IOT-JÄRJESTELMÄ HARJOITUSKÄYTTÖÖN

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulun opinnäytetyö

Automaatiotekniikan koulutusohjelma

Valkeakoski, kevät 2016

Vili Kautto-Seinäjokinen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Valkeakoski
Automaatiotekniikan koulutusohjelma

Tekijä Vili Kautto-Seinäjokinen **Vuosi** 2016

Työn nimi IoT-järjestelmä harjoituskäyttöön

TIIVISTELMÄ

Tämä opinnäytetyö lähti Hämeen ammattikorkeakoulun Valkeakosken yksikön tarpeesta saada opetusmateriaalia IoT:tä eli asioiden internetiä varten. IoT on laaja käsite, josta on lukuisia eri määritelmiä. Niiden yhteisten tekijöiden tiivistäminen ja peruskäsitteen määrittely ovat tärkeää, mikäli aihetta halutaan opettaa.

Tässä työssä selvitetään mitä asioiden internet tarkoittaa sekä tarkastellaan kahta kattavampaa ja toisistaan eroavaa määritelmää. Työssä myös tarkastellaan IoT- ja M2M-markkinaekosysteemeitä.

Työ tiivistää osan asioiden internetistä saatavilla olevasta tietomäärästä mahdollisimman pieneen kokoon avainasioiden tajuamisen helpottamiseksi.

Tämä työ antaa pohjatietoa ja suunnitellun järjestelmän aiheen opettamiseksi, mutta alaa pitää seurata aktiivisesti ja opetusta kehittää muutoksien mukaan.

Työssä käydään läpi vaiheet, joilla koulun henkilökunta voi itse rakentaa ja ottaa käyttöön IoT-järjestelmän. Järjestelmän osat on valittu tulevaisuuden vaatimuksia silmällä pitäen ja niitä voidaan käyttää monimutkaisempiin toteutuksiin. Opinnäytetyössä myös selvitetään suunnitellun harjoitusjärjestelmän ohjelmapohjaisten osien toimintaa.

Avainsanat tietoverkot, mikro-ohjaimet, esineiden internet, ohjelmointi, määritelmät

Sivut 38 s. + liitteet 9 s.

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Valkeakoski
Degree Programme in Automation Engineering

Author Vili Kautto-Seinäjokinen **Year** 2016

Subject of Bachelor's thesis IoT system for practical excercises

ABSTRACT

This Bachelor's thesis began from the need of HAMK University of Applied Sciences' need for study material on the internet of things. IoT is a broad subject which has multiple different definitions. Finding the similarities and creating a base understanding based on them is crucial for teaching the subject.

This thesis aims to clarify what the internet of things means and focuses on two of the more comprehensive definitions which also differ from each other. The market ecosystems are also examined briefly.

The thesis compresses the vast amount of information on IoT as much as possible to make grasping key points easier.

This Bachelor's thesis gives a basic knowledge base and a design for an IoT system for practical exercises but the teachers need to keep themselves updated on the industry and change with it to keep their teachings relevant.

This thesis gives step by step instructions on how the school staff can build their own system and start using it. The parts of the system were chosen specifically to scale with demand and to allow more complex implementations to facilitate future needs.

The IoT system designed for practical exercises has its software components examined and the relevant basic functions explained.

Keywords networks, microcontrollers, internet of things, definitions

Pages 38 p. + appendices 9 p.

TERMIT JA LYHENTEET

Arduino	Halpojen mikrokontrollerien tuoteperhe
ESP8266	WiFi-lähetin/vastaanotin ohjelmoitavalla mikroprosessorilla
ETNO	European Telecommunications Network Operators' Association
ETSI	European Telecommunications Standards Institute
GNU	GNU General Public License
IoT	Internet Of Things
ITU	International Telecommunications Union
M2M	Machine-To-Machine eli koneiden välinen kommunikaatio.
MaCaco	Protokolla muistin ja toimintojen käyttämiseen mikrokontrolleriympäristössä
nRF24L01	Langaton lähetin/vastaanotin
Objekti	M2M-verkon laite tai ohjelmallinen osa
openHAB	Automaatiopalvelinohjelmisto
Raspberry Pi	Pieni yhden piirilevyn tietokone
Souliss	Arduino yhteensopiva älytalo-ohjelmisto
Thing	Fyysinen tai virtuaalinen asia, joka pystytään tunnistamaan ja liittämään IoT-verkkoon
vNet	Vertaisverkkoprotokolla

SISÄLLYS

1 JOHDANTO

Asioiden internet on vielä kehitysvaiheessa ja teollisuudenala on vahvasti pirstaloitunut eikä ole yhtä tai edes muutamaa yleisesti hyväksyttyä ja käytettyä määritelmää. Eri määritelmissä on kuitenkin paljon yhteneväisyyksiä ja tämä työ yrittää esitellä ne helposti ymmärrettävässä muodossa. Sähköautomaatioon erikoistunut insinööri on hyvässä asemassa työllistyä alalle, koska elektroniikka, perinteinen sähköohjaus, tietoliikenneväylät ja ohjelmointi ovat IoT-ratkaisujen keskiössä.

Tämä opinnäytetyö lähti Hämeen ammattikorkeakoulun Valkeakosken yksikön tarpeesta saada opetusmateriaalia IoT:tä eli asioiden internetiä varten. IoT on laaja käsite, josta on lukuisia eri määritelmiä. Niiden yhteisten tekijöiden tiivistäminen ja peruskäsitteen määrittely ovat tärkeää, mikäli aihetta halutaan opettaa.

Tässä työssä selvitetään mitä asioiden internet tarkoittaa sekä tarkastellaan kahta kattavampaa ja toisistaan eroavaa määritelmää. Työssä myös sivutaan IoT- ja M2M-markkinaekosysteemeitä. Lopputavoitteena on antaa käsitys siitä mitä IoT:llä tarkoitetaan sekä mitä se pitää sisällään, ja valmis suunnitelma pienelle järjestelmälle, jota voidaan käyttää opetuksessa.

Teoriapohjaa varten tutkittiin määritelmiin liittyviä dokumentteja ja internetissä olevia artikkeleita asioiden internetistä sekä järjestelmän osien dokumentaatioita ja Soulissin lähdekoodia.

Työ tiivistää asioiden internetistä olevan tietomäärän mahdollisimman pieneen kokoon avainasioiden tajuamisen helpottamiseksi. Opinnäytetyössä myös selvitetään suunnitellun harjoitusjärjestelmän ohjelmapohjaisten osien toimintaa.

Käytännön osaa varten testattiin Soulissin ja openHABin toimintaa. Saadun kokemuksen pohjalta valittiin järjestelmään elektroniset osat ja kirjoitettiin esimerkkiohjelma.

Tämä työ antaa pohjatietoa ja valmiiksi suunnitellun järjestelmän aiheen opettamiseksi, mutta alaa pitää seurata aktiivisesti ja opetusta kehittää muutoksien mukaan.

Järjestelmän osat on valittu tulevaisuuden vaatimuksia silmällä pitäen ja niitä voidaan käyttää monimutkaisempiin toteutuksiin. Osien avoimuuden vuoksi opiskelijat voivat myös helposti syventää osaamistaan ja kehittää itsenäisesti uusia kokoonpanoja sekä käyttötarkoituksia.

2 INTERNET OF THINGS

2.1 Yleistä

Internet Of Things eli asioiden internet ei ole vielä saanut lopullista määritelmää eikä standardia miltään taholta ja määritelmät itsessään eroavat laajasti. Tämä vaikeuttaa huomattavasti mahdollisten ongelmien havaitsemista ja alan toimijoiden yhteistyötä. (Internet of Things (IoT) Ecosystem Study 2014, 3.)

Suomessa esimerkiksi Elisa yhdistää IoT- ja M2M-termit saman teollinen internet -nimikkeen alle ja käyttää ensimmäistä kuvaamaan toimilaitteita ja niiden muodostamaa verkkoa, ja jälkimmäistä kuvaamaan laitteiden välisen keskustelun toteutusta. (Elisa IoT n.d.).

Monet standardointiorganisaatiot kuitenkin tekevät tutkimusta jatkuvasti ja ovat julkaisseet rakennekuvauksia sekä versioita omista määritelmistään. Pääasiallinen ero määritelmien välillä on, kuinka tarkasti vaatimukset ovat laadittu ja menevätkö ne verkkoja ja laitteita syvemmälle aiheeseen kuten ennusteisiin IoT:n luoman markkinan jaosta erilaisten toimijoiden välillä. Tästä johtuen vaatimukset joihin tämän opinnäytetyön teksti perustuu voivat muuttua tulevaisuudessa.

Asioiden internetille räätälöityjä yleisiä laite-, kommunikaatio- ja turvallisuusstandardeja ei ole toistaiseksi lähdetty tarkasti määrittelemään vaan eri organisaatiot hyödyntävät jo valmiina olevia omia sekä muiden luomia yleisesti hyväksyttyjä tekniikan standardeja.

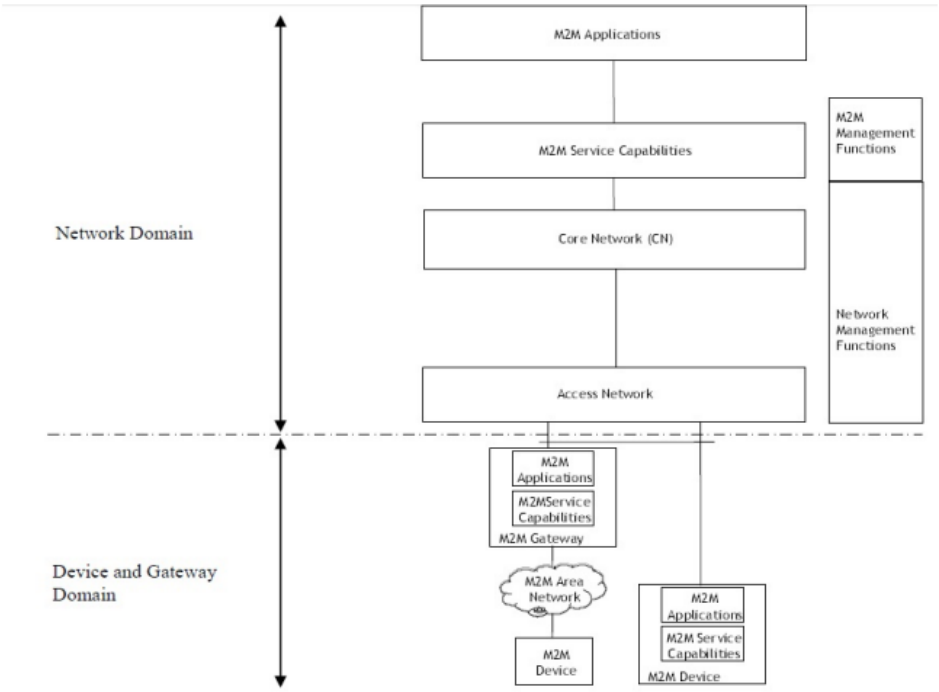
Järjestöt ottavat siis kantaa konseptitasolla asettamalla kriteerit sekä käytettävät olemassa olevat standardit, jotka pitää täyttää, jotta voidaan puhua asioiden internetistä. Järjestelmien, laitteiden sekä palvelujen tekninen toteutus jäävät niitä toimittavien ja myyvien organisaatioiden ja yksilöiden vastuulle.

Yksityiset yritykset valmistavat omia ratkaisujaan, jotka perustuvat yleensä suurelta osin yrityksen omaan suljettuun teknologiaan, joten eri valmistajien laitteiden ja ohjelmien yhteensopivuus on heikko ja vaatii erillisen ratkaisun toimivuuden saavuttamiseksi.

2.2 ETSI:n määritelmä

ETSI on eurooppalainen riippumaton ja voittoa tavoittelematon tieto- ja viestintäteknologioiden standardoimisjärjestö, jolla on jo monta julkaistua standardia ja määritystä M2M-termin alla. ETSI käyttää termiä Machine-To-Machine kuvaamaan asioiden internetiä muistuttavaa konseptia.

M2M-määritelmä on ollut käytössä jo vuosia, mutta ei ole palvelun toiminnan kannalta yhtä avoin kuin IoT vaan se skaalautuu laajaverkkoihin asti palveluntuottajien hallitsemana ja muistuttaa enemmän teollista automaatioverkkoa ja siihen liittyvää tuotannonohjausta. Tämä johtuu M2M-ratkaisujen pääasiallisen käyttäjäkunnan muodostuvan yrityksistä. Määritelmässä itsessään ei rajata loppukäyttäjiä vain yrityksiin.

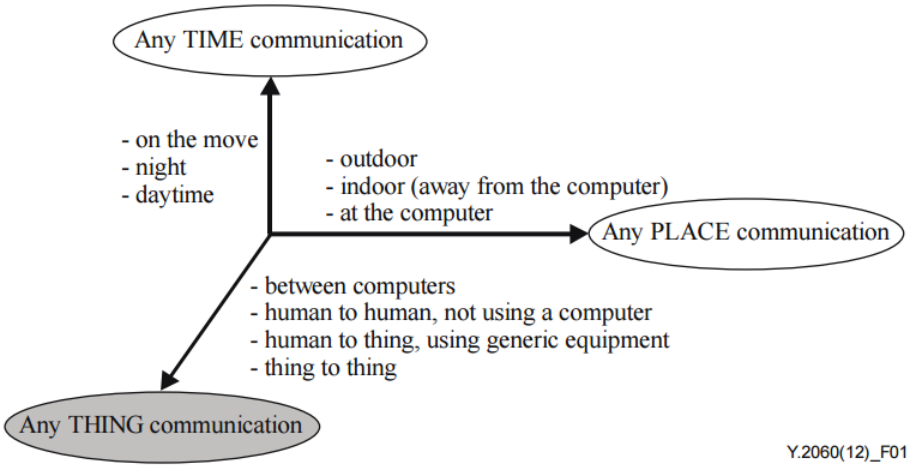


Kuva 1. M2M-verkon rakennekuvaus (M2M service requirements 2010).

Kuvan 1 mukaisesti M2M-verkko muodostuu laiteverkosta, yhdysverkosta sekä keskusverkosta. Laiteverkon muodostaa yhdyskäytävä, jonka kautta laiteverkko voi keskustella ulkomaailman kanssa. Yhdysverkko yhdistää laitteet ja yhdyskäytävät keskusverkkoon, johon voidaan liittää asiakasrajapintoja sekä palvelimia. Palvelu tuotetaan keskus- ja yhdysverkkoon kytkettyjen laitteisiin asennetuilla M2M-ohjelmilla, jotka muodostavat loppukäyttäjän näkemiä objekteja.

