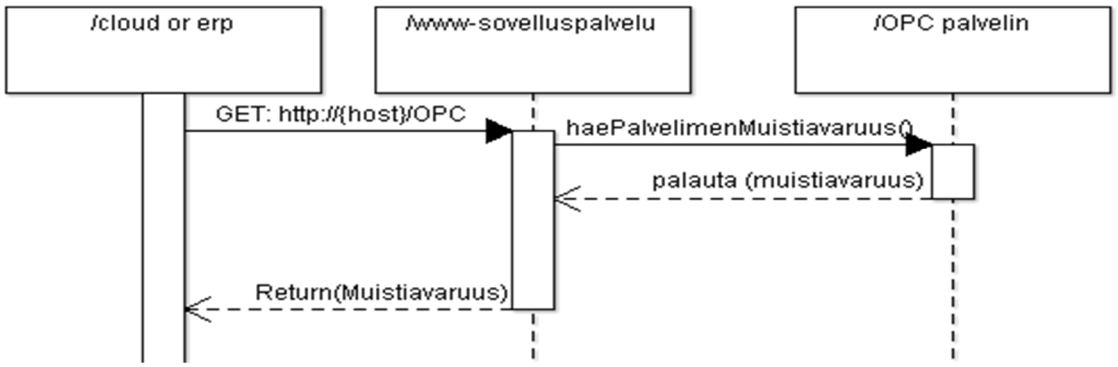


KUVIO 5 Sekvenssikaavio OPC UA -palvelin - hypoteettinen ohjausjärjestelmä

OPC UA -palvelimen avulla pystyttiin hakemaan virtuaalisen ohjausjärjestelmän muistiavaruus, johon oli kerätty virtaus- ja kytkintiedot ks. KUVIO 5. OPC UA -palvelimelle määriteltiin aika millä välillä se haki koko virtuaalisen

ohjausjärjestelmän sen hetkisen muistiavaruuden. Muutokset ohjausjärjestelmästä OPC UA -palvelimelle päivittyivät siis tämän määritellyn ajan puitteissa.

Tiedonsiirto OPC UA –palvelimen ja virtuaalisen ohjausjärjestelmän välillä hoitui OPC UA –protokollia käyttäen. OPC UA –palvelimella saatiin näin pääsy virtuaalisen ohjausjärjestelmän muistiavaruuteen, josta se sai haettua kaikki virtuaalisen ohjausjärjestelmän keräämät virtaustiedot. Oikeassa teollisuuden prosessissa olevasta ohjausjärjestelmästä tämä tiedon keruu tapahtuisi siis niin, että antureilta ja kytkimiltä tulevat tiedot tallentuisivat ohjausjärjestelmän muistiavaruuteen, josta se OPC UA –protokollaa käyttäen haettaisiin tietyn päivitysajan välein OPC UA – Palvelimelle. Toteutuksessa OPC UA –palvelimelle määriteltiin 100 ms:n päivitysaika hypoteettisen ohjausjärjestelmän muistiavaruuden hakemiseen. Päivitysajan olisi voinut laittaa huomattavasti nopeammaksi, mutta toteutuksessa ei virtuaalisiin mittaustietoihin tullut muutoksia kuin käsin niitä muuttamalla, joten 100 ms:n päivitysaika riitti tässä tapauksessa. Eli OPC UA –palvelin haki hypoteettisen ohjausjärjestelmän muistiavaruudesta tiedot 100 ms:n välein jaettavaksi eteenpäin.



KUVIO 6 Sekvenssikaavio OPC UA – palvelin ja www-sovelluspalvelu

Tämän jälkeen luotiin OPC UA –asiakasohjelman ja Visual Studio Web API www-sovelluspalvelun yhteistoiminto, joka pystyi hakemaan OPC UA –palvelimen muistiavaruuden ja lähettämään sen eteenpäin HTTP-protokollan Get –pyynnön vastauksena. Tämä www-sovelluspalvelun ja OPC UA -asiakasohjelman yhteistoiminto toteutettiin tekemällä www-sovelluspalvelu sovellus ASP.NET käyttäen ja lisäämällä sinne OPC UA -kirjasto, jota ohjelmallisesti käytettiin www-sovelluspalvelun kautta. Tämä vaati siis jonkin verran ohjelmointityötä ja OPC UA -kirjaston toiminnan selvittämistä, jotta sillä pystyttiin luomaan yhteys OPC UA -palvelinohjelmistolle ja hakemaan tietoa sieltä.

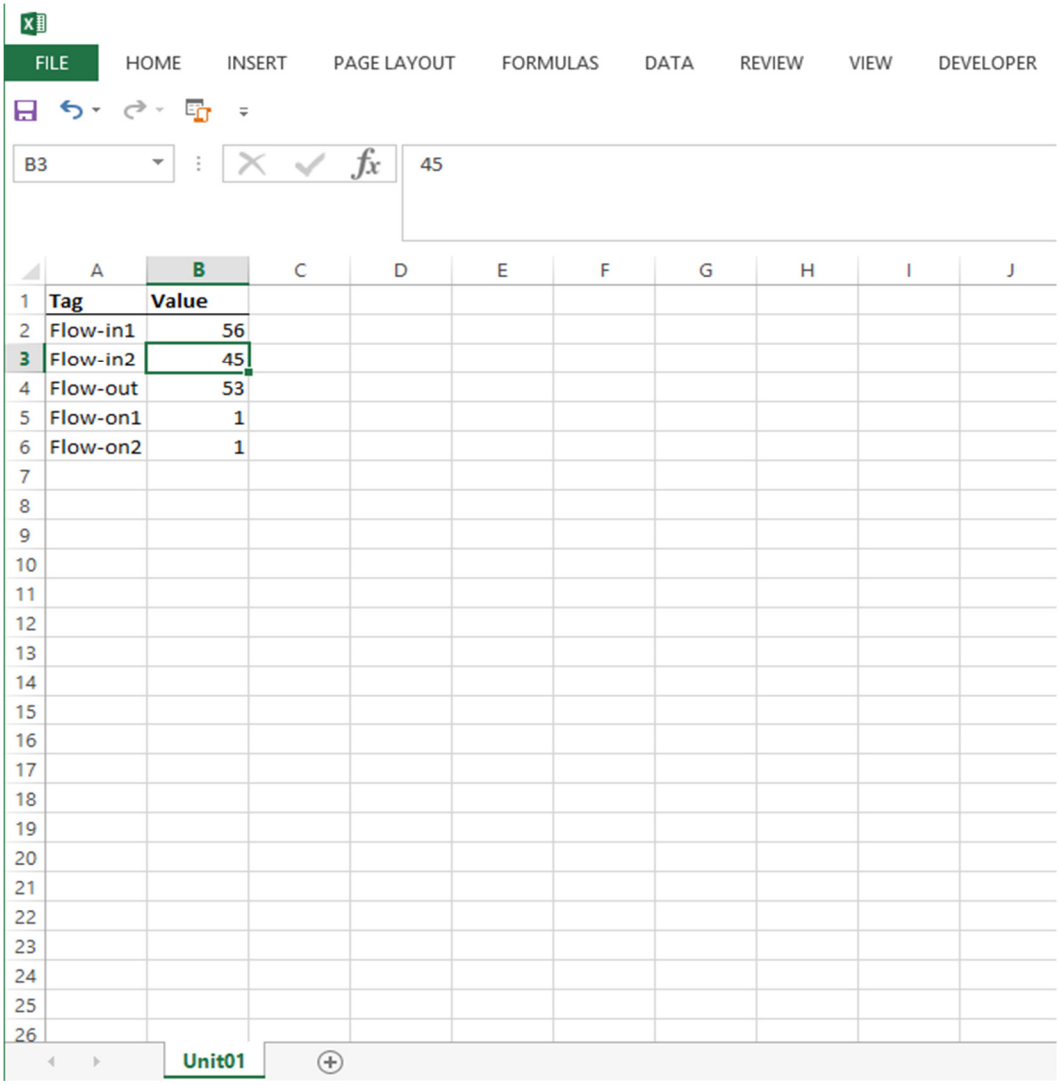
OPC UA –asiakasohjelmalla pystyttiin siis OPC UA –palvelimen hakeman muistiavaruuden kautta olemaan yhteydessä hypoteettiseen ohjausjärjestelmään ja sen keräämään muistiavaruuteen.

