Saku Suppala

Mittaristo avoimen lähdekoodin kotiautomaatiojärjestelmien vertailuun

Tietotekniikan
Pro gradu -tutkielma
3. toukokuuta 2024

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Kokkolan yliopistokeskus Chydenius

Tekijä: Saku Suppala
Yhteystiedot: saku.s.suppala@student.jyu.fi
Puhelinnumero: 044-7877 252
Ohjaaja: Risto T. Honkanen
Työn nimi: Mittaristo avoimen lähdekoodin kotiautomaatiojärjestelmien vertailuun
in English: Metrics for open-source home automation system evaluation
Työ: Tietotekniikan Pro gradu -tutkielma
Sivumäärä: 80
Tiivistelmä: Älykotien määrän lisääntymisen esteenä on usein kuluttajien huoli yksityisyyden suojasta. Kaupallisten järjestelmien lisäksi saatavilla on useita avoimeen lähdekoodiin perustuvia kotiautomaatiojärjestelmiä. Tässä tutkielmassa kehitetään mittaristo, jonka avulla avoimen lähdekoodin kotiautomaatiojärjestelmät voidaan laittaa paremmuusjärjestykseen, kun kriteereinä on järjestelmän yhteisön kasvu ja aktiivisuus, kuluttajien tarpeet ja järjestelmän helppokäyttöisyys, sekä turvallisuus ja yksityisyyden suoja. Tutkimusmenetelmäksi on valittu konstruktiivinen tutkimusote, jossa ratkaisuna tutkimuspotentiaalia sisältävään ongelmaan luodaan konstruktio, joka pohjautuu yleiseen ja aihealueen syvälliseen tietämykseen.

Tutkielmassa käydään läpi ensin sekä kotiautomaation, että avoimen lähdekoodin ohjelmistojen teoriaa, ominaispiirteitä ja historiaa. Erillisessä luvussa perehdytään ohjelmistometriikkaan ja siihen, kuinka se on muuttunut ohjelmistojen koon ja kompleksisuuden mittaamisesta kohti arvosuuntautunutta mittausta. Teoriaosuuteen pohjaten luodaan mittaristo, joka konstruktiiviselle tutkimukselle ominaiseen tapaan testataan käyttäen otantana neljää avoimen lähdekoodin kotiautomaatiojärjestelmää. Testauksen otannasta löytyy mittariston avulla yksi järjestelmä, Home Assistant, joka vastaa erittäin hyvin kaikkia mittariston vaatimuksia.
Avainsanat: kotiautomaatio, avoin lähdekoodi, mittaristo, yksityisyys
Abstract: Privacy has been a major challenge in the context of smart homes and home automation. In addition to commercial systems there is a wide range of open-source home automation systems available. The aim of this thesis is to construct a set of metrics to evaluate open-source home automation systems based on the growth and activity of the community, consumer needs, ease of use, as well as security and privacy policy. A constructive research approach has been chosen as the research method, in which a construction is created as a solution to a problem with research potential. The construction is based on general and in-depth knowledge of the subject area.

The theory part of this thesis contains history and characteristics of home automation and open-source software development. In a separate chapter the development of software metrics from measuring the size and complexity of a software towards measuring the customer value is discussed. Based on the theory a set of metrics will be constructed and tested, as is characteristic of constructive research, using a sample of four open-source home automation systems. Based on the testing of these four systems we will find one system, Home Assistant, which meets all the requirements.
Keywords: home automation, open source, metric, privacy

Esipuhe

Tämän tutkielman valmistumisen myötä saan päätökseen opinnot, jotka muuttuivat pandemia-ajan ajanvietteestä mielenkiintoiseksi matkaksi kohti filosofian maisterin tutkintoa. Monimuoto-opiskelu perheellisenä aikuisopiskelijana vaati usein kompromisseja työn, perheen ja oman vapaa-ajan osalta. Tässä vaiheessa tahtoisin kiittää perhettäni ymmärryksestä iskän epäreilun suurta ruutuaikaa kohtaan.

Kokkolan yliopistokeskus Chydeniuksen opiskelujärjestelyt mahdollistivat tutkinnon suorittamisen käymättä kertaakaan Kokkolassa. Suorittamani opintojaksot olivat kuitenkin tasainen kattaus etäluentoja, nauhoitettuja luentoja, ohjelmointia ja kirjallisia tehtäviä. Haluankin kiittää Informaatioteknologian yksikön koko henkilökuntaa mielenkiintoisista ja opettavaisista opintojaksoista. Erityiskiitokset Risto Honkaselle, tämän tutkielman ohjaajalle, säännöllisistä ohjauspalavereista, jotka positiivisella tavalla potkivat minua eteenpäin kohti tätä hetkeä.