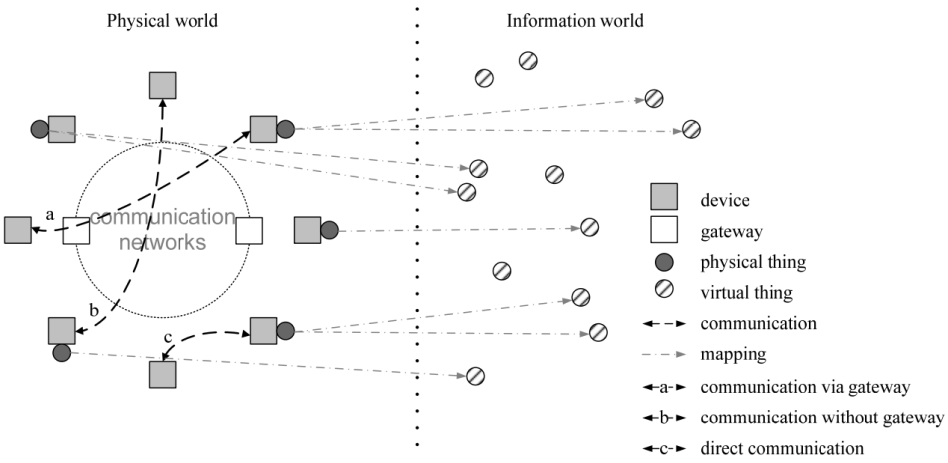
2.3 ITU:n määritelmä

YK:n alainen tieto- ja viestintäteknologian organisaatio ITU määrittelee asioiden internetin globaalina infrastruktuurina, joka mahdollistaa kehittyneet palvelut yhdistelemällä ”asioita” (thing) perustuen olemassa oleviin sekä kehittyviin tieto- sekä viestintäteknologioihin. (Kuva 1.)



Kuva 2. Asioiden internetin lisäämä ajattelutapa (Overview of the Internet of things 2012, 3).

Asioiden konseptin teknologinen avoimuus (Kuva 2.) helpottaa isojen verkkojen järjestelemisen pienempiin järjestelmiin ja huomio kiinnittyy suunnittelussa siitä, miten jokin tehdään siihen mitä halutaan tehdä. Fyysinen toteutus voidaan suunnitella lopuksi ja hyödyntää valmiiksi olemassa olevia laitteita ja tekniikoita.



Kuva 3. Asiat IoT-verkossa (Overview of the Internet of things 2012, 3).

ITU:n mallissa loppukäyttäjä ei välttämättä tule näkemään kuin digitaalisen esityksen verkossa olevista ohjelmista ja laitteista eikä voi erottaa mikä järjestelmässä on digitaalista ja mikä fyysistä.

Asioiden internetin perusominaisuudet

- Asia asioiden internetissä voi olla fyysistä, digitaalista tai molempia (Kuva 3.).
- Mitä tahansa voidaan liittää maailmanlaajuiseen kommunikaatioverkostoon ja se tunnistetaan yksilöllisen tunnuksen avulla osaksi IoT-verkkoa.
- Asioiden internet pystyy tarjoamaan asiakohtaisia palveluita asioiden omien rajoitusten puitteissa, kuten yksityisyydenturvan ja fyysisen laitteen sekä sitä vastaavan virtuaalisen asian vastaavuuden.
- Erilaiset laitteet erilaisista verkoista voivat kommunikoida keskenään erilaisia väyliä pitkin (Kuva 3.).
- Laitteiden tiloja ja tietoja päivitetään dynaamisesti ja aktiivisena olevien laitteiden määrä voi vaihdella rajusti.
- Hallinnoitavien sekä toistensa kanssa keskusteltavien laitteiden määrä moninkertaistuu verrattuna tällä hetkellä internetiin yhteydessä olevaan määrään.
- Laitteiden aiheuttaman viestinnän suhde ihmisten aiheuttamaan viestintään kasvaa huomattavasti.
- Hallittavan ja käytettävissä olevan tiedon määrä kasvaa verrattuna nykyiseen. (Overview of the Internet of things 2012, 5.)

2.4 Markkinat

2.4.1 IoT-ekosysteemi

Markkinat asioiden internetille ovat kasvussa, mutta pirstaloituneet. Toimijat tekevät tuotteita löytämilleen markkinoille ja usein käyttävät omia suljettuja teknologioitansa, joista voi kehittyä yleisesti hyväksyttyjä standardeja.

IoT:n toimijat voidaan jakaa viiteen eri ryhmään:

- fyysisessä maailmassa toimivat yksityiset
- digitaalisessa maailmassa toimivat yksityiset
- akateeminen maailma
- julkinen sektori ja hallinto
- muut.

Fyysisen maailman toimitsijat tekevät, myyvät ja toimittavat laitteita, kuten sensoreita.

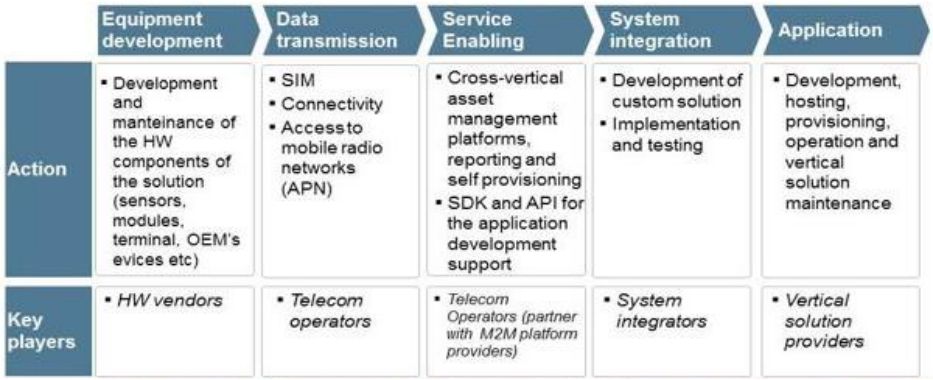
Digitaaliset toimijat tuottavat palveluita ja ohjelmia, joilla toimilaitteita voi hallita. Esimerkiksi pilvipalvelu, jossa pyörii automaatiologiikka, johon yksittäiset laitteet ja asiakasohjelmat ovat yhteydessä.

Akateeminen maailma käsittää käytännössä vain tutkimuksen ja kehityksen asioiden internetin parantamiseksi sekä tulevaisuuden osaajien kouluttamisen.

Julkinen sektori ostaa tai kehittää itse uusia ratkaisuja valvoa sekä hallita kaupunkeja ja infrastruktuuria hajautetusti ja automatisoidusti tehokkuuden parantamiseksi.

Muihin toimitsijoihin luetaan yksityiset henkilöt ja pienet yritykset, jotka esimerkiksi joukkorahoituksen avulla luovat markkinoita mullistavia ideoita. Myös itse asiakkaat ja heidän käyttäytymisensä sekä säännöksiä ja standardeja hallinnoivat organisaatiot ovat elintärkeitä IoT:n jatkuvalle kasvulle. (Internet of Things (IoT) Ecosystem Study 2014, 4-5.)

2.4.2 M2M-ekosysteemi



Kuva 4. M2M-ekosysteemi (ETNO views on M2M 2014).

Machine to Machine on ollut käytännön ratkaisuissa käytössä pidempään kuin nykyinen IoT-konsepti ja sillä on jo vakiintunut yrityslähtöinen ekosysteemi (Kuva 4.).

Se on myös tiukemmin toimitsijoiden hallitsema sekä vähemmän pirstaloitunut M2M-määritelmän kypsyyden takia ja käyttää pääasiassa yritysmarkkinoille suunnattuja laitteita, ohjelmistoja ja palveluita.

Teleoperaattorit ovat tärkeä osa ekosysteemiä, sillä langattomia paikannusvälineitä ja sensoreita hyödynnetään paljon ja niiden tiedonsiirto toimii usein puhelinverkon yli. Suomessa esimerkiksi Sonera ja Elisa myyvät M2M-tuotteita sekä ratkaisuja.

2.5 Käytännön haasteita

Asioiden internet on vielä kehittymässä, joten ns. kipukohtia löytyy. Suurin ongelma on määritelmien, markkinoiden ja toimijoiden pirstaloituneisuus, mutta on muitakin tärkeitä aiheita, jotka kaipaavat huomiota. Nämä voidaan jakaa standardien, teknologian ja yritysten osa-alueisiin. Samat haasteet voivat koskettaa useampaa kuin yhtä osa-aluetta. (Internet of Things (IoT) Ecosystem Study 2014, 4-22.)

2.5.1 Maailmanlaajuinen yhteistyö ja kattavuus

Kaikilla standardoimisjärjestöillä ei ole maailmanlaajuista toimintaa, joten eri järjestöjen tulisi tehdä yhteistyötä, jotta voidaan saavuttaa maailmanlaajuisesti käytössä olevat määritelmät ja standardit.

2.5.2 Yhteinen arkkitehtuuri ja referenssimallit

Ei riitä, että kaikki lukevat samoja standardeja, vaan tarvitaan myös yhteinen arkkitehtuurinen kehys ja referenssimallit siitä, miten rakentaa IoT-verkkoja ja laitteita.

2.5.3 Suojaus, turvallisuus, kyberturvallisuus, yksityisyydensuoja

Turvallisuusstandardit tullaan luomaan erikseen muista määrityksistä, mutta käyttäjien suojaaminen ja turvallisuuden takaaminen ovat erittäin tärkeitä ottaa jatkuvasti huomioon standardeja ja käytettäviä tekniikoita määriteltäessä.

Asioiden internet kasvattaa turvallisuushaasteet uusiin mittasuhteisiin perinteiseen internetiin verrattuna. Laitteiden määrän kasvaessa kasvaa myös ns. pinta-ala mihin hyökkäyksiä voi kohdistaa. Yksittäinen saastutettu laite saattaa taata jopa koko laiteverkon saastumisen.

Yhdysvaltain Federal Trade Commission järjesti asioiden internetiin turvallisuuteen keskittyvän työpajan 19. marraskuuta 2013, joka määritti IoT:lle kolme kuluttajia koskettavaa riskiryhmää.

“According to panelists, IoT devices may present a variety of potential security risks that could be exploited to harm consumers by: (1) enabling unauthorized access and misuse of personal information; (2) facilitating attacks on other systems; and (3) creating safety risks.” (Internet Of Things: Privacy & Security in a Connected World 2013, 10.)

Suurin osa IoT-laitteista tulee keskittymään tiedonkeruuseen tai asioiden ohjaamiseen. Saastunut verkko saattaa siis olla käyttäjälle ja tämän omaisuudelle fyysisesti vaarallinen tiedonkeruun lisäksi.

Yksi tapa turvallisen IoT-palvelun saavuttamiseksi voikin siis olla verkkoprotokollien ja laitteiden tasolla olevien turvallisuusratkaisujen kehittäminen ja käyttäminen. Tämä tapa keskittyy pienentämään pinta-alaa, jolle hyökkäykset voi suunnata vähentämällä suoraan ulkoverkkoon yhteydessä olevien laitteiden määrää esimerkiksi VPN-yhteyksiä hyödyntämällä.

Kerätyn datan minimointi ja sen säilyttämisajan lyhentäminen voivat myös nostaa kynnystä rikollisille toimille, jos haluttu data on jo poistettu, kun rikollinen pääsee tunkeutumaan verkkoon, eikä vähästä datasta voi aina rakentaa kohteesta tarkkoja malleja. Yritysten nykyiset toimintatavat ovat kuitenkin juuri päinvastaisia.

Tärkeäksi nousee myös, ettei loppukäyttäjää päästetä suoraan käsiksi jokaiseen laitteeseen, vaan ohjaus tapahtuu yhdyskäytävän kautta eikä anneta mahdollisuutta muokata laitteen toimintaa ilman varmennusta, tai ottaa yhteyttä laiteverkon ulkopuolelta.

Muita haasteita ovat IoT-alalle lähteneiden yritysten mahdollinen osaamisen puute turvallisuusasioissa, yksinkertaisempien laitteiden päivittämisen vaikeus sekä kuluttajien tieto päivityksistä ja osaaminen sekä halu päivittää oma järjestelmä. Jotkut yritykset saattavat myös jättää päivitystuen pois suunnitelmistaan rahan säästämiseksi, jolloin kuluttajalle jää turvaton laite oston jälkeen.

2.5.4 Skaalausmahdollisuudet

Vanhat järjestelmät ja tavat tehdä asioita eivät yksinkertaisesti pysy mukana asioiden internetin kasvussa. Yksi isoimmista ongelmista on oman tunnisteen antaminen jokaiselle laitteelle.

Tällä hetkellä käytössä on 48 bittinen MAC-osoiteavaruus, jonka oli tarkoitus kestää 100 vuotta, mutta kun osoitteita aletaan jakaa kaikelle valaisimista lähtien ei MAC-osoitteita tule riittämään niin pitkään.

Ongelmaksi muodostuu myös isojen laitemäärien verkottaminen ja vertaisviestintä.

On siis kehitettävä laiteratkaisuja ja uusia ohjelmistoja, jotka pystyvät erittäin suurien laitemäärien hallintaan sekä yleistää uusi laiteosoitestandardi, joka on tarpeeksi suuri tulevaisuuden laitemäärille.

2.5.5 Ohjelmistostandardit

Tällä hetkellä suurin osa standardeista ottaa huomioon vain osia IoT:n vaatimasta ohjelmistokehyksestä. Tarvitaan ohjelmistostandardeja, jotka mahdollistavat yhteensopivuuden eri tuotteiden välillä ohjelmistopuolella.

2.5.6 Yhteensopivuus

Yhteensopivuus on vakava ongelma lukuisten suljettujen ratkaisujen aiheuttamissa suljetuissa puutarhoissa.

Tarvitaan yhteensopivuuden takaavia standardeja eri verkkotekniikoiden, markkinasegmenttien palveluiden ja viestintäprotokollien välillä.

2.5.7 Käytettävyys

Asioiden internetin helppokäyttöisyys on tärkeää, kun halutaan saavuttaa laaja käyttäjäkunta ja saada suurimmasta osasta maksavia asiakkaita. Tällä hetkellä suurin osa IoT-ratkaisuista on tarkoitettu yrityskäyttöön tai toimii tee-se-itse -periaatteella ja ne ovat monimutkaisia.

Järjestelmien tulisi olla viimeistään asennuksen jälkeen helppokäyttöisiä ja turvallisia kuluttajalle, jolla ei ole kokemusta tai koulutusta järjestelmästä.

2.5.8 Koulutus

Sekä käyttäjät, että tuottajat tulevat tarvitsemaan koulutusta asioiden internetistä, mihin se pystyy, mitä se tekee, miten se toimii sekä sen turvallisuudesta.

Tämä tulee olemaan erityisen hankalaa kuluttajien kohdalla, koska he haluavat mahdollisimman yksinkertaisia tuotteita eivätkä reagoi positiivisesti jyrkkiin oppimiskäyriin.

Myös tulevaisuuden ammattilaisia täytyy kouluttaa, eikä asioiden internetille ole vielä yleisesti omia koulutuslinjojaan. Vanhat ammattilaiset kaipaavat myös koulutusta, jotta he eivät yrittäisi tehdä ratkaisua osa-alueelle, josta eivät ymmärrä. Tämä on erityisen tärkeää turvallisuusratkaisujen toteuttamisessa.

3 PÄÄTELMÄT

3.1.1 Mitä on asioiden internet?

IoT:lle löytyy monia määritelmiä, jotka pääpiirteittäin noudattavat samoja ajatusmalleja. Pääasiallinen ero löytyy toteutuksesta, markkinaekosysteemistä ja mitä sen pitäisi minimissään tarjota käyttäjälle.