OPC UA –asiakasohjelmalla pystyttiin myös toteuttamaan OPC UA –palvelimen kautta tiedon kirjoitus hypoteettiseen ohjausjärjestelmään. Yhteistoiminto toteutettiin Visual Studio:n .NET Web API www-sovelluspalvelulla, johon lisättiin OPC UA –asiakasohjelman OPC UA-kirjasto, jonka avulla pystyttiin hakemaan tietoa OPC UA –palvelimelta. Luodun www-sovelluspalvelun avulla muistiavaruus pystyttiin HTTP-protokollan avulla lukemaan myös eteenpäin toiminnanohjausjärjestelmään (Enterprise Resource Planning ,ERP) tai pilvipalveluun (Cloud Service). Järjestelmä toimii siis niin, että OPC UA –palvelimen hypoteettisesta ohjausjärjestelmästä hakemaan tietoon päästiin käsiksi www-sovelluspalvelun kautta. Hypoteettisen automaatiojärjestelmän keräämä tieto pystyttiin hakemaan HTTP-protokollaa käyttäen pilvipalveluun tai toiminnanohjausjärjestelmään. Tämä on kuvattu kuviossa 6.

Toteutuksen koko arkkitehtuuri on esitetty kuvion 4 komponenttikaaviossa. Siitä nähdään, että ohjausjärjestelmä sijaitsee erillään teollisuustietokoneesta, joka taas pitää sisällään toteutukset OPC UA -palvelimelle ja www-sovelluspalvelulle. Ohjausjärjestelmä voi siis sijaita jossain teollisuuden prosessin alueella, jossa prosessiin liitetyt eri anturit keräävät tietoa ohjausjärjestelmään. Ohjausjärjestelmästä taas haetaan tietoa liittämällä se teollisuus-pc:hen, josta OPC UA -palvelimen kautta www-sovelluspalvelu pystyy lukemaan sen muistipaikat HTTP-protokollan avulla halutulle palvelulle. Toteutuksessa www-sovelluspalvelulle lähetettiin HTTP GET –pyyntö, jonka vastauksena se palautti hypoteettisen automaatiojärjestelmän muistiavaruuden kaikki muuttujat ja niiden sen hetkiset arvot.

3.2 Järjestelmän toteutus

Automaatiojärjestelmän virtuaalinen mallintaminen ja OPC UA -palvelin päätettiin toteuttaa Kepware Technologies:n KEPServerEx OPC UA -Palvelimen avulla.(KepWare, 2016.) WWW-sovelluspalvelun ja OPC UA- asiakasohjelman yhteistoiminto taas toteutettiin käyttämällä Unified Automation GmbH:n tarjoaman .NET pohjaisen sovelluskehittimen OPC UA-kirjastoa.(Unified Automation , 2016.) Tämä .NET -pohjainen sovelluskehitintyökalu OPC UA –asiakasohjelman laatimiseen tarjosi joko valmiin ohjelmistopaketin tai mahdollisuuden toteuttaa OPC UA -asiakasohjelmisto mukana tulleiden lähdekoodien avulla. Järjestelmän toteutuksessa käytettiin lähdekoodissa olevaa OPC UA-kirjastoa, joka otettiin käyttöön Microsoft:n Visual Studio:lla toteutetussa ASP.NET pohjaisessa www-sovelluspalvelu ratkaisussa. Www-sovelluspalvelu lisäsi ratkaisuun mahdollisuuden HTTP GET- pyynnöillä tehtävään tiedonhakuun virtuaalisesta automaatiojärjestelmästä.