> "Live as if you were to die tomorrow. Learn as if you were to live forever."
> - Mahatma Gandhi (1869-1948)

Sanasto

ACBC	Adaptive Choice-Based Conjoint, adaptiivinen valintoihin perustuva conjoint-analyysi
ASM	Attack Surface Metric, hyökkäyspinta-alan mittausmenetelmä
BLE	Bluetooth Low Energy, lyhyenmatkan vähäenerginen WPAN tekniikka
CFR	Change Failure Rate, muutosvirheprosentti
CVE	Common Vulnerabilities and Exposures, haavoittuvuuksien ja paljastuneiden tietoturvaheikkouksien kirjausjärjestelmä
CVSS	Common Vulnerability Scoring System, standardi ohjelmistojen tietoturvahaavoittuvuuksien vakavuuden arvioimiseksi
DF	Deployment Frequency, päivitysten tiheys
ECHO IV	The Electronic Computing Home Operator, ensimmäinen kotiautomaatiojärjestelmä
FPA	Functional Point Analysis, toimintopistemalli
GDPR	General Data Protection Regulation, Euroopan unionin tietosuoja-asetus
HA OS	Home Assistant Operating System, Home Assistant käyttöjärjestelmä
IoT	Internet of Things, esineiden internet
IPv6	Internet Protocol version 6, suuremman IP-osoiteavaruuden mm. IoT-laitteiden tarpeeseen mahdollistava protokolla
LOC	Lines of Code, ohjelmakoodin rivien määrä
MAC	Media Access Control, verkkosovittimen yksilöivä osoite
Matter	Connectivity Standard Alliancen avoimen lähdekoodin standardi kotiautomaatioon ja IoT-laitteille
MAU	Monthly Active Users, kuukausittaiset aktiiviset käyttäjät
MAUT	Multi-Attribute Utility Theory, monitavoitteinen hyötyteoria

MLT	Mean Lead-Time for Changes, muutosten keskimääräinen läpimenoaika
MQTT	Message Queue Telemetry Transport, avoimen lähdekoodin julkaisija/tilaajaperustainen viestintäprotokolla
MTTR	Mean Time to Recover/Restore/Remediate, keskimääräinen palautumis-/palautus-/korjausaika
OpenBRR	Open Business Readiness Rating, avoimen lähdekoodin ohjelmistojen arviointimenetelmä
PbD	Privacy by Design, sisäänrakennettu tietosuoja
RASQ	Relative Attack Surface Quotient, suhteellinen hyökkäyspinta-osamäärä, mittari hyökkäyspinta-alan laskentaan
RFID	Radio Frequency Identification, radiotaajuinen etätunnistus
SSH	Secure Shell, protokolla salattuun tietoliikenteeseen
Thread	Thread Groupin IPv6-pohjainen IEEE 802.15.4-standardiin perustuva verkkoprotokolla IoT-laitteille
Wi-Fi	Wi-Fi Alliancen rekisteröity tavaramerkki IEEE 802.11-standardiin perustuvalle langattomalle teknologialle
WPAN	Wireless Personal Area Network, langaton lähiverkko
WSN	Wireless Sensor Network, langaton sensoriverkko
YAML	YAML Ain't Markup Language, konfiguraatiotiedostoissa käytetty merkintäkieli
ZHA	Zigbee Home Automation, Zigbee-laitteet kotiautomaatioon yhdistävä integraatio
Zigbee	Connectivity Standard Alliancen tavaramerkki IEEE 802.15.4-standardiin perustuvalle WPAN tekniikalle
Z-Wave	Z-Wave Alliancen tavaramerkki ja patentoima langaton kommunikaatioprotokolla

Sisällys

Esipuhe **i**

Sanasto **ii**

1 Johdanto **1**

2 Tutkimusmenetelmä **3**

3 Avoimen lähdekoodin kotiautomaatiojärjestelmät **6**
3.1 Kotiautomaatiojärjestelmä 6
3.1.1 Kotiautomaatiojärjestelmien historiaa 7
3.1.2 Esineiden internet 8
3.1.3 Nykyiset kotiautomaatiojärjestelmät 9
3.2 Avoin lähdekoodi 11
3.2.1 Avoimen lähdekoodin ohjelmistojen kehitys 11
3.2.2 Tietoturva ja yksityisyys 12
3.3 Avoimen lähdekoodin kotiautomaatiojärjestelmiä 13
3.3.1 Gladys Assistant 14
3.3.2 Home Assistant 14
3.3.3 Jeedom 15
3.3.4 OpenHAB 16