Tiivistetysti yksittäinen IoT-ratkaisu koostuu vähintään yhdestä vertaisverkosta, johon on kytketty elektronisia laitteita ja sensoreita, joka on vähintään yhdellä tapaa yhteydessä internetiin ja kaikkia sen asioita pystytään hallitsemaan mistä tahansa ja milloin tahansa verkkoyhteyden avulla. Asioiden vertaisverkko voi myös vaihtaa ja analysoida kerättyä tietoa sekä linkittää toimintoja toimilaitteiden välillä eikä välttämättä tarvitse aktiivista ihmiskäyttäjää missään vaiheessa käyttöönoton jälkeen. Esimerkiksi perinteinen kiinteistöautomaatio eroaa siinä, että ohjaus on yleensä keskitettyä yhdelle alueelle eivätkä laitteet prosessoi tietoa itse vaan sen tekee keskuslogiikka.

IoT voi skaalautua isoihin monia verkkoja ja laitteita käsittäviin kokonaisuuksiin. Isot palvelut tulevat olemaan pääasiassa internet-palveluntarjoajien tai isojen ICT- ja automaatioyritysten tuottamia. Pienempien toimijoiden tarjonta tulee rajoittumaan pieniin ratkaisuihin yksityishenkilöiden ja pienyritysten käyttöön tai tuottamaan isomman palvelun osia, kuten laitteita ja ohjelmistoja.

Asioiden internetin perusperiaatteet perustuen useamman järjestön määritelmiin

- Yhteys internetiin ja mahdollisuus käyttää omaa järjestelmää rajoitteitta.
- ”Milloin vain, missä vain ja mitä vain” -ohjaus.
- Laitteiden oma vertaisverkko ja keskenään keskustelu mahdollisia.
- Laitteiden välinen keskustelu ja toimintojen suorittaminen vertaisverkossa.
- Toimintojen ei kuitenkaan kannata olla täysin riippuvaisia toisistaan.
- Yhdistys ulkoverkkoon yhdyskäytävän kautta tai suoraan laitteelta.
- Käyttäjälle näkyvä asia tai toiminto ei välttämättä ole pelkkä fyysinen laite vaan yhdistelmä eri laitteita, ohjelmia ja niiden toimintoja.
- Yhden asian takana voi olla useampi laite ja niillä pyörivät ohjelmistot sekä niistä riippuvaiset toimilaitteet (valaisimet, releet, anturit yms.).
- Laitteisto- ja väyläriippumattomuus.
- Yhden valmistajan ratkaisu harvoin kattaa kaikkia tilanteita eikä yksi kommunikaatiotekniikka aina riitä.
- Skaalaus suurelle laitemäärälle.
- Jokaisessa kodinkoneessa ja muussa toimilaitteessa mahdollisuus IoT-versiolle.
- Laitteet hoitavat suurimman osan työstä perustuen ihmisen rutiineihin ja tarpeisiin.
- Turvattu mahdollisimman hyvin luvatonta käyttöä vastaan.

3.1.2 Sähköautomaatioinsinöörin rooli asioiden internetissä

Vaikka eri määritelmät ja standardit tulevat suurella todennäköisyydellä sulautumaan yhteen, tällä hetkellä kaksi vallitsevaa konseptia; Machine to Machine ja Internet of Things ovat markkinoiltaan muotoutuneet erilaisiksi, joten automaatioon erikoistuneen insinöörin rooli on kummassakin hieman erilainen.

Sähköautomaatiokoulutuksen saanut insinööri sopii parhaiten neljänteen kategoriaan M2M-ekosysteemissä (Kuva 4.), sillä se vastaa koulutuksen toimenkuvaa, mutta insinööri voi myös suuntautua palvelu- tai laitetoimitukseen. Monet yritykset tulevat luultavasti tekemään molempia, sillä ohjelmistot pyörivät pääasiassa laitteiden elektroniikan päällä tai niihin yhteydessä.

IoT:n määrittelemättömyydestä johtuva avoimuus helpottaa koko palvelun tuottamisen yhden pienemmän toimitsijan kautta hyödyntäen asiakkaan resursseja ja avoimia standardeja, sekä omia tai kolmansien osapuolien pilvi- ja VPN-palveluja. Tässä tapauksessa insinööri voi alkaa yrittäjäksi tai tehdä ja ottaa käyttöön palveluratkaisun osia toisen yrityksen työntekijänä. Tällöin kuvan neljä mukainen ekosysteemi kutistuu laitetoimittajiin ja palvelutoimittajiin. IoT-toimitsijat voivat myös halutessaan noudattaa M2M-mallia kasvattaessaan palveluiden kokoa ja monipuolisuutta.

4 LAITTEIDEN VERTAISVERKKO

4.1 Souliss yleisesti

Souliss on Arduino-laitteille tehty älytalo-ohjelmisto, jolla pystyy hoitamaan tavallisimmat IoT:n vaatimat toimenpiteet kuten mittareiden lukemisen, laitteiden automaattisen ohjaamisen sekä hälytykset ja hajautetun ohjauksen.

Souliss vaatii IP-verkon järjestelmän ohjaamiseen, koska käyttäjärajapinnan ohjelmat tukevat vain Ethernet/IP-yhteyttä, mutta voi myös kommunikoida vertaisverkon osien kanssa itsenäisesti useiden eri tekniikoiden välityksellä.

Tähän opinnäytetyöhön valitsin Soulissin, koska se on sopiva pohja aloittelijatason harjoituksiin ja täyttää useimmat IoT:n asettamat vaatimukset. Se on helppo ottaa käyttöön ja sille on yksinkertaista tehdä pieniä rutiineja ja etähallinta, jotka käyttävät asioiden vertaisverkon periaatetta. Ohjelmisto on vapaa muokattavaksi, jaettavaksi ja käytettäväksi avoimella GPL-lisenssillä, joten oppilas voi myös käyttää järjestelmää syventämään osaamistaan.

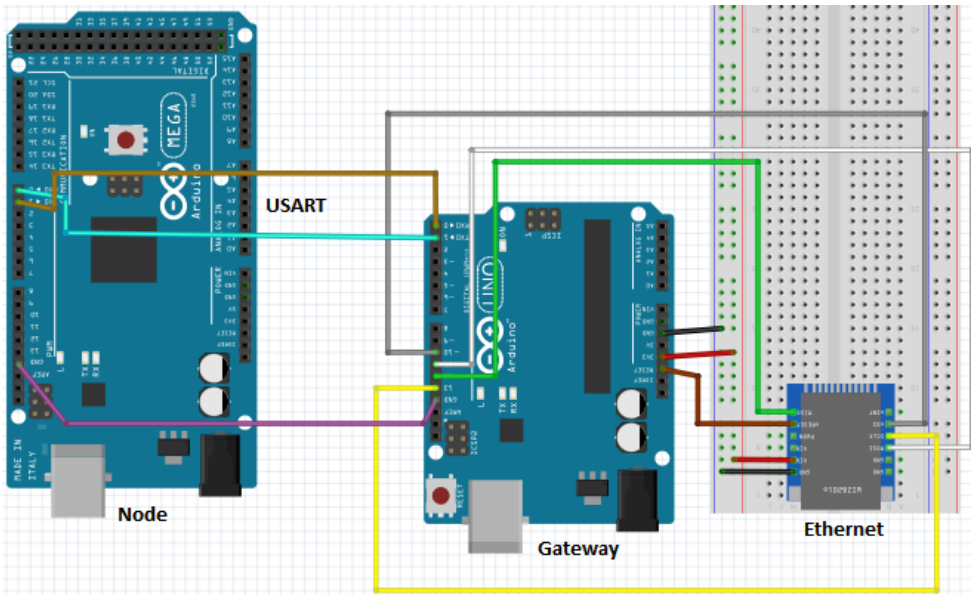
Muita tarkasteltuja vaihtoehtoja olivat http:n yli tehtävät suorat käskyt, REST-rajapinnalla ja CoAP (Constrained Application Protocol). Nämä vaihtoehdot pystyvät siirtämään tietoa ainoastaan IP-verkon yli. Lisäksi listatut tavat tarvitsevat syvemmän perehtymisen kuin Souliss, jossa on oma rajapinta valmiina ja aloittelijalle helpot funktiot. Kun otetaan huomioon, että harjoitusten on tarkoitus opettaa IoT:n idea eikä omien ohjelmien luomista yksittäisten protokollien päälle, niin Souliss on paras vaihtoehto.

Soulissin suurin heikkous on verkkotason turvallisuusominaisuuksien puuttuminen, joten ulkoisia hyökkäyksiä vastaan pitää varautua muulla tavalla, kuten eristämällä se omaan lähiverkkoonsa ja kierrättämällä kaikki liikenne esimerkiksi openHABin kautta. Soulississa on myös mahdollisuus lukita käyttöliittymä yhteen asiakkaaseen, mutta ominaisuuden toimivuutta hyökkäyksiä vastaan ei ole testattu kattavasti. Aiemmin mainituista vaihtoehdoista ainoastaan CoAP tarjoaa suojauksen yhteyksille. Kehittäjien mukaan se vastaa 3072bit RSA -avaimiin perustuvaa suojausta.

4.2 Verkko ja vNet

4.2.1 Verkon rakenne

Souliss-vertaisverkko toteutetaan vNet-protokollalla, joka on suunniteltu mahdollistamaan laiteverkon toteuttamisen usean eri verkkotekniikan yli. Jokainen laite verkossa saa oman vNet-osoitteen ja sille määritetään aliverkon peite. Lisäksi määritetään supersolu, jos tiedonsiirtoon käytetty tekniikka vaihtuu tai langattoman verkon kantomatka ei riitä ja kantoaluetta jatketaan solulla.



Kuva 5. Esimerkki pienestä Souliss-järjestelmästä

Verkon yksittäinen osa voi olla solu, supersolu tai yhdyskäytävä. Solu on yksittäinen laite, joka voi vastaanottaa ja lähettää tietoa. Supersolu voi myös reitittää eri verkkotekniikkaa käyttäville soluille muualta tulevia paketteja tai jatkaa saman langattoman tekniikan verkon kantomatkaa. Yhdyskäytävä on verkon rajapinta ulkomaailmaan ja reitityksen lisäksi sen kautta pystytään keskittämään laitteilta saatua tietoa käyttäjälle sekä tallentamaan sitä muistiin.

Kuvassa 5 on esimerkki järjestelmästä, jossa verkkoon päästään käsiksi Ethernet-yhteydellä, mutta mikrokontrollerien välinen kommunikointi tapahtuu USART-sarjayhteydellä. Yhdyskäytävä reitittää paketit solun ja lähiverkon välillä sekä keskustelee itsenäisesti sekä solun, että lähiverkon kanssa.

4.2.2 Verkkotyypit ja vNet-protokolla

Taulukko 1. Tuetut kommunikointiväylät ja niiden osoitealueet (Souliss Wiki n.d.).

Tekniikka	Osoitealueen alku	Osoitealueen loppu
M1 – UDP/IP	0x0000	0x00FF
M1 – UDP/IP Käyttäjätila	0x0100	0x64FF
M2 - 2.4 GHz IEEE 802.15.4	0x6500	0xAAFF
M3 – UDP/IP Broadcast	0xAB00	0xBBFF
M4 – Ei Käytössä	0xBC00	0xCCFF
M5 – USART	0xCE00	0xDFFF

Virallinen Souliss-julkaisu tukee UDP/IP, 2.4 GHz IEEE 802.15.4 ja USART-tekniikoita sekä niihin perustuvia ratkaisuja kuten nRF24L01. Jokaisella tavalla viestiä on oma osoitealueensa (Taulukko 1.), joten erillisiä

asetuksia ei tarvitse tehdä ohjelmaa kirjoittaessa käytettävän väylätekniikan tarvitseman ajurin tuonnin lisäksi.

Soulissin avoimuus takaa sen, että vaihtoehtoisia väylätekniikoita ja laitealustoja voidaan hyödyntää viestintään ja laitteisiin käytettäviä kirjastoja muokkaamalla tai tekemällä uusia kirjastoja.

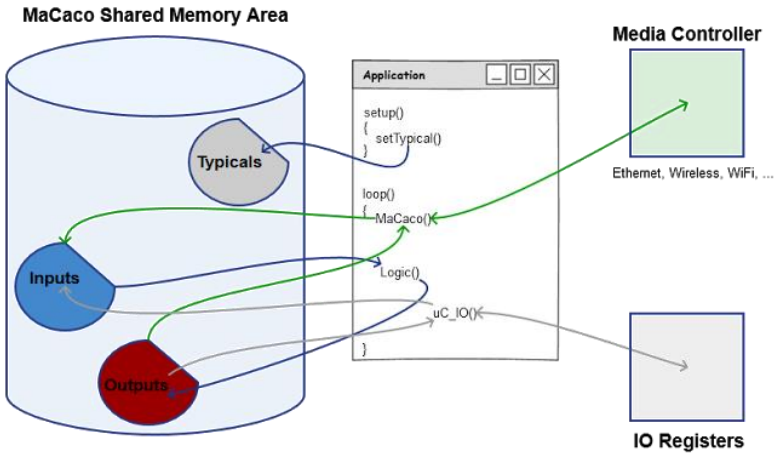
Taulukko 2. vNet-kehyksen rakenne (Souliss Wiki n.d.).

Pituus (1 tavu)	Portti (1 tavu)	Vastaanottaja (2 tavua)	Lähettäjä (2 tavua)	Tieto (6 tavua)

Vertaisverkko pakkaa käskyt omiin kehyksiinsä (Taulukko 2.), joista 6 tavua on otsikkotietoa ja sisältää pituuden, portin, lähettäjän ja vastaanottajan. Loput tavut varataan käyttötiedolle, mutta tietopaketin koko voi vaihdella ja kehyksen lopullinen koko riippuu toteutuksesta. Valmis kehys pakataan tietona käytettävän väylätekniikan kehyksen sisälle ja siirretään eteenpäin.

4.3 Soulissin tietorakenne ja MaCaco-protokolla

4.3.1 Tietorakenne



Kuva 6. Tietorakenne ja sen toiminnan kuvaus (Souliss Wiki n.d.).

Kuvan kuusi mukaisesti muistialue on jaettu kolmeen osaan ja jokainen alue on jaettu yksittäisiin paikkoihin (Slot), jotka muodostavat toimintojen muistipaikat. Toiminto (Typical) ja sen ominaisuudet määritetään yhteen tai useampaan muistipaikkaan ohjelman konfiguraatiovaiheessa (setup()). Ajon aikana toiminnon logiikka prosessoi paikallisesti tai verkosta tuloihin saamansa tiedon ja sen perusteella muuttaa lähtöjen tiloja.

Tietyt toiminnot tarvitsevat enemmän muistia, joten ne varaavat useamman kuin yhden paikan. Tällä toteutuksella tiedetään automaattisesti, mitkä paikat ovat sidottuja toimintoihin ja toimintojen I/O-tietoon, joten tiedon päivittämiseen tarvitsee tietää vain toiminnon nimi ja solun osoite.

4.3.2 MaCacon perustoiminta

MaCaco -protokollalla on kolme eri tapaa kommunikoida toisen laitteen kanssa.

- Pyyntö lähettää yhden viestin ja saa takaisin yhden vastauksen.
- Tilaus lähettää yhden pyynnön ja saa sen jälkeen automaattisesti tietoja alkuperäisessä viestissä määritellyn asian muutoksista.
- Pusku aloittaa kommunikaation pakotusviestillä eikä saa vastausta takaisin.

Taulukko 3. MaCaco -kehyksen rakenne (Souliss Wiki n.d.).