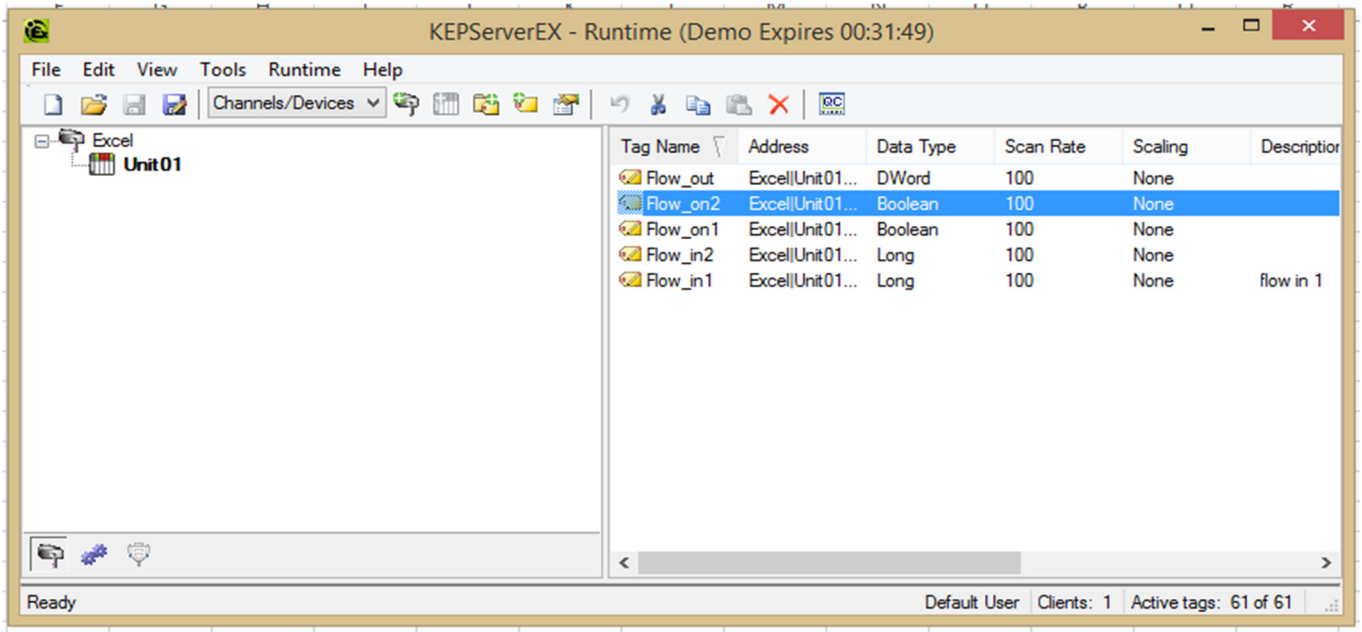


	A	B
1	Tag	Value
2	Flow-in1	56
3	Flow-in2	45
4	Flow-out	53
5	Flow-on1	1
6	Flow-on2	1

KUVIO 7 Excelillä toteutettu ohjausyksikön malli.

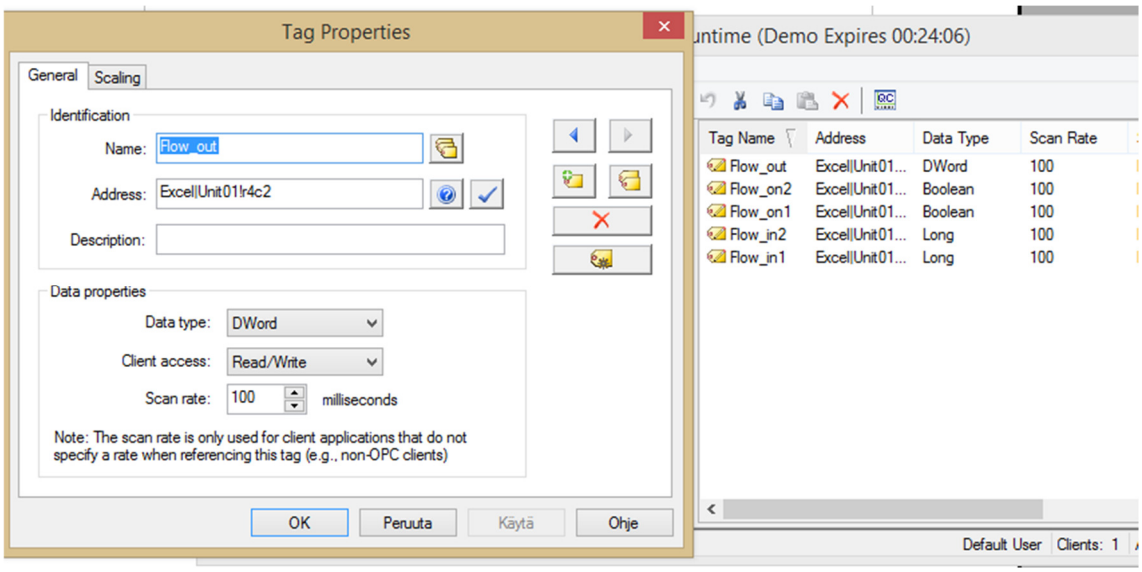
KEPServerEx tarjoaa oikean ohjausyksikköön luotavan yhteyden lisäksi mahdollisuuden toteuttaa virtuaalisen automaatiojärjestelmän joko simulaattorin tai Excel -ohjelmiston avulla. Jälkimmäisessä? voidaan itse määritellä laitteiden tunnisteet (Tag), attribuutit, attribuuttien tyypit ja niiden toiminnallisuudet sekä nopeus, jolla näitä tietojen päivittymistä seurataan OPC UA -palvelimen toimesta. KEPServerEx -simulaattorin ongelmaksi nousi se, että sen simuloima arvoja ei voinut itse määritellä, vaan se generoi jatkuvalla syötöllä eri arvoja sinne määritellyistä tunnisteista, eli tässä tapauksessa järjestelmätiedoista. Valitsimme Excel-pohjaisen toiminnon, joten saimme itse määriteltyä laitteiden tunnisteet ja muunneltua itse niiden arvoja, jotka siis tapauksessamme mallinsivat virtausmittareita ja -kytkimiä. Excel-tiedostoon luotiin ohjausjärjestelmän ohjaimen välilehti Unit01 ja tämän sisälle virtuaaliset virtausmittarit ja –kytkimet. Tämä on esitetty kuviossa 7. Ohjausjärjestelmän ohjausyksikön nimeksi tuli siis Unit01, johon simuloidun automaatiojärjestelmän tiedot kerääntyivät. Excelin välilehdellä pystyttiin simuloimaan oikean ohjausyksikön muistipaikkoja, josta järjestelmän tietoja pystyttiin hakemaan eteenpäin OPC UA -

protokollaa käyttäen. Excelillä Flow-in tunnisteella simuloitiin järjestelmään tulevia virtausmittarien tietoja, Flow-on tunnisteella taas virtauskytkimien tietoja, jotka kuvaavat ovatko venttiilit auki ja pääseekö järjestelmään näin virtaamaan mitään. Vastaavasti Flow-out tunniste simuloi järjestelmästä pois menevää virtausta. Tällä tavoin pystyttiin simuloimaan oikeaa prosessia, jossa olisi ollut virtausmittareita ja kytkimiä, jotka on esitetty kuvion 7 Excel:llä toteutetussa ohjausyksikön mallissa.



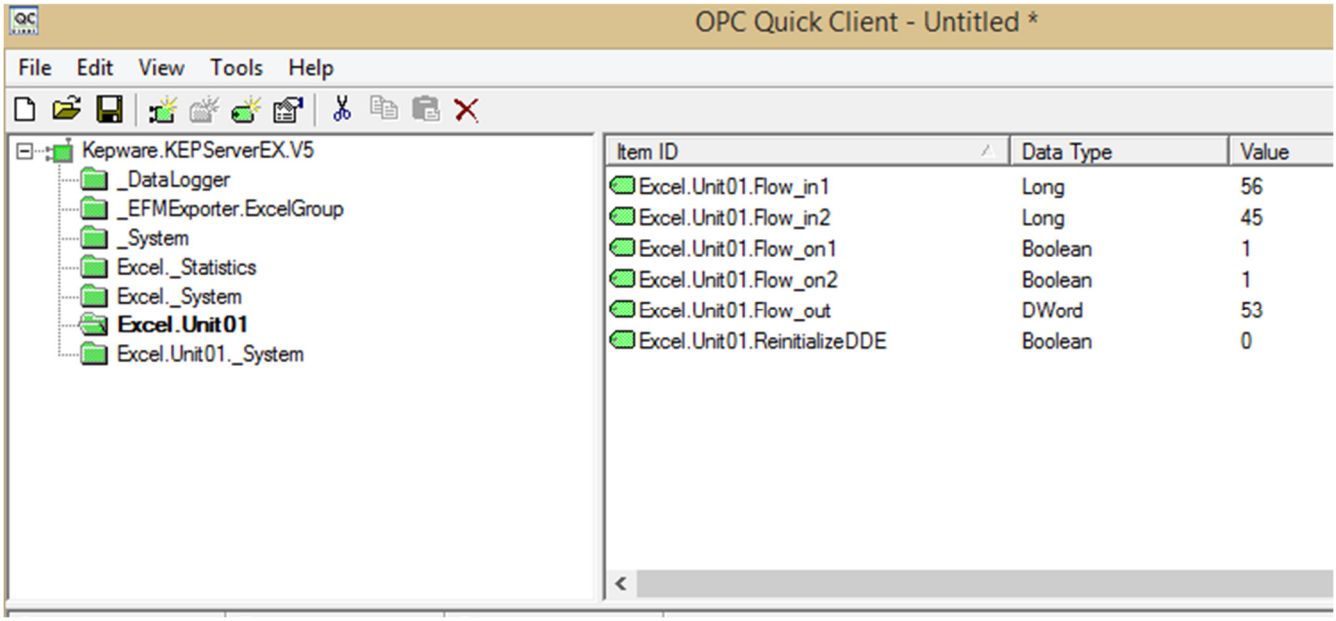
KUVIO 8 Ohjaimen lisäys KepServerEx ohjausjärjestelmään

Hypoteettisen ohjausjärjestelmän luonnin jälkeen järjestelmälle piti konfiguroida OPC UA -palvelin. KepServerEx OPC UA -palvelimelle täytyi lisätä nyt tämä virtuaalinen Unit01 ohjausyksikkö. KepServerEx OPC -Palvelin ohjelmaan lisättiin uusi Kanava (Channel) nimeltä Excel ja sen laiteohjaimeksi (Device driver) määriteltiin DDE Client, jota käytetään Excel-pohjaisessa simuloinnissa. Tämän jälkeen sille lisättiin laite (Device), joka tässä tapauksessa oli Unit01 ks KUVIO 8. Näin oli mallinnuksessa kanavana Excel, jonka sisällä mainitaan laite Unit01, joka meidän tapauksessamme oli Excel-tiedoston välilehti Unit01. Ohjausjärjestelmäosuus OPC UA –palvelimelle oli näin luotu.



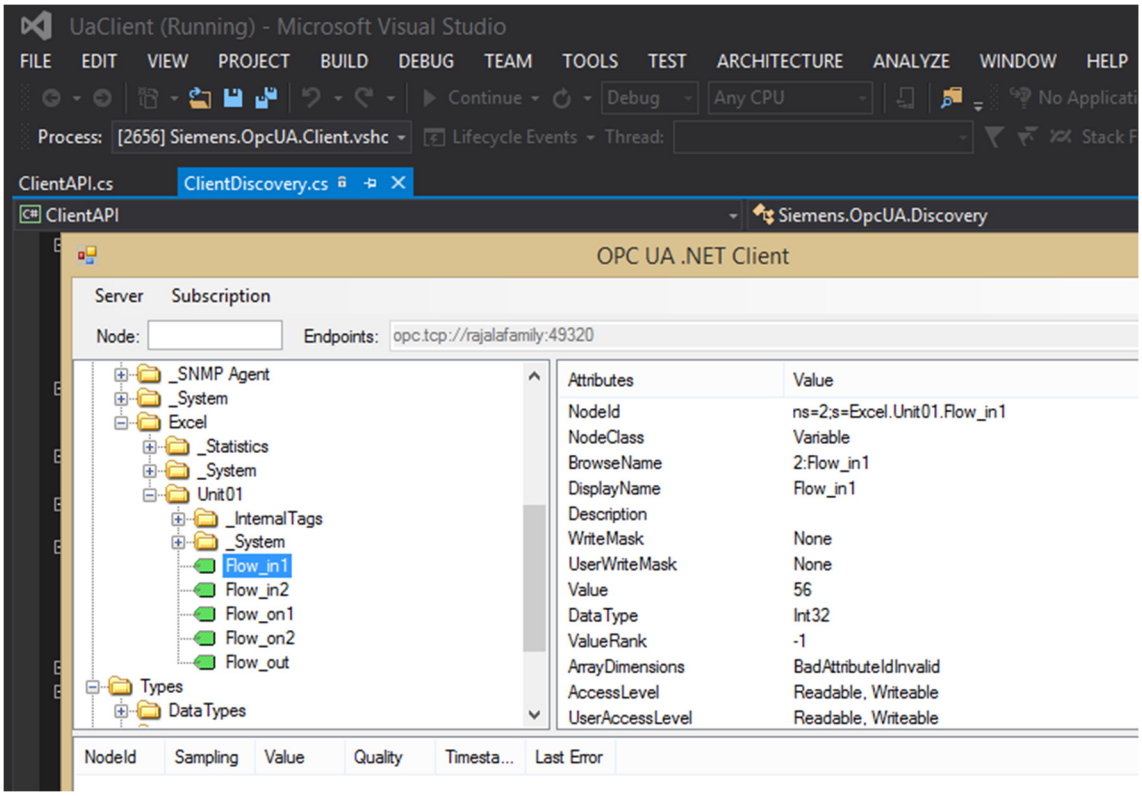
KUVIO 9 KepserverEx -käyttöliittymäkuva tunnisteiden ja virtuaalisten komponenttien lisäyksestä

Tämän jälkeen OPC UA -palvelin yhdistettiin Excel:n Unit-välilehdellä mallinnettujen virtuaalistenvirtausmittareiden ja kytkimien arvoihin. Jokainen tunniste lisättiin OPC UA -palvelimen Unit01-laitteelle ja sille piti määritellä uudestaan nimi sekä osoite. Osoitteeseen merkattiin missä sen arvo sijaitsee Excel-tiedostossa. Tämän jälkeen merkattiin välilehden nimi, jossa tämä tunniste sijaitsee ja kohta, millä rivillä ja sarakkeella se on. Kuviossa 9 oleva virtausmittarin ulos menevä arvo Flow_out -tunniste oli Excelin välilehdellä Unit01 ja sijaitsi rivillä 4 sarakkeessa 2, joten sen sijainti merkataan osoitteeseen Excel|Unit01!r4c2. KepServerEx OPC UA -palvelin osaa automaattisesti hakea samalla tietokoneella auki olevasta Excel-tiedostosta välilehdeltä Unit01 osoitekenttään merkatusta sijainnista arvon, koska tämän kanavan laitteen tyypiksi oli määritelty DDE client. KepServerEx palvelinohjelmistossa voidaan määritellä Data properties -kohdassa haettavan tunnisteen tietotyyppi sekä OPC UA -asiakasohjelman kirjoitus- ja lukuoikeudet tähän hypoteettisen ohjausjärjestelmän muistipaikkaan ja tietojen päivitysnopeus eli kuinka useasti tätä tietoa haetaan ohjausjärjestelmän ohjausyksiköltä OPC UA –palvelimen toimesta.