Käskykoodi (1 tavu)	Tunniste (2 tavua)	Aloitustavu (1 tavu)	Esitieto (1 tavu)	Tieto (Muuttuva koko)

vNet-kehysten tieto-osa koostuu MaCaco-protokollalla tehdystä kehyksestä. Pyynnön sisältävä kehys sisältää vain otsikkotiedon ja vastauksen sisältämä kehys otsikon sekä tieto-osan. Poikkeuksena pusku -viesti voi sisältää tieto-osan lähettäjältä.

Otsikkotiedoissa käskykoodi määrittää mitä vastaanottajalta halutaan. Lähettäjä antaa kehykselle tunnisteen ja vastaanottaja käyttää samaa tunnistetta vastauksessa oikean kehyksen varmistamiseksi.

Ohjelmaa kirjoittaessa ei ole pakollista ymmärtää kehysten rakennetta tai Soulissin tietorakennetta, mutta monimutkaisemmissa sovelluksissa voidaan myös käyttää kirjastoissa määriteltyjä suoraan muistipaikkoja muokkaavia tai lukevia funktioita.

4.3.3 Viestintä solulta solulle

Tämän tyypin keskustelu tapahtuu verkossa olevien laitteiden välillä pois lukien käyttäjärajapinnan laite kuten älypuhelin. Solut puhuvat toisilleen käskykoodeilla, jotka määrittävät minkä tyypin tietoa vaihdetaan ja miten se pitää tulkita.

Käskyt ovat yhden tavun pituisia (Taulukko 3.) ja noudattavat taulukon 4 tyyppejä. Solujen välisessä viestinnässä on olemassa myös ping (taulukko 4.), joka on tiedoton pyyntö ja luku tehdään pelkästään otsikkotiedoille. Ping on puhtaasti yhteyksien testaamista varten oleva toiminto.

Solulta solulle -keskustelussa aloitustavu kertoo halutun muistialueen ensimmäisen tavun, jotta käsky käsitellään oikealla alueella ja lisätieto kertoo muistialueelta halutun tiedon pituuden tavujen määränä.

Taulukko 4. Käskykoodien arvot ja niiden merkitykset (Souliss Wiki n.d.).

I/O-tieto			
0x01	Digitaalinen pyyntö	0x11	Digitaalinen vastaus
0x02	Analoginen pyyntö	0x12	Analoginen vastaus
Luetaan tuloilta tietoa.			
Tilaukset			
0x05	Tilauspyyntö	0x15	Tilauksen vastaus
Luetaan I/O-tietoa ensin kerran ja saadaan uusi tieto aina muutoksen sattuessa.			
Puskut			
0x13	Pakottaa vastaanottajan lähettämään saman viestin takaisin		
0x14	Pakottaa kirjoittamaan annetun arvon rekisteriin		
0x16	Pakottaa kirjoittamaan yhden tavun bit-wise AND-operaatiolla		
0x17	Pakottaa kirjoittamaan yhden tavun bit-wise OR-operaatiolla		
Ping			
0x08	Ping -pyyntö	0x18	Ping -vastaus
Lähetetään tiedoton viesti, johon vastataan heti.			
Virhekoodit (Lähetetään vastauksena)			
0x83	Käskykoodi ei tuettu		
0x84	Tieto alueen ulkopuolella (väärä aloitustavu)		
0x85	Tilaus hylätty		
Virhekoodeja vian etsintää varten.			

Nämä käskyt (Taulukko 4.) ovat abstraktoituja Soulissin kirjastoissa, joten niitä ei tarvitse ohjelmaa kirjoittaessa, mutta ne ovat hyödyllisiä vikatilanteissa ja uusia toimintoja luotaessa.

Esimerkiksi Send-funktio käyttää 0x14-puskukäskyä kirjoittaakseen toisen solun muistialueelle.

```
Send(vNet-osoite, Toiminto vastaanottavalla solulla, Tieto);
```

4.3.4 Yhdyskäytävän kommunikaatio

Yhdyskäytävä hoitaa käyttäjärajapinnan ja Souliss-verkon välisen keskustelun. Aloitustavun ja esitiedon (Taulukko 3.) merkitys muuttuu kuvaamaan verkkoa. Aloitustavu kertoo mistä solusta aloitetaan ja esitieto, kuinka monta solua saa lähetettävän kyselyn. Asiakasrajapinnan kanssa keskustelu ei kuulu vertaisverkkoon vaan noudattaa perinteistä asiakas ja palvelin -arkkitehtuuria. Taulukko 5 listaa yhdyskäytävän tukemat käskyt ja niiden koodit.

Taulukko 5. Yhdyskäytävän käskykoodit (Souliss Wiki n.d.).

Tila			
0x21	Tilanluku tilauspyynnöllä	0x31	Vastaus
Muistuttaa solujen tilauspyyntöä. Luetaan pyynnössä määritellyiltä soluilta muistipaikkojen tiloja.			
Toiminnot			
0x22	Toiminnon lukupyyntö	0x32	Vastaus
Kysytään toiminnot ja niiden tiedot pyynnössä määritellyiltä soluilta.			
Pakotus			
0x33	Pakottaa tietoa tuloille		
0x34	Pakottaa tietoa toiminnoille		
Kunto			
0x25	Solujen kuntopyyntö	0x35	Vastaus
Yhdyskäytävä tilaa määritellyiltä soluilta kuntotiedon yhdessä tavussa. Parametreina ovat tiedon päivitystaajuus ja yhteyden kunto. Jos tieto esitetään käyttäjälle, se näkyy kokonaislukuna jolloin 255 on paras kunto.			
Tietokannan rakenteen päivitys			
0x26	Tietokantapyyntö	0x36	Vastaus
Käyttäjän asiakasohjelma hakee kaikki yhdyskäytävään rekisteröidyt solut, solujen tuetun maksimimäärän, käytettyjen muistipaikkojen määrän ja yhdyskäytävän salliman tilausten maksimimäärän.			
Tieto			
0x27	Tietopyyntö	0x37	Vastaus
Sama kuin Tila, mutta ei muodosta tilausta.			
Yhdyskäytävän etsiminen			
0x28	Yhdyskäytävän tunnistuspyyntö	0x38	Yhdyskäytävän vastaus
Asiakasohjelma lähettää verkkoon viestin yleislähetyksenä ja yhdyskäytävä vastaa.			

4.4 Toiminnot

Toiminnot tunnetaan Soulississa nimellä typical ja ovat pääasiallinen tapa käskyttää asioita. Ne ovat funktioita, jotka suorittavat tietyn toiminnan saadun tulon perusteella ja päivittävät tulokset funktion argumenteissa määriteltyyn muistialueeseen.

Jokainen toimintoryhmä (Taulukko 6.) varaa koodit väliltä Tn1-TnF itselleen, mutta kaikki toiminnot eivät käytä jokaista ryhmälle varattua koodia.

Taulukko 6. Soulissin toimintojen tyypit (Souliss Wiki n.d.).

Toimintoryhmä	Tehtävä
T1n	Pääasiassa valojen, ovien ja porttien hallintaan.
T2n	Moottoriohjatut rullaverhot sekä rullaovet
T3n	Käytetään lämpötilatietoon ja ilmanvaihdon hallintaan
T4n	Varashälytystoiminnot
T5n	Analogisia mittauksia varten (esim. paine tai jännite)
T6n	Analogisia raja-arvoja varten

Toimintojen käyttämisen esimerkki. Kytkintoiminnon T11 paikallinen ohjaus palautuvalla kytkimellä. Toiminta toteutettu kuin fyysinen relekytkentä. Yksi painallus muuttaa lähdön tilaa.

```
#define kytkin1 0 //toiminto kytkin1 paikassa 0

void setup()
{
Initialize();
Set_T11(kytkin1);     //Määritellään toiminnon tyypiksi
                      //ON/OFF-kytkin T11.

    pinMode(2, INPUT);  //Määritellään pinni 2 tuloksi
    pinMode(11, OUTPUT);//ja pinni 11 lähdöksi.
}

void loop()
{
EXECUTEFAST(){
UPDATEFAST();
FAST_50ms() {   //Prosessoidaan 50ms välein

//Luetaan tulo ja lähtö sekä ajetaan ne käsittelevä logiikka.
            DigIn(2, Souliss_T1n_ToggleCmd, kytkin1);
            Logic_T11(kytkin1);
//Luetaan saatu lähtötieto ja ohjataan fyysistä lähtöä
            DigOut(11, Souliss_T1n_Coil, kytkin1);
        }
}
}
```

Normaalilla valokatkaisijalla toteutettuna pitää määrittää kaksi tuloa, ja kaksi DigIn-funktiota.

```
DigIn(input1, Souliss_T1n_OnCmd, kytkin1);
DigIn(input2, Souliss_T1n_OffCmd, kytkin1);
```

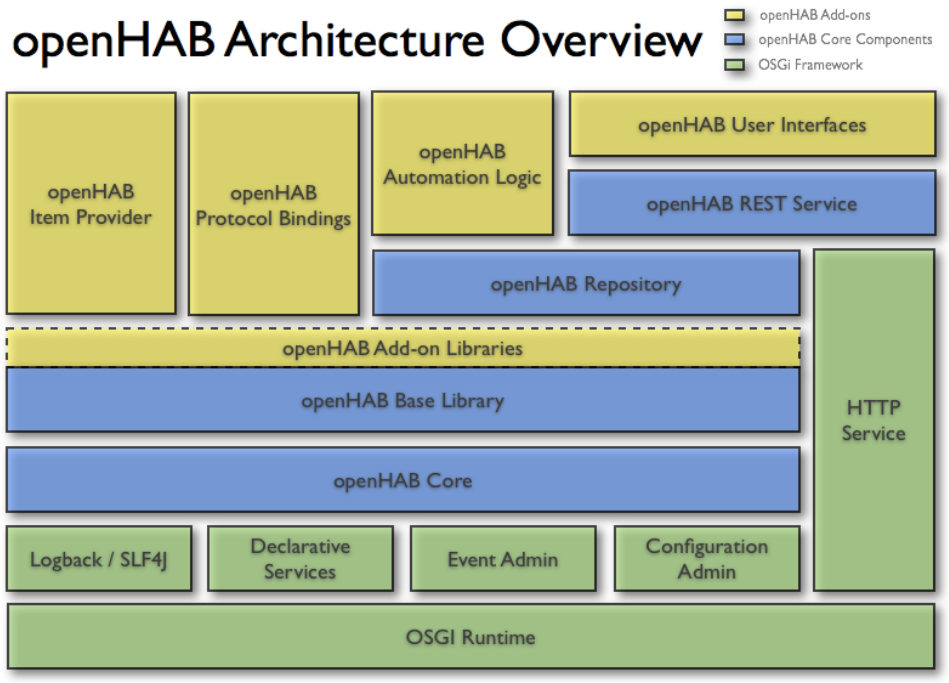
Tietyt toiminnot vaativat tietyn verran muistipaikkoja, esimerkiksi RGB LED -valaisintoiminto T16 vaatii 4 paikkaa. Ensimmäinen paikka on itse toiminnolle ja 3 seuraavaa paikkaa käytetään RGB-tiedon tallentamiseen ja käyttämiseen.

5 AUTOMAATIOPALVELIN JA PILVIPALVELU

5.1 Yleistä

openHAB on Java-pohjainen automaatiopalvelin, joka tukee useita eri automaation väylätekniikoita sekä järjestelmiä ja sitä voidaan käyttää yhdistelemään niitä saman käyttöliittymän alle.

Ohjelma pystyy myös itse suorittamaan rutiineja ja toimintoja sekä ajamaan niitä käyttäen mitä tahansa yhdistettyjä järjestelmiä ollessa samalla yhteydessä pilvipalveluun. Eri automaatiojärjestelmien yhdistämiseksi ja niiden käyttämiseksi pitää asentaa tarvittavat sidokset.



Kuva 7. openHAB -arkkitehtuurin rakenne (openHAB Wiki n.d.).

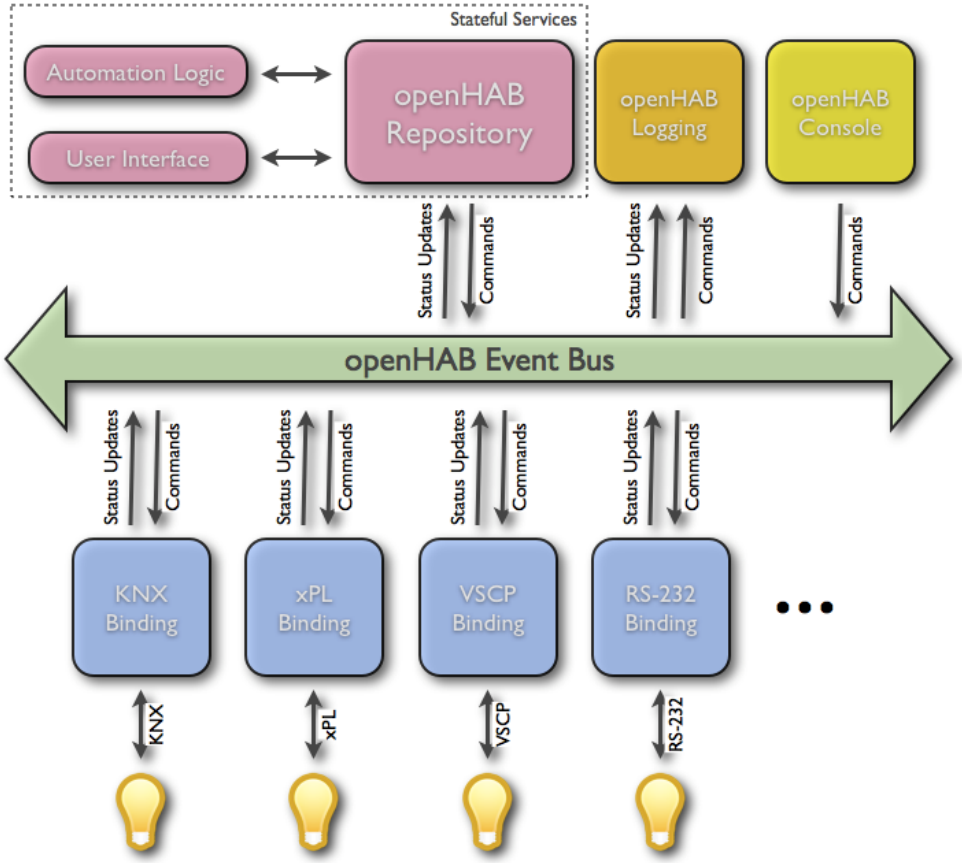
openHAB on rakennettu OSGi-rungon päälle (Kuva 7.), joka mahdollistaa ohjelman lisäosien lisäämisen ja poistamisen sekä asetusten muuttamisen ohjelman ollessa ajossa. OSGi on sidottu Javaan, joten openHAB ei pyöri ilman Java Virtual Machinea ja tarvittavia kirjastoja.

Sidokset (binding) ovat lisäosia, joita openHAB käyttää ymmärtääkseen yhdistettyjen järjestelmien, palveluiden ja laitteiden protokollia. Sidoksia on useita kymmeniä ja ne kattavat useimmat kuluttajille suunnatut automaatioprotokollat sekä laitteet.

REST-palvelu mahdollistaa yksinkertaisten http-käskyjen käyttämisen palvelimen etähallintaan. REST-rajapinnan avulla voidaan esimerkiksi rakentaa IoT-verkkoja, joissa openHAB täyttää yhdyskäytävän roolin ja yhdistää ne yhdeksi isoksi kokonaisuudeksi. Ohjelma on lisäksi hallittavissa ohjelman komentorivin, Web-käyttöliittymän, my.openHAB -pilvipalvelun sekä Android-, Windows Phone- ja iOS-applikaatioiden kautta.

5.2 Kommunikaatio

Palvelimella on kaksi tapaa hallita tapahtumia ja asioita.
Tapahtumaväylä (Event Bus) ja Asiakirjasto (Item Repository).



Kuva 8. Tapahtumaväylän toiminta (openHAB Wiki n.d.).

Tapahtumaväylä on openHABin toimintojen selkäranka. Se välittää tiedon ohjelman eri osien välillä ja sidokset yhdistävät sen kautta palvelimeen (Kuva 8.). Tapahtumaväylän pääasiallinen tehtävä on välittää tilatietoa, joten se ei itse tallenna tiloja.

Käyttäjärajapinnat ja automaatiologiikka kuitenkin tarvitsevat tiloja toimiakseen. Tätä varten asiakirjasto hoitaa kunkin asian sen hetkisen tilan tallentamisen. Tällä toteutuksella voidaan pitää huoli, että kaikki järjestelmän

osat pysyvät ajan tasalla eri asioiden tiloista eikä niiden tarvitse itse tallentaa tilatietoa. Asiakirjaston sisällön voi myös tallentaa ulkoiseen tietokantaan tai paikallisesti, josta sen hetkisen tilanteen voi hakea esimerkiksi järjestelmän kaatumisen jälkeen.

5.3 Hallinta

Järjestelmän hallintaan openHAB tarjoaa monia mahdollisuuksia, mutta tämän opinnäytetyön puitteissa tulen keskittymään vain my.openHAB -palveluun ja ohjelman sisäänrakennettuihin ominaisuuksiin. Automaatiopalvelin tukee perinteisemmin käytetyistä tekniikoista mm. KNX-väylää, EnOceania ja Modbusia. Lisäksi sitä voi ohjata http:n yli suorilla komennoilla REST-rajapinnan kautta, joka mahdollistaa itse tehdyt ratkaisut käyttäjärajapinnoille.

openHABin hallinta perustuu käyttäjärajapintaan, sääntöihin ja skriptoihin. Käyttäjärajapinta muodostuu asioista ja sivukartasta. Asiat eivät näy käyttöliittymässä, jos niitä ei ole määritelty sivukartassa. Asioiden tiloja muuttavat joko käyttäjä käyttöliittymästä, järjestelmään liitetyt ulkoiset osat tai skriptat ja säännöt.

5.3.1 Asiat

Asiat (Taulukko 7.) määritellään samalla syntaksilla riippumatta siitä mihin sidokseen se on liitoksissa. Sidoksen vaatimat lisätiedot tulevat perustietojen perään.

Asiat kirjoitetaan muodossa:
`Tyyppi Nimi ["Nimike"] [<Ikoni>] [(Ryhmä1, Ryhmä2, ...)] [{Sidostiedot}]`
Hakasulkeissa olevat tiedot ovat valinnaisia.

Esimerkki:
`Switch Kytkin1 ”Kuistin valo” (Ulkovalot) {souliss=”T11,0,0”}`
Asia nimeltä Kytkin1 ohjaa kytkintoimintoa T11 muistipaikassa nolla solussa nolla.

Kun asia on määritelty se saa kyseisen asiatyypin perustoiminnot eikä käyttäjän tarvitse tehdä skriptoja tai sääntöjä, jos monimutkaisempaa ohjausta ei tarvita.

openHABin asiat toimivat ja ovat jaoteltu samankaltaisesti Soulissin toimintojen kanssa mikä helpottaa suuremman kokonaisuuden rakentamista, koska voidaan käyttää samoja termejä kuvaamaan molempien toimintaa.

Taulukko 7. openHAB version 1.8 asiat (openHAB Wiki n.d.).

Asia	Tietotyyppi	Käsky(t)	Kuvaus
Call	Call	-	Puhelinominaisuuksia varten
Color	OnOff, Percent, HSB	OnOff, IncreaseDecrease, Percent, HSB	Värinvalinta
Contact	OpenClosed	-	Binäärinen kontaktitieto
DateTime	DateTime	DateTime	Aikatieto
Dimmer	OnOff, Percent	OnOff, IncreaseDecrease, Percent	Himmennin
Group	Sama kuin siihen kuuluvalla asialla tai asioiden yhteisellä tyypillä	Sama kuin siihen kuuluvalla asialla tai asioiden yhteisellä tyypillä	Voidaan ryhmittää useampi asia ja muokata saman tyyppisiä asioita
Location	Point	Point	Käytetään tallentamaan koordinaatteja
Decimal	Decimal	Decimal	Käytetään numeroiden tallennukseen esim. lämpötila
RollerShutter	UpDown, Percent	UpDown, StopMove, Percent	Esim. rullaoven hallintaan
String	String, DateTime	String	Mikä tahansa merkkijono tai DateTime merkkijonona
Switch	OnOff	OnOff	Normaali kaksiasentoinen kytkin

5.3.2 Säännöt ja skriptat

Säännöt ja skriptat mahdollistavat automatisoitujen toimintojen ja rutiinien ajamisen automaatiopalvelimessa itsessään. Niiden pääasiallinen ero on, että skripta on yksittäinen tiedosto, joka kutsutaan erikseen ja se palauttaa arvon. Skriptan sisällä täytyy olla kaikki sen tarvitsemat määritykset (Scripts n.d..)

Säännöt muistuttavat Javaa ja niissä on kirjastojen tuonti sekä kun jotain tapahtuu niin tee jotain rakenne. Yksittäisen säännön suoritusosa on sama kuin skripta, mutta sen voi kirjoittaa suoraan säännön sisään. (Rules n.d..)

```
var Number x //voidaan määritellä muuttujia ja tuoda kirjas-
toja koodin yläosassa kuten Javassa

rule esim //Määritellään sääntö nimellä esim

when //Listataan ehdot ja tarkastetaan ne
System started or
Item Asia1 changed or
Time is morning

then //Ajetaan säännön skripta
x = x + 1
say(”Moi”)
sendCommand(Asia2, ON)
callScript("Skripta1") //Voidaan myös kutsua skriptoja
end //Lopetetaan suoritus
```

Käytettävissä olevia käskyjä asioiden tilojen muokkaamiseen ja muihin toimintoihin on lukuisia ja niitä voi halutessaan asentaa lisää tiettyjä sidoksia varten lisäosana.

5.3.3 Sivukartta

Sivukartta määrittää, mitä näytetään käyttäjälle. Niitä voi tehdä useita ja näyttää eri käyttäjille eri tietoja. Jos sivukarttaa ei määritetä, ei openHAB osaa luoda käyttöliittymää, joten järjestelmää ei voi hallita normaalisti.

Sivu rakennetaan elementeistä (Taulukko 8.) ja perinteisen nettisivun tapaan pitää määrittää, mitkä asiat näytetään milläkin sivulla ja samoissa ryhmissä. Erona perinteiseen sivuun web-palvelin hoitaa graafisen ulkoasun rakentamisen ja kytkee elementit niitä vastaaviin asioihin.

```
sitemap Esim [label="Päävalikko"]

Frame label="Autotalli" {
                Group item=Valot
                Text item=Lämpötila
        }
```

Esimerkissä määritetään sivukartta nimeltä Esim, joka näkyy käyttäjälle Päävalikkona ja sisältää Autotalli nimisen osion. Autotallin alla on kaksi osaa. Valot vie uuteen valikkoon, joka näyttää kaikki ryhmän Valot alla olevat asiat. Lämpötila näyttää sen hetkisen lämpötilan tekstimuodossa.

Sivukartan elementit vastaavat asioita, vaikka niiden nimikkeet eivät ole täysin samat. Esimerkiksi Color tyypin asiaa ohjataan Colorpicker elementillä.

Taulukko 8. Sivukartan elementit (openHAB Wiki n.d.).

Elementti	Kuvaus
Colorpicker	Antaa käyttäjän valita värin värirenkaasta
Chart	Piirtää kuvaajan mitattavasta asiasta ajan funktiona
Frame	Määrittää uuden osion elementeille
Group	Näyttää kaikki määriteltyyn ryhmään kuuluvat asiat
Image	Näyttää kuvan annetusta lähteestä
List	Ei tuettu tällä hetkellä
Selection	Määrittelee uuden valikon, jossa käyttäjä voi valita valmiiksi määritellyistä arvoista
Setpoint	Näyttää raja-arvot, joita voi muokata; minimi, maksimi ja askel
Slider	Näyttää liukusäätimen
Switch	Näyttää kahden tilan kytkimen
Text	Näyttää tekstiä
Video	Näyttää videon lähteestä
Webview	Näyttää web-sisältöä lähteestä

5.4 my.openHAB-pilvipalvelu

5.4.1 Yhdistäminen ja turvallisuus

openHAB ei ole pelkkä sisäverkon automaatiopalvelin vaan tarjoaa myös IoT:lle elintärkeän pilvipalvelun ilmaiseksi ja mahdollistaa turvallisen etäohjauksen, jossa ohjelma hoitaa yhdistämisen automaattisesti pilveen perustuen kahteen ensimmäisellä käynnistyskerralla luotuun koodiin, jotka sidotaan käyttäjätiliin.

Käyttäjän pitää myös ladata tarvittava sidos ja siirtää se sidoskansioon, sekä luoda persistence-kansioon tiedosto nimellä openhab.persist, ja määritellä mitä asioita ohjelma päivittää pilvipalveluun. Tällä hetkellä my.openHAB tukee vain yhtä ohjelmaa per tili, joten hajautetussa ratkaisussa pitää yhden toimia pääpalvelimena, joka ohjaa muita http-käskyillä REST-rajapinnan kautta. Nämä käskyt voidaan määritellä asioiksi ja sitä kautta päivittää palveluun.

5.4.2 IFTTT-integraatio

my.openHAB on myös yhdistettävissä IFTTT-palveluun. Nimi tulee sanoista ”If This Then That”, joka kuvaa palvelun toimintamallia. IFTTT tarjoaa automatisoinnin liittämisen satoihin eri verkkopalveluihin kuten Google drive, Twitter, Instagram ja verkkokauppojen hintaseurannat.

Tämä mahdollistaa mitä vain -osan toteuttamisen IoT:n konseptista ja kattavia palvelu- ja tekniikkariippumattomia ohjauksia sekä turvallisen ja yksinkertaisen käyttöliittymän loppukäyttäjälle.

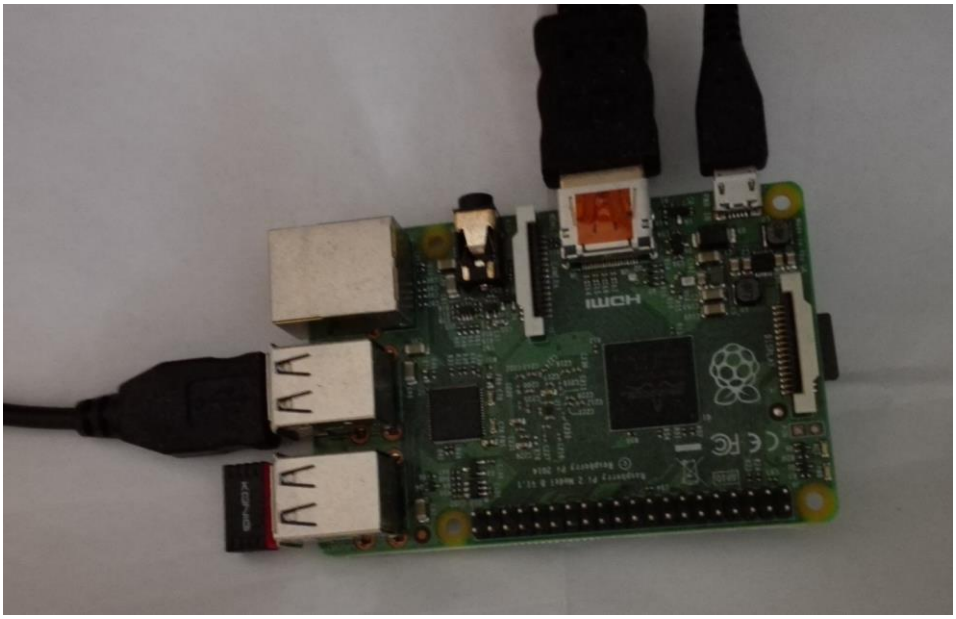
6 IOT-HARJOITUSJÄRJESTELMÄ

6.1 Järjestelmän kokoonpano

Laitteet on valittu ottaen huomioon tulevaisuuden skaalaamisen ja mahdollisten uusien harjoitusten monipuolisuuden. Sekä openHAB, että Souliss pystyvät yhdistämään useiden valmistajien laitteita yhteen kokonaisuuteen, joten niillä voi toteuttaa tulevaisuudessa laajempia IoT-harjoitusprojekteja.

Liitteissä käydään läpi mahdollisia kokoonpanoja ja laitehankintojen hintaa.

6.1.1 Raspberry Pi



Kuva 9. Raspberry Pi 2 käyttövalmiina

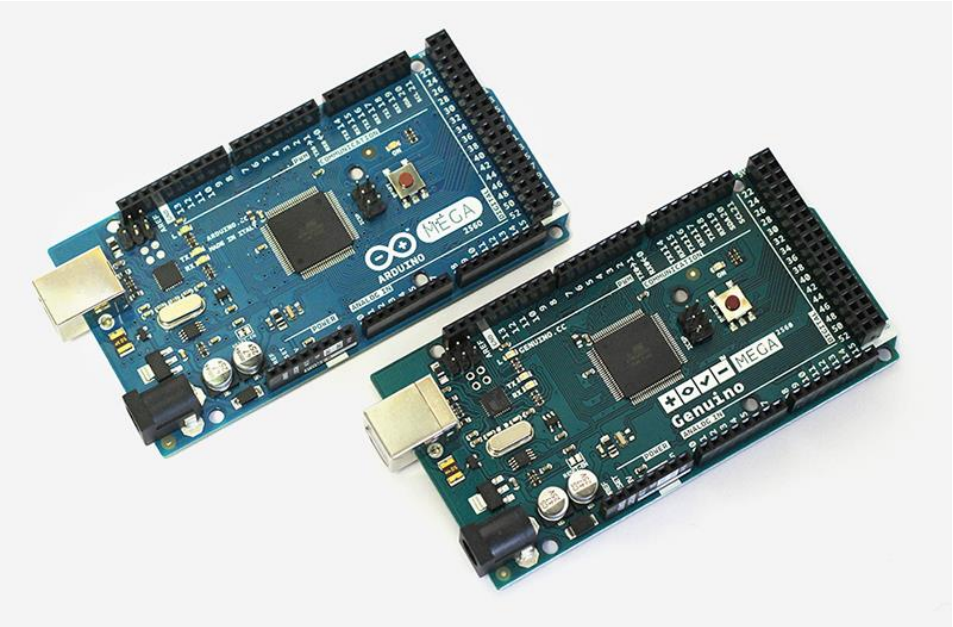
Raspberry Pi (Kuva 9.) on yhden piirilevyn tietokone luottokortin kokoisella pinta-alalla. Siihen on sisällytetty ohjelmoitavia I/O-pinnejä erilaisten ohjaus- ja sensoripiirien rakentamista varten.