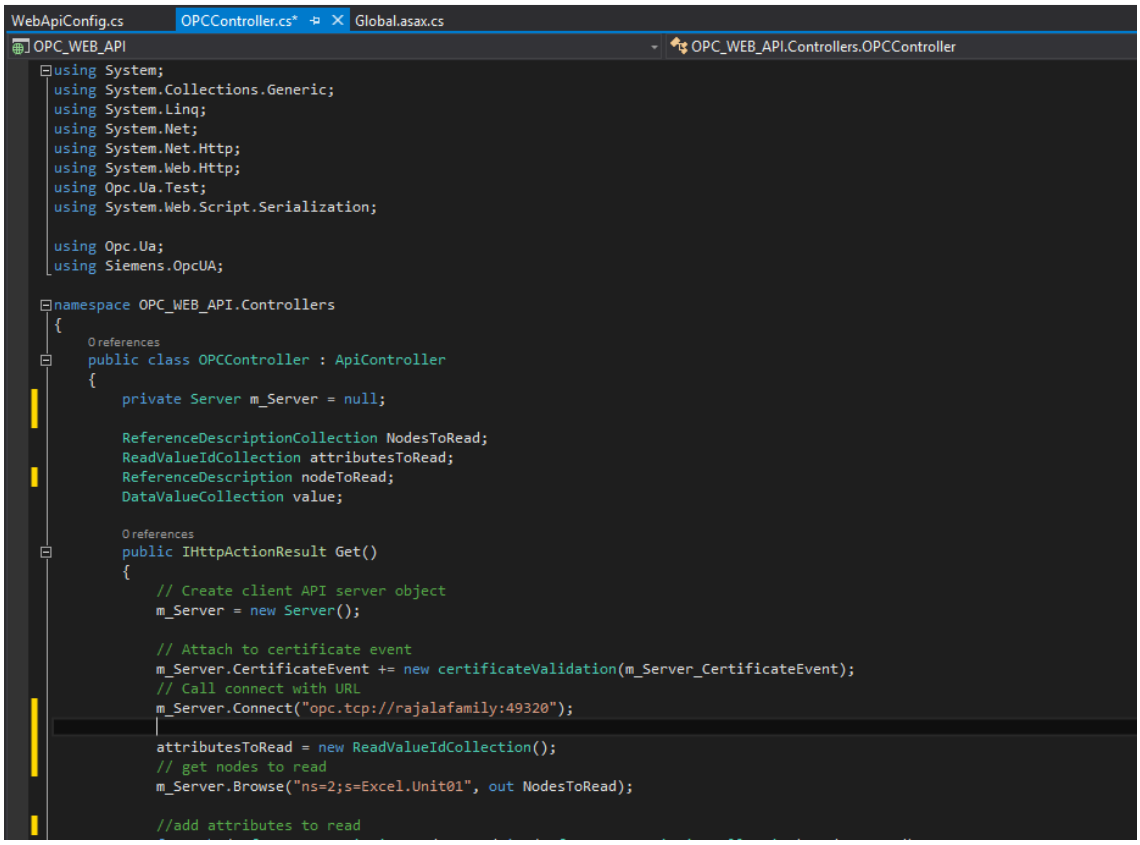
KUVIO 10 KepserverEx OPC quick client -asiakasohjelma

KepServerEx -palvelimen käynnistyksen jälkeen voitiin tarkistaa KepServerEx OPC quick client -asiakasohjelmiston avulla, että OPC UA -asiakasohjelmistolla pystyttiin hakemaan nämä tiedot Excelissä toteutetusta virtuaalisesta ohjausjärjestelmästä KepServerEx:n -palvelimen kautta. OPC quick client -asiakasohjelmistolla pystyttiin hakemaan virtuaalisen ohjausjärjestelmän sisältö, joka oli määritelty palvelinohjelmistolla luettavaksi ohjausjärjestelmän sisältä. Tämä on esitetty kuviossa 10. OPC quick client –asiakasohjelma osasi hakea tiedot automaattisesti OPC UA palvelimelta eikä sitä tarvinnut konfiguroida erikseen. Quick client -asiakasohjelmistolla pystyttiin selvittämään KepServerEx –palvelinohjelmiston, Excelin ja OPC UA -asiakasohjelmiston välisen yhteyden tila ja tarkistamaan sen toimivuus.