Alun perin Pi luotiin helpoksi ohjelmoinnin ja tietokoneen toiminnan opetusvälineeksi, mutta sitä käytetään laajalti projekteissa, joissa tarvitaan pientä vapaasti ohjelmoitavaa tietokonetta.

Tähän työhön Raspberry on hyvä valinta, koska harjoitustehtävä on hyvä tapa tutustuttaa oppilaat sen ja Linuxin perustoimintaan. Koululla on niitä jo valmiiksi, joten ylimääräisiä hankintoja ei välttämättä tarvitse tehdä ja ne pystyvät ajamaan openHABia.

openHAB tukee myös Raspberryn omia I/O-pinnejä sidoksen kautta, joten se voi myös ohjata laitteita. Lisäksi laitetta voi käyttää muihin harjoituksiin, kuten Web-palvelimella ohjattuihin laitteisiin tai yksinkertaiseen robottiin.

6.1.2 Arduino



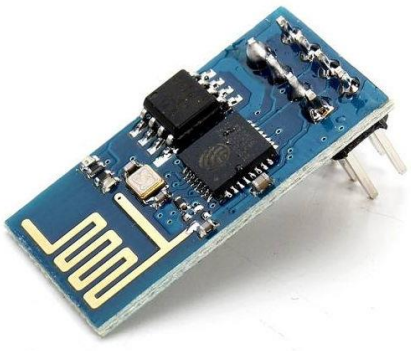
Kuva 10. Arduino Mega 2560 ja sen Yhdysvaltojen ulkopuolella myytävä sisarmalli (Arduino.cc n.d.).

Arduinot (Kuva 10.) ovat avoimia Atmelin mikroprosessoreihin perustuvia laitteita, jotka on tarkoitettu nopeaan prototyyppien iterointiin ja ideoiden testaamiseen.

Niiden vahvuuksia ovat halpa hinta, valmiit I/O-portit piirilevyllä ja lukuisat kirjastot eri toiminnoille, joten kaikkea ei tarvitse lähteä tekemään tyhjästä. Laitteen toimintaa on helppo muokata ohjelmistopohjaisesti valmiita laajennuskirjastoja ja C- ja C++-ohjelmointikieliä käyttäen.

Arduino valittiin harjoitusjärjestelmään, koska Souliss on luotu sen päälle ja koululla valmiiksi kappaleita varastossa, joten uusia hankintoja ei tältä osin tarvitse tehdä. Järjestelmä käyttää kahta Arduinoa.

6.1.3 ESP8266



Kuva 11. ESP-01-moduuli (ESP8266 Community Wiki n.d.).

On myös mahdollista käyttää ESP8266-piiriin perustuvaa laitetta (Kuva 11.) ja ladata Souliss suoraan sille. Sitä voi käyttää reitittämään liikennettä USART:n kautta Arduinolle tai itsenäisesti ohjaukseen WiFi:n kautta.

ESP on WiFi-moduulien tuoteperhe, joissa on ohjelmoitava prosessori, I/O-pinnejä sekä runsaasti muistia Arduinoon verrattuna. Se on optimaalinen ratkaisu Souliss-verkon yhdyskäytäväksi.

Tällä hetkellä Souliss tukee vain vanhempaa ESP-laitekantaa, eikä kaikkien valmistajien malleja ole testattu, mutta päivitys laitetukeen on luvattu seuraavaan Souliss-päivitykseen.

ESP-moduulien sarjaliikennöinti tapahtuu USART-väylällä, joten sen voi liittää RS485-verkkoon muiden Souliss-laitteiden kanssa. Tämä mahdollistaa moduulien käyttämisen siltana IP-verkon ja sarjaverkon välillä.

ESP-moduulien liikennöinti toimii 3.3V jännitteellä eivätkä ne kestä suurempia arvoja. Jos moduuli kytketään suoraan Arduinoon, tarvitaan RS485-lähetin/vastaanotin, jänniteregulaattori tai jännitteen jako vastuksilla.

6.1.4 Yhteydet

Raspberryn yhteys lähiverkkoon muodostetaan langattomasti WiFi-verkon yli tai Ethernet-kaapelilla. Yhdistäminen langattomasti tapahtuu erillisellä USB WiFi -adapterilla.

Arduinojen langattomassa liikennöinnissä toiseen niistä liitetään nRF24L01-piiri, jolla se pystyy keskustelemaan langattomasti. Tämä ratkaisu vaatii myös Arduino Ethernet Shieldin tai muun Ethernet-moduulin, jotta voidaan yhdistää lähiverkkoon. Jäljelle jäävä Arduino yhdistetään SPI-väylällä toiseen nRF24L01-piiriin ja lähiverkosta tuleva liikenne reititetään ensimmäisen Arduinon läpi. Näin voidaan toteuttaa eri yhteystekniikoiden yli toimiva vertaisverkko.

On myös mahdollista yhdistää Arduinot sarjaliikennettä käyttävän USART-väylän kautta, jos ei haluta hankkia erillisiä piirejä. ESP-moduulit ovat suoraan yhteydessä lähiverkkoon WiFi:n kautta.

6.2 Järjestelmän toiminta opetuksessa

Harjoitustehtävät voidaan jakaa viiteen osaan:
- kysymyksiä asioiden internetistä ja järjestelmän osista
- ohjelmistojen käyttöönotto ja testaus
- toteutus pelkällä Soulissilla
- toteutus openHABilla ja Soulissilla
- my.openHAB-pilvipalveluun liittäminen ja ulkoinen ohjaus.

Harjoitusjärjestelmä perustuu puhtaasti Arduinoon ja USART-väylään. Esimerkkejä vaihtoehtoisista kokoonpanoista ja niillä pyörivistä ohjelmista on liitteessä 2. Liite 1 käy läpi mahdollisia hintoja järjestelmälle.

Tällä järjestelmällä voidaan myös toteuttaa eri kokoonpanoja, mutta nämä viisi osa-aluetta kattavat IoT:n yleisen idean ja toiminnan sekä tarvittavat tekniset taidot. Lisätehtäväksi voi esimerkiksi ehdottaa Soulissin ajamista MathWorksin ThingSpeak-pilvipalvelun kautta openHABin sijasta.

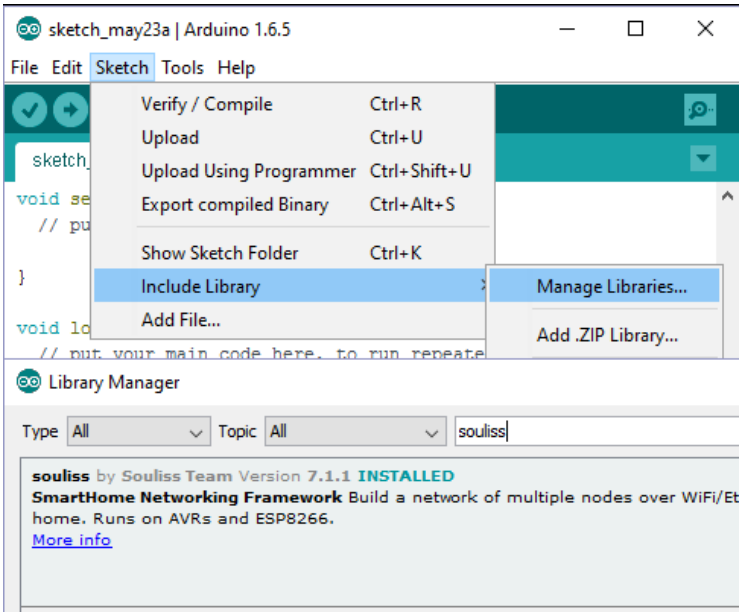
Jatkossa openHAB voi myös olla yhdyskäytävä monen eri automaatioväylän verkossa monimutkaisempien harjoitusaiheiden tai esiteltävien järjestelmien toteuttamiseksi.

Ohjaukseen kannattaa myös liittää oikeita laitteita esimerkiksi ledeillä simuloimisen sijaan, jotta saadaan tuntuma siitä, mitä järjestelmä tekisi käytännön projekteissa.

Tämän työn liitteissä esitellään lisää vaihtoehtoja, ja kokoonpanoihin liittyvät valinnat voidaan tehdä myöhemmin. Itse järjestelmän toimintaan kokoonpanot eivät vaikuta johtuen käytettävien ohjelmistojen tekniikkariippumattomuudesta, joten harjoituksia voidaan toteuttaa monipuolisesti.

7 KÄYTTÖÖNOTTO JA TESTAUS

7.1 Souliss

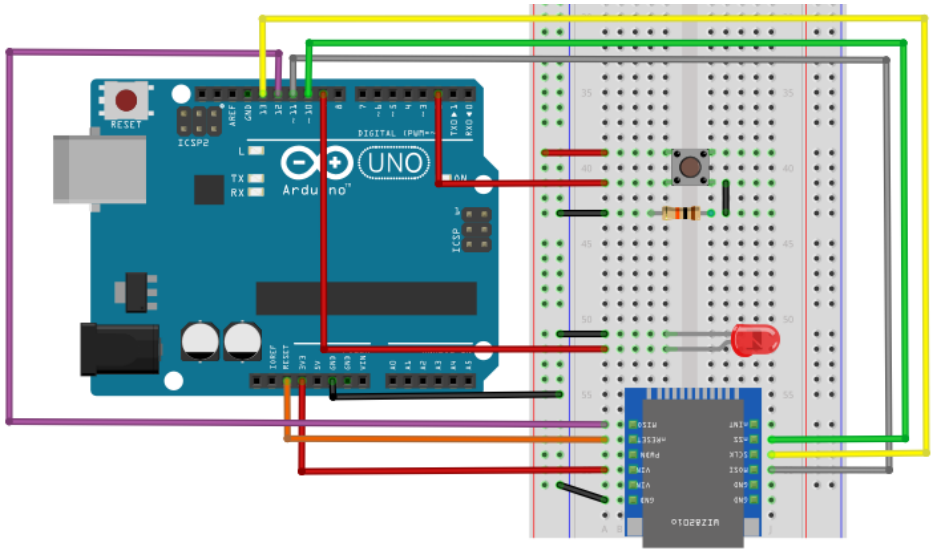


Kuva 12. Souliss asennettuna Arduino IDE 1.6.5:lle

Tällä hetkellä Souliss toimii parhaiten, kun on käytössä Arduino IDE:n versio 1.6.5 ja Souliss 7.1.1. Uudempaa kehitysympäristöä käytettäessä pitää Souliss-kansioon luoda tyhjä header-tiedosto nimellä SoulissBase ja viitata siihen #include-komennolla ensimmäisenä ohjelmassa tai Arduino IDE ei osaa hakea kirjastoja oikein.

Souliss-kirjastojen käyttöönottamiseksi Arduino IDE:ssä pitää navigoida kirjastomanageriin kuvan 12 mukaisesti.

Souliss löytyy hakukenttään kirjoittamalla. Kun kirjastot ovat ladattuna valitaan esimerkkivalikosta Souliss ja Ethernet-esimerkeistä Hello World. Valitaan käytettävän Ethernet-moduulin kirjasto ohjelman alussa kommenttien ohjeiden mukaan.



Kuva 13. Hello World -ohjelman vaatimat fyysiset kytkennät

Kytkennät tehdään kuvan 13 mukaisesti. Ohjelmassa on valmiina määritettynä pinni 2 tulona ja pinni 9 lähtönä. Jos käytössä on Wiz820io pitää ajurikirjasto vaihtaa W5200-piirille sopivaksi.

Jos DHCP on käytössä pitää ohjelmaan kirjoittaa seuraavat asiat.

Ohjelman alkuun:

```
#include "conf/ethW5200.h" //Jos käytössä Wiz820io
#include ”conf/Webhook.h”
```

Setup-funktion sisään:

```
GetIPAddress();
SetAsGateway(myvNet_dhcp);
```

Manuaalisesti määritettäessä verkon osoitteisto tulee määrittää ennen Setup-funktiota ja tarvittaessa MAC-osoite ennen Souliss.h:n tuontia.

```
…
#define MAC_INSKETCH
uint8_t MAC_ADDRESS[] = {0x1A, 0xA6, 0x49, 0x6B, 0x00, 0x01};
…
uint8_t ip_address[4]  = {192, 168, 1, 77};
uint8_t subnet_mask[4] = {255, 255, 255, 0};
uint8_t ip_gateway[4]  = {192, 168, 1, 1};
#define myvNet_address  ip_address[3]

void setup(){
SetIPAddress(ip_address, subnet_mask, ip_gateway);
SetAsGateway(myvNet_address);
…
}
```

Hello World -ohjelma ladataan Arduinoon ja se liitetään lähiverkkoon. Ladataan puhelimeen Souliss App ja käynnistetään se. Souliss App tarkistaa käynnistyksessä, onko verkkoa jo määritetty ja vie käyttäjän oikealle sivulle, jos ei. Tehdään määritys ohjelman antamien ohjeiden mukaan ja haetaan verkosta soluja. Jos ongelmia ei ole, niin esimerkkitoiminnon pitäisi tulla näkyviin Node 0:n alla Manual-välilehteen.

Jos solua ei näy niin, yleisimmät ongelmat ovat väärän Ethernet-moduulin laitekirjasto, väärin tehdyt määritykset tai reititin saa jostain syystä yhteyden Arduinoon vain staattisesti määritetyillä osoitteilla.

7.2 openHAB ja my.openHAB-pilvipalvelu

7.2.1 Asennus

Windows-ympäristössä riittää, kun asentaa uusimman JDK:n ja sen jälkeen openHABin Windows-version.

Raspberry käyttää Linuxia, jossa asennus tapahtuu komentoriviltä. Tämä ohje on tehty Raspbian-distribuutiolle, joka on yleisin Linux-distribuutio kyseisellä Raspberry Pi:llä. Vaihtoehtoisesti voi ohjelman ladata ja purkaa suoraan työpöydälle, josta sen pitäisi toimia suoraan.

openHAB designer -kehitysympäristö pitää aina ladata erikseen. Tiedostoja voi myös muokata normaalilla tekstinkäsittelyohjelmalla.

Terminaaliohjelmassa tehdään seuraavat vaiheet:

Varmistetaan, että Java on versio 1.6 tai uudempi.
sudo update-alternatives --config java
Valitaan jdk8

Päivitetään Raspbian
sudo apt-get update
sudo apt-get upgrade

Lisätään openhab pakettilistaan ja suoritetaan asennus
wget -qO - 'https://bintray.com/user/downloadSubjectPublicKey?username=openhab' |sudo apt-key add -

echo "deb http://dl.bintray.com/openhab/apt-repo stable main" | sudo tee /etc/apt/sources.list.d/openhab.list

sudo apt-get update
sudo apt-get install openhab-runtime
sudo update-rc.d openhab defaults

Varmistus, että openhab-kansioiden omistaja on openHAB, eikä root
sudo chown -hR openhab:openhab /etc/openhab

sudo chown -hR openhab:openhab /usr/share/openhab

Demo-talon tiedostot ladataan ohjelman kotisivuilta ja puretaan testaamista varten

start.sh:sta tehdään executable ja käynnistetään openhab
sudo chmod +x start.sh
sudo ./start.sh

Testaa openHABin toimivuus lähiverkon tai paikallisen osoitteen kautta
http://192.168.X.XXX:8080/openhab.app?sitemap=demo
http://127.0.0.1:8080/openhab.app?sitemap=demo

7.2.2 Sidosasetukset

Kopioidaan openhab-default.cfg ja uudelleen nimetään se openhab.cfg. Tämä tiedosto sisältää kaikki perus- ja sidosasetukset.
Siirretään Souliss- ja my.openHAB-sidokset sidoskansioon.