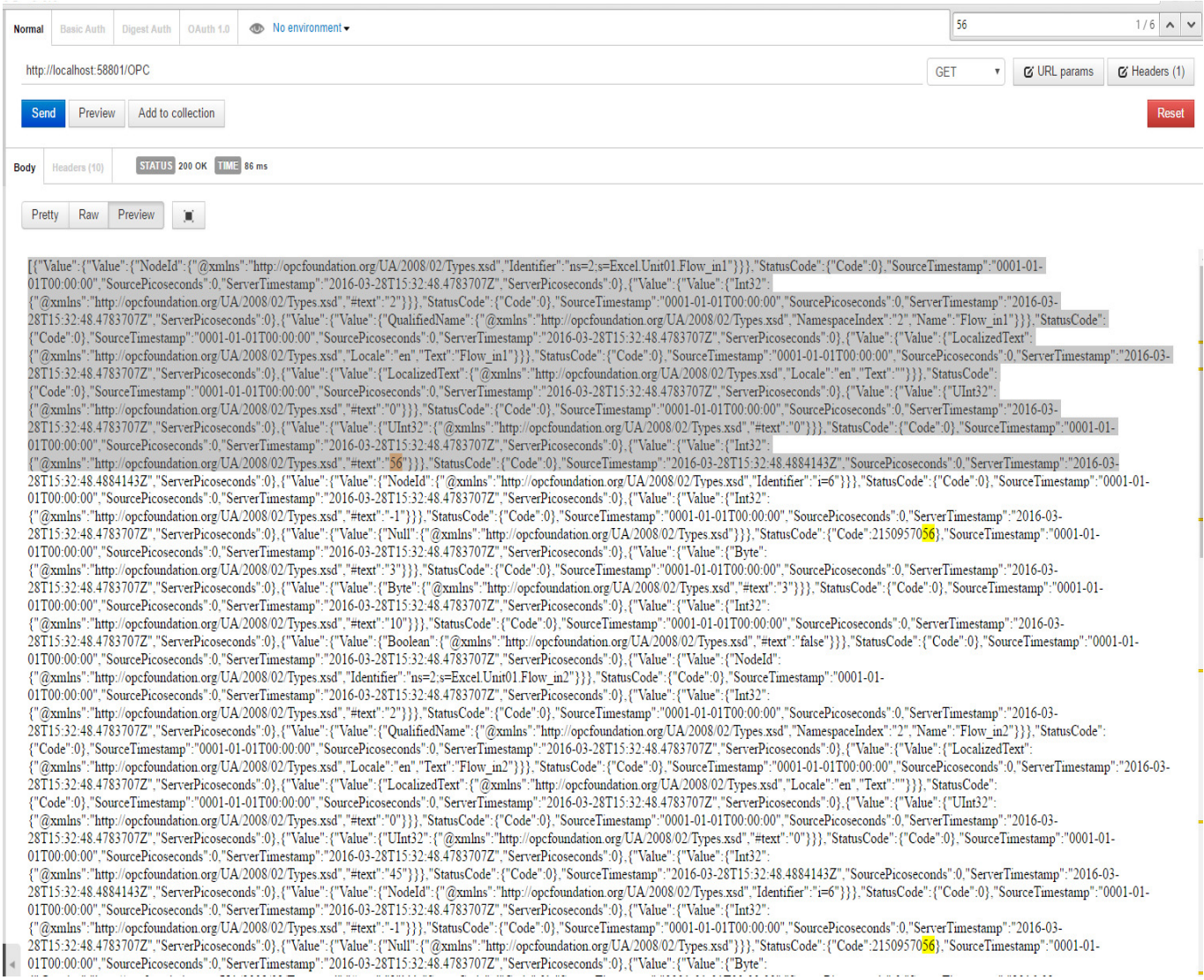
KUVIO 11 OPC UA -asiakasohjelma

OPC UA -asiakaspalvelinohjelmiston toteutuksessa taas käytettiin Unified Automation GmbH:n sivuilta saatavaa .NET pohjaista sovelluskehitintyökalua.(Unified Automation ,2016.) Tämän mukana tuli kaksi esimerkkitoteutusta, joista toisen lähdekoodin pohjalta toteutusta lähdettiin tekemään. OPC UA -asiakasohjelman ja palvelimen välistä salausta ei käytetty tässä toteutuksessa, joten yhteys onnistui suoraan yhdistämällä OPC UA -asiakasohjelmistolla KepServerEx:n OPC UA –palvelimelle, eikä sille tarvinnut luoda erillistä sertifikaattia. Kepserverin UA -palvelimen osoite pystyttiin suoraan lisäämään OPC UA -asiakaspalveluohjelmalle ja näin UA -palvelimen kautta pystyttiin lukemaan palvelimen keräämä osoiteavaruus ohjausjärjestelmän muistipaikoista. Tämä on esitetty kuviossa 11. Tässä vaiheessa yhteys virtuaaliselta ohjausjärjestelmältä OPC UA palvelimen kautta OPC UA -asiakasohjelmalle oli saatu toteutettua ja pystyttiin lukemaan ja myös kirjoittamaan tietoa suoraan ohjausjärjestelmään. Toteutuksesta puuttui siis vielä www-sovelluspalvelun osuus.



KUVIO 12 .NET WebAPI –ratkaisu ja OPC UA -kirjasto

WWW-sovelluspalvelua varten luotiin Visual Studiolla .NET WebAPI –ratkaisu. Tälle WWW-sovelluspalvelulle tehtiin yksi GET –reititys, jota kutsumalla käytettiin Unified Automation GmbH:n OPC UA- kirjastoa. Kirjaston metodien avulla avattiin ensin yhteys OPC UA -palvelimelle, tämän jälkeen haettiin muistipaikat ja sen perusteella koko muistiavaruus. OPC UA –kirjaston hakema palvelimen muistiavaruus palautettiin GET –pyynnön vastauksena. Toteutuksessa oli tiedossa määritelty ohjausjärjestelmän ohjausyksikön nimi, joten kaikki ohjausyksiköltä OPC UA –palvelimen hakema muistiavaruus ja sen sisältö voitiin hakea tämän tiedon perusteella. WWW-sovelluspalvelulle lähetettäessä HTTP GET- pyyntö, www-sovelluspalvelin loi OPC UA –kirjaston avulla yhteyden OPC UA –palvelimelle ja haki sieltä ohjausyksikön Unit01 muistiavaruudessa olevat tiedot. Tämä on esitetty kuviossa 12.



KUVIO 13 HTTP GET -vastaus koko muistiavaruuden sisällöstä

HTTP Get-pyynnön vastauksena saatiin koko OPC UA -palvelimen hakema hypoteettisen automaatiojärjestelmän muistiavaruuden sen hetkinen sisältö. Muistiavaruuden kaikki tiedot, jotka tuolta ohjausyksiköltä Unit01 oli kerätty, lähetettiin vastauksena tälle Get –pyynnölle. Pyynnön vastaus koko muistiavaruudesta on esitetty kuviossa 13.

Näin koko yhteys WWW-sovelluspalvelulta aina automaatiojärjestelmälle oli luotu ja tieto, jonka automaatiojärjestelmä oli kerännyt pystyttiin hakemaan edelleen eteenpäin WWW-sovelluspalvelua käyttäen. Näin tieto voitiin toimittaa eteenpäin halutuille palveluille tai sovelluksille HTTP –protokollaa käyttäen. Tutkimusongelmaan saatiin ratkaisu tällä toteutuksella. Ratkaisu vastasi karnouksen (2011) esittämää WWW-sovelluspalveluratkaisua missä automaatiojärjestelmään toteutettiin rajapinta, jota käyttäen pystyttiin hakemaan tietoa automaatiojärjestelmästä pilvipalveluun tai toiminnanohjausjärjestelmään.

3.3 Tutkimuksen tulos

Konstruktiivisen tutkimuksen tuloksena syntyi rajapintatotutus jossa yhdistettiin OPC UA ja www-sovelluspalvelu. Rajapintatoteutusta hyödyntäen pystyttiin hakemaan tietoa automaatiojärjestelmän muistiavaruudesta hyödyntäen TCP/IP ja HTTP –protokollia. Tutkimusongelman ratkaisussa käytettiin hyväksi Karnouksen (2011) esittämää WWW-sovelluspalveluratkaisua sekä hyödynnettiin Nagorny & kumppanien (2012) ISA-95 standardin arkkitehtuuria. Toteutuksessa onnistuttiin luomaan automaatiojärjestelmän ohjausyksikölle rajapinta, jota hyödyntäen pystyttiin hakemaan www-sovelluspalvelulla HTTP Get- pyynnöllä tietoa automaatiojärjestelmästä. Näin oltiin luotu mahdollisuus pilvipalvelulle sekä erilaisille osajärjestelmille hakea tietoa automaatiojärjestelmän ohjausyksiköltä. Toteutuksessa käytetty OPC UA –standardi ja www-sovelluspalvelun yhteistoiminto osoittautuivat hyväksi vaihtoehdoksi toteuttaa rajapinta.