7.2.3 Souliss

Tiedostossa openhab.cfg on kohta ”Souliss binding”. Souliss-asetusten edestä pitää poistaa #-merkit, jotta ne tulevat voimaan.

Vähintään 3 asetusta vaatii käyttäjän huomion; IP_LAN, USER_INDEX sekä NODE_INDEX.
IP_LAN on Souliss-yhdyskäytävän osoite. USER_INDEX ja NODE_INDEX eivät saa olla samat minkään muun käyttäjärajapinnan kanssa. Android-sovelluksen indeksinumerot voi tarkistaa sovelluksesta itsestään.

7.2.4 my.openHAB

Sidos ladataan osoitteesta http://my.openhab.org, jossa myös luodaan käyttäjätili. Ohjelman tarkistuskoodit palveluun, UUID ja Secret, löytyvät kansiosta openhab/webapps/static kun openHAB on käynnistetty kerran.

openHAB vaatii asetuksiin kolme muutosta. Asetetaan security:option=EXTERNAL ja lisätään kohtaan security:netmask= oman lähiverkon aliverkon peite, sekä asetetaan persistence:default=myopenhab.

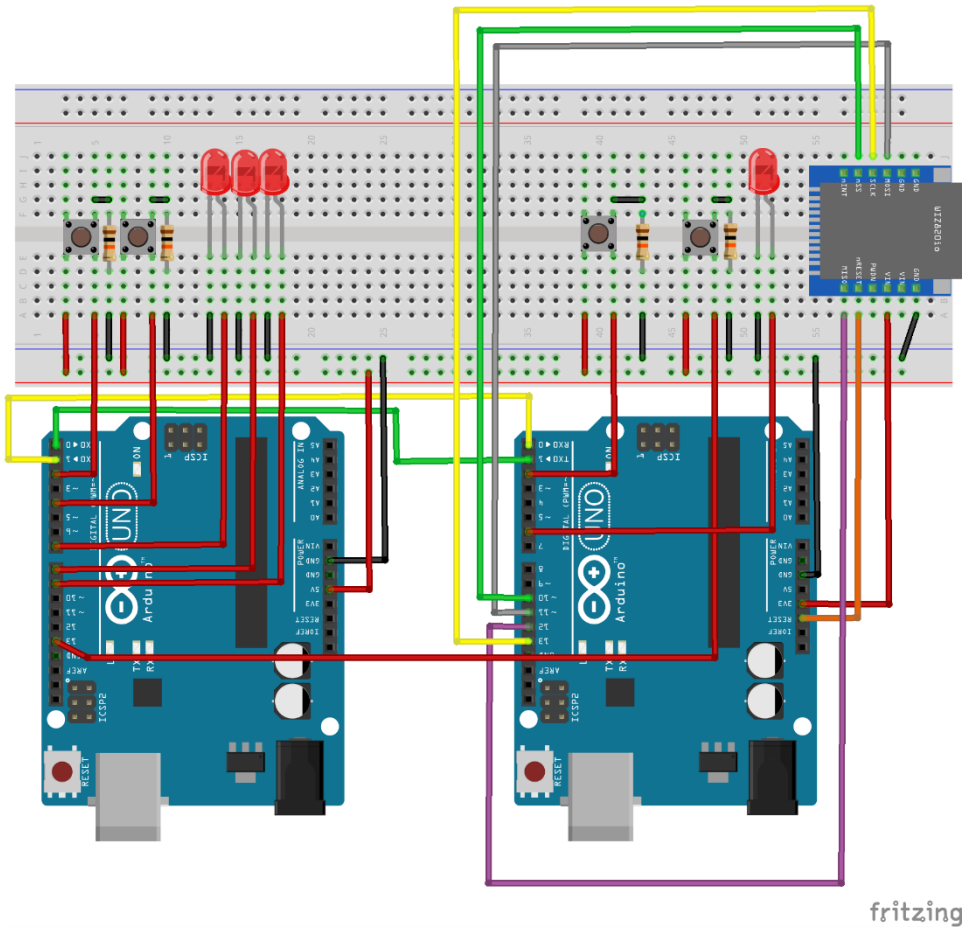
Persistence kansioon luodaan tiedosto myopenhab.persist.
Tiedoston sisältö voi olla esimerkiksi:

```
Strategies {
            default = everyChange
}
Items {
            * : strategy = everyChange
}
```

Tämä päivittää jokaisen asian jokaisen muutoksen pilvipalveluun.

8 HARJOITUSTEHTÄVÄN ESIMERKKI

8.1.1 Souliss



Kuva 14. Esimerkkijärjestelmän kytkennät

Avaa esimerkeistä USART-esimerkki ja lisää yhdyskäytävän ohjelmaan valo-ohjaus. Esimerkissä on käytössä Wiz820io-moduuli IP-verkkoa varten ja se toimii W5200-piirin kirjastolla. Jos käytössä on Ethernet Shield, pitää tarkistaa mihin piiriin se perustuu.

Kuten kuvassa 14 näkyy, pitää tuloilla olla veto maahan vastuksilla virheiden välttämiseksi. Ledeille voi laittaa etuvastukset varmuuden vuoksi, mutta Arduinon lähdöt eivät riko ledejä, vaikka vastuksia ei olisi. Vasemman puoleiset 3 lediä kuvaavat esimerkkiohjelman rullaoven tiloja ja oikeanpuoleinen ledi kattovaloa.

Lisäykset yhdyskäytävän ohjelmaan:

```
…
#include "conf/ethW5200.h"
…
#define valo1         2
#define valo1_in      3
#define valo1_out     4
…

void setup(){
…
pinMode(valo1_in, INPUT);
pinMode(valo1_out, OUTPUT);
Set_T11(valo1);
…
}

void loop(){
EXECUTEFAST(){
UPDATEFAST();
FAST_510ms(){
…
DigIn(valo1_in, Souliss_T1n_ToggleCmd, valo1);
Logic_T11(valo1);
DigOut(valo1_out, Souliss_T1n_Coil, valo1);
…
}}}
```

Lisäys soluun:

```
…
#define valo1         2
#define valo1_in      13
…
void setup(){
…
pinMode(valo1_in, INPUT);
…
}
```

Seuraava muiden toimintojen edelle FAST-funktiossa.

```
…
//Yhdyskäytävä ohjaa releitä sähkökaapissa, ja
//solu on autotallissa reitittämässä käskyä.
if(DigIn(valo1_in, Souliss_T1n_ToggleCmd, valo1))
{
Send(Gateway_RS485, valo1, mInput(valo1));
ResetInput(valo1);
}
…
```

Solussa pitää myös GarageDoor-toiminto muuttaa toiminnoksi T22, koska openHAB tukee T2n-ryhmästä vain toimintoa T22. GarageDoor on käyttäjäystävälliseksi naamioitu T21-toiminto.

Setup-funktioon:

```
Set_T22(GARAGEDOOR_NODE2);
```

FAST-funktioon:

```
Logic_T22(GARAGEDOOR_NODE1);
```

8.1.2 openHAB

Automaatipalvelin tarvitsee vähintään sivukartan ja asioiden määritykset toimiakseen. Sääntöjä ja skriptoja ei tarvitse käyttää, jos haluaa, että Souliss hoitaa kaiken automatisoinnin. Tällöin ei olla välttämättä riippuvaisia Ethernet-lähiverkosta automatisoitujen tapahtumien toiminnassa.

Asioiden määritykset items-tiedostoon:

```
Group All
Group gAutotalli (All)

/*autoupdate=”true” jos halutaan, että openHAB hakee muutok-
sia tiloihin automaattisesti*/

Rollershutter      Rullaovi      "Rullaovi"      (gAutotalli)
{souliss="T22:1:0", autoupdate="false"}

Switch     garagelight "Rullaoven valo" <switch> (gAutotalli)
{souliss="T11:1:1", autoupdate="false"}

Switch     valo1      "Kattovalo"    <switch>    (gAutotalli)
{souliss="T11:0:2", autoupdate="false"}
```

Asioiden määritykset sitemap-tiedostoon:

```
sitemap demo label="Talo"
{
           Frame label="Autotalli" {
           Switch item=Rullaovi label="Rullaovi"
           Switch item=garagelight label="Rullaoven valo"
           Switch item=valo1 label="Kattovalo"
           }
}
```

9 LOPPUYHTEENVETO

9.1 Tulokset

Tämän opinnäytetyön tarkoituksena oli saada vastaus sille mitä asioiden internet on ja mitä se pitää sisällään, sekä antaa Hämeen ammattikorkeakoululle materiaalia opetusta ja harjoitustehtävien luomista varten. Työssä kesti pidempään kuin oli aluksi suunniteltu, mutta saavutettiin asetetut tavoitteet.

IoT on aihe, jossa voi helposti hukkua tiedon määrään, mutta tämä työ tuo esiin pääkohtia, jotta aiheeseen sisäistäminen helpottuisi. Lukija saa käsityksen siitä, minkälainen on asioiden internetinä pidettävä laitteiden ja verkkojen kokonaisuus sekä kuvan alan toimijoista ja haasteista.

Haastavinta aiheen tutkimisessa oli löytää luotettavia lähteitä, jotka eivät ole teollisten toimijoiden ovien takana. Varsinkin turvallisuusnäkökulmasta kertovat avoimet julkaisut olivat harvassa.

Työ antaa opettajalle pohjatiedon IoT:stä, järjestelmän ohjelmistoista, valituista laitteista ja valmiin harjoitusjärjestelmän, joka on helppo ottaa käyttöön. Tässä on onnistuttu ja aihetta opettavalla pitäisi olla työkalut osaamisensa kehittämiseen asioiden internetin kanssa.

9.2 Kehitettävää

Tämä työ on vain pintaraapaisu siitä tietomäärästä, mitä asioiden internetistä on olemassa. Standardit ja määritykset tulevat myös todennäköisesti muuttumaan. Jää siis aihetta opettavan vastuulle seurata alaa ja päivittää tietämystään jatkossa.

Järjestelmän osat on valittu tulevaisuuden laajennuksia ajatellen ja esimerkiksi openHAB voi yhdistää eri väylätekniikoita kuten KNX ja Modbus saman käyttöliittymän alle. Vanhanaikaisessa automaatiossa eri valmistajat saattavat käyttää eri tekniikoita tiedonsiirtoon, joten harjoitus, jossa pitää ohjata kahta eri väylää samalta käyttöliittymältä voi olla hyödyllinen.

Myös uusia IoT-harjoituksia voidaan harkita muilla toteutuksilla. Opinnäytetyössä mainitaan IFTTT-palvelu, CoAP-protokolla ja ThingSpeak-palvelu. Niitä hyödyntämällä voidaan rakentaa uusia harjoituksia, jotka vievät opiskelijan syvemmälle aiheeseen.

Laajempi IoT-harjoitus tai projekti demonstraatioympäristöä varten voidaan rakentaa Soulissin ja openHABin varaan. Hankkimalla ESP-moduuleita tai langattomia lähetin-vastaanottimia Arduinoille voidaan asioita sijoittaa esimerkiksi eri puolille laboratoriotilaa, jotta saadaan luotua IoT:lle keskeinen hajautettu laiteverkko.

LÄHTEET

Arduino.cc (n.d.)
Arduino Mega 2560 ja sen Yhdysvaltojen ulkopuolella myytävä sisarmalli
Viitattu 22.05.2016
https://www.arduino.cc/en/Main/ArduinoBoardMega2560

Elisa IoT (n.d.)
Elisa Oyj
Viitattu 01.4.2016.
https://oma.elisa.fi/yrityksille/info/tuotteet-ja-palvelut/tuotteet/elisa-iot/

ESP8266 Community Wiki (n.d.).
ESP-01-moduuli.
Viitattu 22.05.2016
http://www.esp8266.com/wiki/doku.php?id=esp8266-module-family

ETNO views on M2M. 2014.
M2M-ekosysteemi.
Viitattu 20.2.2016.
https://www.etno.eu/home/positions-papers/2014/305

ETSI TS 102 689 V1.1.1. (2010-08) Machine-to-Machine communications (M2M); M2M service requirements. 2010.
European Telecommunications Standards Institute.
Viitattu 15.12.2015.
http://www.etsi.org/deliver/etsi_ts/102600_102699/102689/01.01.01_60/ts_102689v010101p.pdf

Internet Of Things: Privacy & Security in a Connected World. 2013.
Federal Trade Commission.
Viitattu 10.5.2016
https://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf

Towards a definition of the Internet of Things (IoT) Revision 1. 2015.
Institute of Electrical and Electronics Engineers.
Viitattu 15.12.2015.
http://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf

IEEE-SA Internet of Things (IoT) Ecosystem Study. 2014.
Institute of Electrical and Electronics Engineers.
Viitattu 20.2.2016
http://standards.ieee.org/innovate/iot/study.html

Recommendation ITU-T Y.2060 Overview of the Internet of things. 2010. International Telecommunications Union.
Viitattu 15.12.2015.
https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.2060-201206-I!!PDF-E&type=items

Scripts (n.d.).
openHAB Wiki.
Viitattu 15.3.2016
https://github.com/openhab/openhab/wiki/Scripts

Rules (n.d.).
openHAB Wiki.
Viitattu 15.3.2016
https://github.com/openhab/openhab/wiki/Rules

openHAB Wiki (n.d.).
openHAB -arkkitehtuurin rakenne.
Viitattu 15.3.2016
https://github.com/openhab/openhab/wiki

openHAB Wiki (n.d.).
Tapahtumaväylän toiminta.
Viitattu 15.3.2016
https://github.com/openhab/openhab/wiki

openHAB Wiki (n.d.).
openHAB version 1.8 asiat.
Viitattu 15.3.2016
https://github.com/openhab/openhab/wiki/Explanation-of-items

openHAB Wiki (n.d.).
Sivukartan elementit.
Viitattu 15.3.2016
https://github.com/openhab/openhab/wiki/Explanation-of-Sitemaps

Souliss Wiki (n.d.).
Tuetut kommunikointiväylät ja niiden osoitealueet.
Viitattu 25.2.2016
https://github.com/souliss/souliss/wiki/Addressing

Souliss Wiki (n.d.).
vNet-kehyksen rakenne.
Viitattu 25.2.2016
https://github.com/souliss/souliss/wiki/vNet%20Details

Souliss Wiki (n.d.).
Tietorakenne ja sen toiminnan kuvaus.
Viitattu 25.2.2016
https://github.com/souliss/souliss/wiki/Data%20Structure

Souliss Wiki (n.d.).
MaCaco -kehyksen rakenne.
Viitattu 25.2.2016
https://github.com/souliss/souliss/wiki/MaCaco%20Light-Data%20Protocol

Souliss Wiki (n.d.).
Käskykoodien arvot ja niiden merkitykset.
Viitattu 25.2.2016
https://github.com/souliss/souliss/wiki/MaCaco%20Light-Data%20Protocol

Souliss Wiki (n.d.).
Yhdyskäytävän käskykoodit.
Viitattu 25.2.2016
https://github.com/souliss/souliss/wiki/MaCaco%20Light-Data%20Protocol

Souliss Wiki (n.d.).
Soulissin toimintojen tyypit.
Viitattu 25.2.2016
https://github.com/souliss/souliss/wiki/Typicals

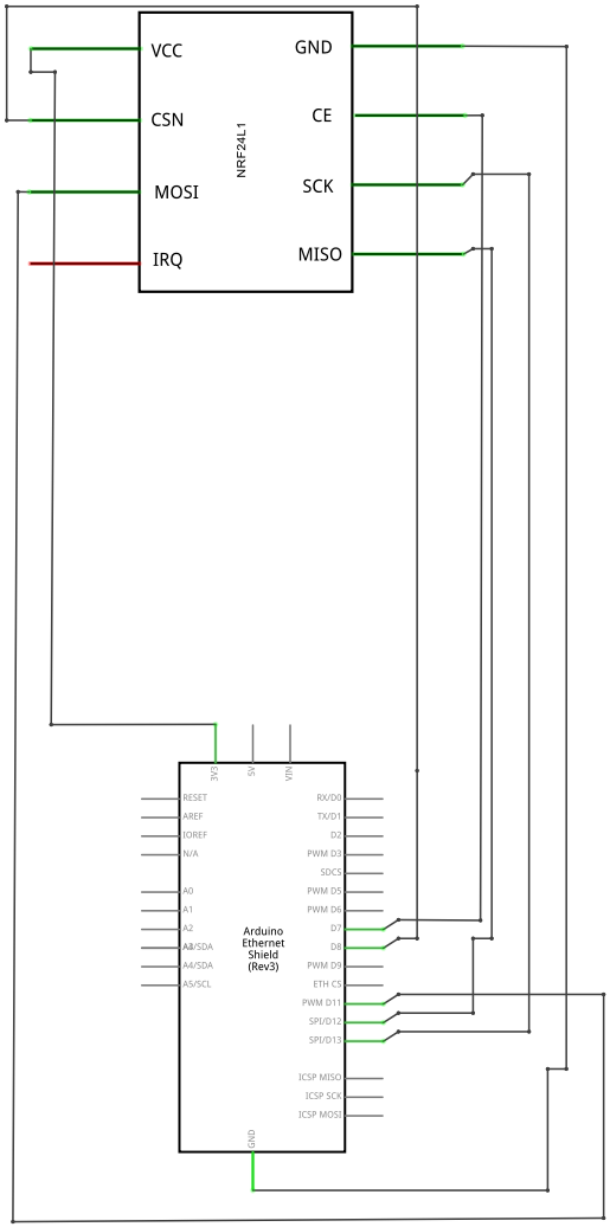
Liite 1

Laite	1kpl hinta	Löytyy koululta	Lisätietoja
Raspberry Pi	0.00 €	Kyllä	
Arduino	0.00 €	Kyllä	
Wiz820io	29.26 €	Ei	Halvempi kuin Arduino Ethernet Shield
Olimex ESP8266-EVB	9.95 €	Ei	Tarvitaan vähintään 1
FTDI-piiri	0.00 €	Voidaan ottaa Arduino UNO:sta prosessori irti	ESP8266 tarvitsee
5V/500mA DC virtalähde	0.00 €	Kyllä	ESP8266 tarvitsee
nRF24L01(+)	3.95 €	Ei	Tarvitaan vähintään 2
Kaikki komponentit	43.16 €		
Ilman ESP8266	33.21 €		
Ilman nRF24L01(+)	39.21 €		
Ilman ESP tai nRF24	29.26 €		

Liite 2

1. nRF24L01(+)-kokoonpano ja asetukset

Ensin tehdään kytkennät. Molempien Arduinojen oletetaan olevan UNO-malleja. Kytke CSN pinniin 8 ja CE pinniin 7. MOSI, MISO ja SCK ovat kytketty UNO:n tietojen mukaan pinneihin 11, 12 ja 13. Kytkentä tapahtuu samoin, oli Ethernet Shieldiä tai ei.



1.1. Yhdyskäytävän asetukset

```
#include "bconf/StandardArduino.h"
#include "conf/ethW5100.h" //Ethernet shield käytössä
#include "conf/nRF24L01.h"      //nRF24:n asetukset määri-
tetään kirjastoissa
#include "conf/Gateway.h"

#include <SPI.h>
#include "Souliss.h"

//IP-verkon asetukset. Voidaan myös käyttää dynaamista mää-
ritystä osoitteille.
uint8_t ip_address[4]  = {192, 168, 1, 77};
uint8_t subnet_mask[4] = {255, 255, 255, 0};
uint8_t ip_gateway[4]  = {192, 168, 1, 1};

//vNet-osoitteet
#define myvNet_address   ip_address[3]   //vNet-osoite sama
kuin IP-osoitteen viimeinen numero
#define myvNet_subnet 0xFF00 //vNet-verkon aliverkko
#define rf24_bridge 0x6501   //RF24-verkon osoite


void setup()
{
    //IP-verkon asetukset ja yhdyskäytäväksi ilmoittautumi-
nen
    SetIPAddress(ip_address, subnet_mask, ip_gateway);
    SetAsGateway(myvNet_address);

    //Omat osoiteasetukset. 0x0000 tarkoittaa, että solu on
kyseisen verkon supersolu.
    SetAddress(rf24_bridge, myvNet_subnet, 0x0000);

    SetAsPeerNode(0x6502, 1); //Toisen solun osoite ja numero
}
```

1.2. Solun asetukset

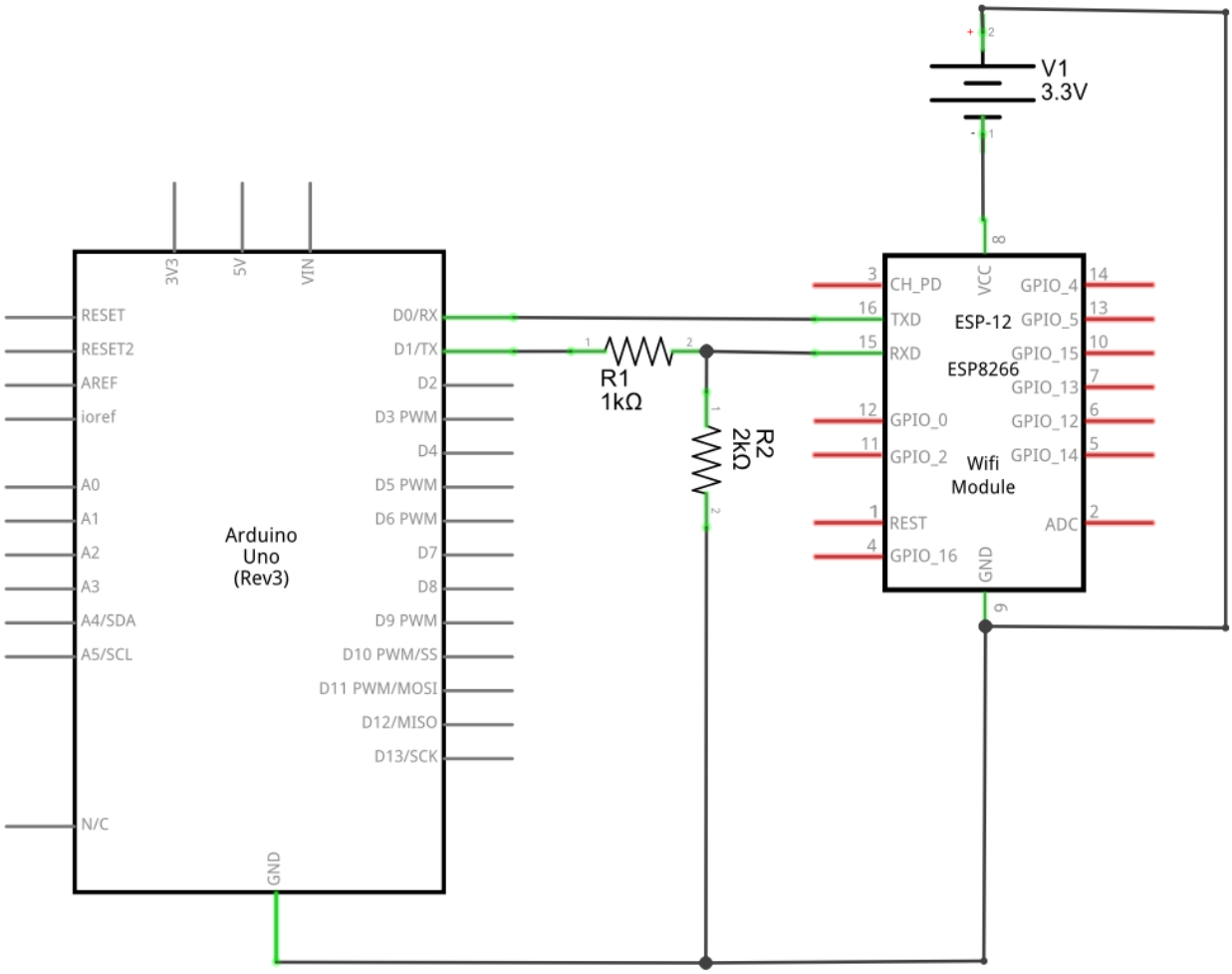
```
#include "bconf/StandardArduino.h"
#include "conf/nRF24L01.h"

#include <SPI.h>
#include "Souliss.h"


#define network_address     0x6502        //Oma vNet-osoite
#define network_my_subnet   0xFF00     //vNet-verkon alipeite
#define network_my_supern   0x6501    // Yhdyskäytävän osoite


void setup() {
//määritetään oma osoite
    SetAddress(network_address,    network_my_subnet,    net-
work_my_supern);
}
```

2. ESP8266-kokoonpano ja asetukset

ESP8266 käyttää 3.3V jännitettä sarjaliikennöintiin eikä kestä viittä volttia. Arduino käyttää 5V jännitettä omaan liikennöintiinsä, joten täytyy tehdä jännitteenjako tai käyttää regulaattoria. Arduinon 3.3V-syöttö ei ehkä riitä ESP:n käynnissäpitoon, joten kannattaa harkita 5V syötön käyttämistä jänniteregulaattorilla tai erillistä jännitelähdettä.

Kytkentä on yksinkertainen, jos ESP:n GPIO-pinnejä ei käytetä ohjauksiin. TX ja RX ristiinkytketään Arduinon kanssa. Jännitteenjako tehdään Arduinon TX-pinnin ja ESP:n RX-pinnin välille. 3.3V-lähde ESP:n Vcc-pinniin. Arduinon pitäisi tunnistaa 3.3V loogiseksi yhdeksi, joten toiseen datalinjaan ei tarvitse kytkeä mitään ylimääräistä. Vastuksen R2 jälkeen voidaan kytkeä diodi, jos halutaan varmistua, ettei häiriöitä pääse syntymään. Kytkennässä on esimerkkinä geneerinen ESP-12-moduuli, pinnien asettelu ja määritykset voivat vaihtua eri moduuleilla.

ESP8266 tarvitsee omat Arduino IDE -laitekirjastonsa ja USB to serial -adapterin, mutta ohjelmointi on muuten samankaltaista Arduinoon verrattuna.



2.1. ESP8266-asetukset siltaavana

```
Yhdyskäytävän osoiteasetukset:
SetAddress(0xAB01, 0xFF00, 0x0000);
SetAsPeerNode(0xD002, 1);


//ESP8266-asetukset alkavat tästä, voidaan muokata toimi-
maan itsenäisesti
#include "bconf/MCU_ESP8266.h"              //Alustana
ESP8266
#include "conf/SuperNode.h"                   // Supersolu
reititystä varten
#include "conf/usart.h"                      // USART
#include "conf/IPBroadcast.h"

// WiFi-asetukset
#define WIFICONF_INSKETCH
#define WiFi_SSID               "mywifi"
#define WiFi_Password           "mypassword"

#include <ESP8266WiFi.h>
#include <EEPROM.h>
#include "Souliss.h"

void setup()
{
    Initialize();
    GetIPAddress();
    SetAsGateway(myvNet_dhcp);

    //Määritä osoitteet. Alempi poistettava, jos Arduinoa
ei kytketä.
    SetAddress(0xAB02, 0xFF00, 0xAB01);
    SetAddress(0xD001, 0xFF00, 0x0000);
}

void loop()
{
    EXECUTEFAST(){
    UPDATEFAST();
    //Siltaus Arduinolle
    FAST_BridgeComms();
} }
```

2.2. Arduinon asetukset

```
#include "bconf/StandardArduino.h"
#include "conf/usart.h"

#include <SPI.h>
#include "Souliss.h"

void setup()
{
    Initialize();

    //Oma osoite ja supersolu
    SetAddress(0xD002, 0xFF00, 0xD001);
}

void loop()
{
EXECUTEFAST(){
UPDATEFAST();
FAST_PeerComms();
//Etsitään yhdyskäytävä ja yhdistetään
START_PeerJoin();
}

}
```

3. Dynaamiset osoitteet

Osoitteita voidaan myös määrittää automaattisesti. Toiminto käyttää paljon muistia yhdyskäytävällä. Vastaavia esimerkkejä löytyy Souliss-kirjaston esimerkeistä nimellä ZeroConf.

HUOM! openHABin IP-asetus yhdyskäytävälle on staattinen, joten jos yhdyskäytävän IP vaihtuu, putoaa openHAB Souliss-verkosta.

HUOM! Yhdyskäytävällä tärkeää ajaa ensin erase_ZeroConf, joka puhdistaa eepromin. Se löytyy esimerkkivalikosta, kun Souliss-kirjasto on asennettuna Arduino IDE:lle.

3.1. Yhdyskäytävän asetukset

```
#include "conf/Gateway.h"
#include "conf/Webhook.h"      //Käytä DHCP ja DNS
#include "conf/DynamicAddressing.h"         //Käytä dy-
naamisia vNet-osoitteita, käyttää muistia
#include "conf/IPBroadcast.h"               //IPBroadcast
soluille ilman staattista osoitetta

#include <SPI.h>
#include <EEPROM.h>
#include "Souliss.h"

void setup()
{
    Initialize();

    //Hae IP-osoite ja määritä yhdyskäytäväksi
    GetIPAddress();
    SetAsGateway(myvNet_dhcp);
    //Määritä vNet-osoitepalvelimeksi
    SetAddressingServer();
}
Void loop()
{
EXECUTEFAST() {
UPDATEFAST();
FAST_GatewayComms();
}
}
```

3.2. Solun asetukset

```
#include "conf/Webhook.h"
#include "conf/DynamicAddressing.h"
#include "conf/IPBroadcast.h"

#include <SPI.h>
#include <EEPROM.h>
#include "Souliss.h"

void setup()
{
Initialize();

GetIPAddress();

//Etsitään vNet-verkko
SetDynamicAddressing();
GetAddress();
}

void loop()
{

EXECUTEFAST() {
UPDATEFAST();

FAST_PeerComms();
}

EXECUTESLOW() {
UPDATESLOW();

//Kuunnellaan löytyykö yhdyskäytävää
SLOW_PeerJoin();
}
```