Tutkimus rajattiin koskemaan automaatiojärjestelmän ohjausyksikön rajapinnan luomista, joten pilvipalvelulle luotiin vain mahdollisuus hakea tietoa rajapinnan kautta. Toteutusta hyödyntäen teollisuuden ohjaus- ja valvontajärjestelmistä voitaisiin hakea tietoa jatkojalostusta ajatellen. Aikaisemmin nämä asiat ovat tapahtuneet järjestelmän sisällä samassa toimitilassa, jossa järjestelmä sijaitsee ja tässä suljetussa ohjausverkossa on tapahtunut kaikki tiedonkeruu. Toteutuksen avulla automaatiojärjestelmän historiatiedot voitaisiin siirtää pilvipalvelulle tai muulle osajärjestelmälle, joka sijaitsee eri toimitilassa järjestel män kanssa.

Yhteenveto

Tutkimuksessa selvitettiin onko pitkään suljettuihin teollisuuden ohjaus- ja valvontajärjestelmiin mahdollista toteuttaa www-sovellusrajapinta TCP/IP- ja HTTP-protokollaa käyttäen. Tutkimuksen aihe oli rajattu koskemaan tiedonkeruuta automaatiojärjestelmästä ja tavoitteena oli selvittää kuinka tiedonkeruuseen tarvittava www-sovellusrajapinta voitaisiin toteuttaa käyttäen näitä protokollia.

Tutkimuksen varsinainen tarkoitus oli vastata kysymykseen miten teollisuuden ohjausjärjestelmän ja pilvipalvelun välinen tiedonkeruurajapinta voidaan toteuttaa TCP/IP ja HTTP-protokolliin tukeutuen..

Tutkimus toteutettiin kirjallisuuskatsauksena ja konstruktiivisena tutkimuksena. Kirjallisuuskatsauksessa pyrittiin esittelemään tämän hetkinen tieto www-sovelluspalveluratkaisujen käytöstä automaatiojärjestelmien rajapintaratkaisujen luomiseen. Kirjallisuuden avulla pyrittiin selvittämään miten tätä aihetta on tähän mennessä tutkittu ja minkälaisia tutkimustuloksia aiheesta löytyi. Kirjallisuuden avulla selvitettiin myös tutkimuksen keskeiset käsitteet sekä miten teollisuuden standardit ja tietoturva tulisi ottaa huomioon toteutuksessa. Kirjallisuuskatsaus päädyttiin jakamaan kolmeen osioon, joissa selvitettiin aiempia tutkimuksia aiheesta. Nämä osiot olivat langattomat anturiverkot ja pilvipalvelut, palvelukeskeisen arkkitehtuurin käyttö automaatiojärjestelmien ja pilvipalvelun yhdistämisessä sekä tietoturva.

Tutkimusongelmaan haettiin vastausta konstruktiivisella tutkimuksella. Konstruktiivisessa osuudessa esitettiin, että miten tiedonsiirto TCP/IP ja HTTP-protokollaa käyttäen voidaan toteuttaa automaatiojärjestelmältä WWW-sovelluspalvelua käyttäen. Ratkaisuun toteutuksessa käytettiin Karnouksen (2011) esittämää WWW-sovelluspalveluratkaisua, missä automaatiojärjestelmään toteutettiin rajapinta, jota käyttäen pystyttiin hakemaan tietoa automaatiojärjestelmästä pilvipalveluun tai toiminnanohjausjärjestelmään. Järjestelmän arkkitehtuurillinen toteutus perustui Karnouksen ja kumppanien (2012) esittelemään versioon ISA-95 standardin arkkitehtuurista. WWW-sovelluspalvelu ja ISA-95 standardin ratkaisut esiteltiin tutkimuksen kirjallisuuskatsauksessa otsi-

kon alla, jossa käytiin läpi palvelukeskeisen arkkitehtuurin käyttö automaatiojärjestelmien ja pilvipalvelun yhdistämisessä.

Ratkaisun arkkitehtuuri toteutettiin niin, että automaatiojärjestelmästä tehtiin hypoteettinen mallinnus, jonka avulla pystyttiin demonstroimaan oikean automaatiojärjestelmän toimintaa.

OPC UA -palvelimen avulla pystyttiin hakemaan virtuaalisen automaatiojärjestelmän sisältämän ohjausjärjestelmän muistiavaruus. Ohjausjärjestelmän muistiavaruus sisälsi kerättyjä virtaus- ja kytkintietoja automaatiojärjestelmästä, jota virtuaalinen mallinnus esitti. OPC UA -palvelin haki virtuaalisen ohjausjärjestelmän muistiavaruuden määritellyn aikavälin mukaan. Ohjausjärjestelmän muutokset päivittyivät OPC UA -palvelimelle tämän määritellyn ajan puitteissa. Tämän jälkeen OPC UA-asiakaskasohjelman ja Visual Studio WebAPI www-sovelluspalvelun yhteistoiminnolla pystyttiin hakemaan OPC UA -palvelimen keräämä muistiavaruus ja lähettämään se eteenpäin HTTP-protokollan Get-pyynnön vastauksena. Yhteistoiminto toteutettiin Visual Studio:n .NET Web API www-sovelluspalvelulla, johon lisättiin OPC UA –asiakasohjelman OPC UA-kirjasto, jonka avulla pystyttiin hakemaan tietoa OPC UA –palvelimelta. Luodun www-sovelluspalvelun avulla mahdollistettiin, että muistiavaruus pystyttiin HTTP-protokollan avulla lukemaan eteenpäin muihin järjestelmiin.

Automaatiojärjestelmän virtuaalinen mallintaminen ja OPC UA -palvelin toteutettiin Kepware Technologies:n KEPServerEx OPC UA -Palvelimen avulla. (KepWare, 2016.) KEPServerEx tarjosi oikean ohjausyksikköön luotavan yhteyden lisäksi mahdollisuuden toteuttaa virtuaalisen automaatiojärjestelmän Excel -ohjelmiston avulla. OPC UA -asiakasohjelman ja www-sovelluspalvelun yhteistoiminto taas toteutettiin käyttämällä Unified Automation GmbH:n tarjoaman .NET pohjaisen sovelluskehitintyökalun OPC UA -kirjastoa. (Unified Automation ,2016.) WWW-sovelluspalvelua varten luotiin Visual Studiolla .NET WebAPI –ratkaisu. Tälle WWW-sovelluspalvelulle määriteltiin yksi GET –reititys, jota kutsumalla haettiin Unified Automation GmbH:n OPC UA- kirjaston avulla OPC UA –palvelimen hakema muistiavaruus ja palautettiin se GET –pyynnön vastauksena.

Konstruktiivisen tutkimuksen tuloksena saatiin esitettyä ratkaisu tutkimuksen tutkimusongelmaan, eli että miten teollisuuden ohjausjärjestelmän ja pilvipalvelun välinen tiedonkeruurajapinta voidaan toteuttaa itse TCP/IP- ja HTTP – protokollaan tukeutuen ja muokkaamalla valmiiden ratkaisujen lähdekoodia. Tutkimuksen tuloksena syntyi järjestelmä, jonka avulla pystyttiin www-sovelluspalvelua käyttäen palauttamaan HTTP Get -pyynnön vastauksena hypoteettisen ohjausjärjestelmän koko muistiavaruus.

LÄHTEET

Ahmed, K. & Gregory, M. (2011). Integrating Wireless Sensor Networks with Cloud Computing. Proc. Of Seventh International Conference on eMobile Ad-hoc and Sensor Networks (MSN 2011). IEEE p. 364-366.

Armbrust, M., Fox, A., Griggith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. & Zaharia, M.(2010). A View of Cloud Computing. Communications of the ACM, 53, 50-58.

Badger, L., Grance, T. , Patt-Corner, R. & Voas, J. (2012). Cloud Computing Synopsis and Recommendations. Gaithersburg, USA: National Institute of Standards and Technology. Haettu 26.1.2017 osoitteesta https://www.nist.gov/node/588576

Byres, E. & Oulton, B. (2013). Industrial Networks Under Attack – Hackers and cyber security have become top-of-mind for executives tasked with protecting critical industrial systems. Security technology executive, January/February, 28-30. Haettu 20.11.2016 osoitteesta http://securitytechnologyexecutive.epubxp.com/i/107544-jan-feb-2013/3

Cavalier, S. (2012). Evaluating overheads introduced by OPC UA Specifications. Human - Computer Systems Interaction 2012, Part 1, 201–221.

Cucinotta, T. Mancina, A. Anastasi, G., Lipari, G., Mangeruca, L., Checcozzo, R. & Rusina, F. (2009). A Real-time service-oriented architecture for industrial automation. IEEE, 5, 267-277.

Cupec, R. Ziebinski, A. & Franek, M. (2013). FPGA Based opc ua embedded industrial data server implementation. Journal of Circuits, Systems, and Computers 2012, 22.

Fokert, K. Fojcik, M. & Cupek, R. (2011). Efficiency of OPC UA Communication in Java-Based Implementations. CN2011, CCIS 160, 348-357.

Girbeal, A., Nechifor, S., Sisak, F. & Perniul, L. (2010).Design and implementation of an OLE for process control unified architecture aggregating server for a group of flexible manufacturing systems. IET Software 2011, 5, 406-414.

Givenhchi, O., Trsek, H. & Jasperneite, J. (2013). Cloud Computing for Industrial Automation Systems - A Comprehensive Overview. IEEE 21st International Conference 9(2013).

Goldschmidt, T. & Mahnke, W. (2012). An Internal Domain-Specific Language for Constructing OPC UA Queries and Event Filters. ECMFA 2012, Springer LNCS 7349, 62–73.

Grossman, R. (2009). The Case for Cloud Computing. IEEE, 1520-9202, 23-27. IT Professional (Volume:11 , Issue: 2) pp: 23 - 27

Hassanien, A. E., Salem, A. M., Ramadan, R. & Kim, T. (2014). Advanced Machine Learning Technologies and Applications. Second International Conference 11 (2014). Cairo: AMLTA.

Hogan, B. P., Warren, C., Weber, M., Johnson, C. & Godin, A. (2012).Web Development Recipes. Dallas, Raleigh: Pragmatic Programmers, LLC.

Jansen, W. & Grance, T. (2011). Guidelines on Security and Privacy in Public Cloud Computing. Gaithersburg, USA: National Institute of Standards and Technology.

Jerhotova, E., Sikora, M. & Stluka, P. (2012). Dynamic Alarm Management in Next Generation Process Control Systems. APMS 2012, 398, 224-231.

Jestratjew, A. & Kwiecień, A. (2013). Performance of HTTP Protocol in Networked Control Systems. IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS 9(2013), 271-276.

Järvinen, P & Järvinen, A. (2004). Tutkimustyön metodeista. Tampere: Opinpajan kirja.

Karnouskos, S. (2011). Realising next-generation web service-driven industrial systems. Int J Adv Manuf Technol, 60, 409–419.

Karnouskos, S., Colombo, A.W., Bangemann, T., Manninen, K., Camp, R., Tilly, M., Stluka, P., Jammes, F., Delsing, J., & Eliasson, J.(2012). A SOA-based architecture for empowering future collaborative cloud-based industrial automation. IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, 25-28 Oct. 2012. 5766 - 5772.

KepWare (2016). KepServerEx. Haettu 28.3.2016 osoitteesta https://www.kepware.com/products/kepserverex/

Kozlowski, K.R. (2011). Robot Motion and Control 2011. London: Springer-Verlag.

Lager, M. (2005). What is SOA?. ProQuest Central, 11, 17-18.

Leidigh, K.(2000). Configuring TCP/IP Hosts, Methods, and Protocols. ProQuest Central, 13, 71-83.

Mell, P. & Grance, T. (2011). The NIST Definition of Cloud Computing (Draft). Gaithersburg: National Institute of Standards and Technology.

Nagorny, K. Colombo, W. & Schmidtmann , U. (2012). A service- and multi-agent-oriented manufacturing automation architecture An IEC 62264 level 2 compliant implementation.Computers in Industry, 63,813–823.

OPC Foundation (2008).Unified Architecture. Haettu 28.3.2016 osoitteesta https://opcfoundation.org/about/opc-technologies/opc-ua/

OPC Foundation (2009). OPC UA – Standardised communication acc. to IEC 62541. Haettu 26.1.2017 osoitteesta https://opcfoundation.org/wp-content/uploads/2014/05/OPC-UA_CollaborationOverview_EN.pdf

Raj, P., Venkatesh, V. & Amirtharajan R. (2013). Envisioning the Cloud-Induced Transformations in the Software Engineering Discipline. Springer-Verlag London 2013.

Rinaldi, J. (2016). OPC UA Unified Architecture The everyman's Guide to the Most Important Information Technology in Industrial Automation, USA, Lexington: John S. Rinaldi.

Shah, S. H., Khan, F. K., Ali, F. & Khan, J. (2013). A New Framework to Integrate Wireless Sensor Networks with Cloud Computing. IEEE International Conference on 3(2013).

Tan, V., Yoo, D. & Yi, M. (2009). Device Integration Approach to OPC UA -Based Process Automation Systems with FDT/DTM and EDDL. ICIC 2009, 5755, 1001-1012.

Unified Automation, (2016). Haettu 28.3.2016 osoitteesta https://www.unified-automation.com/

Uslar, M., Specht, M., Rohjans, S., Trefke, J. & Gonzalez, J. (2012). The Common Information Model CIM. POWSYS, 8, 179–186.

Vouk, M.A. (2008). Cloud Computing – Issues, Research and Implementations. Journal of Computing and Information Technology, 4, 235-246.

Zhang, Q., Cheng, L. & Boutaba, R.(2010). Cloud computing: state-of-the-art and research challenges. J Internet Serv Appl, 1, 7–18.