4 Ohjelmistometriikka **17**
4.1 Ohjelmistometriikan kehitysvaiheet 18
4.2 Metriikka ketterissä menetelmissä 19
4.3 Arvosuuntautuneet mittarit 21

5 Mittaristo avoimen lähdekoodin kotiautomaatiojärjestelmän valintaan **24**
5.1 Aikaisempi tutkimus 24
5.2 Mittausalue Y: Yhteisö 26
5.2.1 Mittari Y1: Kasvava yhteisö 27
5.2.2 Mittari Y2: Aktiivinen yhteisö 27

5.2.3 Mittari Y3: Projektin kiinnostavuus ja suosio (Stargazers) . . . 28
5.3 Mittausalue K: Käyttöarvo ja käytönaikainen laatu 28
5.3.1 Mittari K1: Käyttöarvo 32
5.3.2 Mittari K2: Käytönaikainen laatu 33
5.4 Mittausalue T: Tietoturva ja yksityisyys 34
5.4.1 Mittari T1: Tietoturva 37
5.4.2 Mittari T2: Yksityisyys 37

6 Testaus **39**
6.1 Konteksti ja käytetty laitteisto 39
6.2 Mittausalue Y 42
6.3 Mittausalue K 44
6.3.1 Gladys Assistant 44
6.3.2 Home Assistant 49
6.3.3 Jeedom 53
6.3.4 OpenHAB 58
6.4 Mittausalue T 63
6.5 Testitulosten yhteenveto 66

7 Yhteenveto ja johtopäätökset **68**

Lähteet **71**

1 Johdanto

Älykotien ja kotiautomaation yleistymistä on odotettu kauan. Statistan tutkimuksessa älykodiksi luokitellaan kotitaloudet, joissa mahdollisen älytelevision lisäksi jokin muu kodin laitteista on suoraan tai yhdyskäytävän kautta yhteydessä internetiin [99]. Älykodeiksi määritellään myös kodit, joissa on kotiautomaatioon osallistuvia laitteita, jotka kommunikoivat keskenään. Statistan ennustuksen mukaan Suomessa on noin kaksi miljoonaa älykotitaloutta vuonna 2028, mikä tarkoittaa lähes 200% kasvua vuodesta 2023, jolloin ennustus on julkaistu [99].

Älykotien lisääntymisen esteenä on usein ihmisten pelko omasta yksityisyyden suojasta sekä laitteiden tietoturvasta. Schusterin ja Habibourin [93] tutkimuksessa laitevalmistajiin ja palveluntarjoajiin luotti vain alle kolmannes vastaajista. Uutiset, joissa kerrotaan älytelevision ääniohjauksen lähettävän puhetta kolmannelle osapuolelle ilman suostumusta, lisäävät epäluottamusta [30].

Älykodit ja kotiautomaatio kuitenkin helpottavat elämää ja voivat auttaa säästämään energiaa. Schomakers et al. [92] tutkivat mitä kuluttajat odottavat kotiautomaatiojärjestelmältä. Heidän tutkimuksensa perusteella kuluttajat halusivat, että tiedot tallennetaan vain paikallisesti. Laitevalmistajien ja palvelutarjoajien konseptiin usein kuuluu pilvipalvelu, koska sen perusteella asiakkaalta voidaan veloittaa säännöllisesti kuukausi- tai vuosimaksuja.

Kotiautomaatioon on tarjolla myös useita avoimeen lähdekoodiin perustuvia järjestelmiä, joiden kehittämisestä vastaa yleensä järjestelmän käyttäjistä koostuva yhteisö. Kehitys alkaa yleensä pienen ydinryhmän projektina jossain versionhallintajärjestelmässä ja uusien jäsenten ja heidän tarpeidensa myötä järjestelmän toiminnallisuudet laajenevat. Tässä tutkielmassa pyritään vastaamaan seuraaviin kysymyksiin:

- Millaisella ohjelmistomittarilla voidaan laittaa avoimen lähdekoodin kotiautomaatiojärjestelmät paremmuusjärjestykseen, kun kriteereinä ovat:
    1. kuinka aktiivinen ja kasvava kehittäjien yhteisö on?
    2. kuinka hyvin se vastaa kuluttajien tarpeita ja onko se helppo käyttää?
    3. kuinka siinä on otettu huomioon turvallisuus ja yksityisyyden suoja?

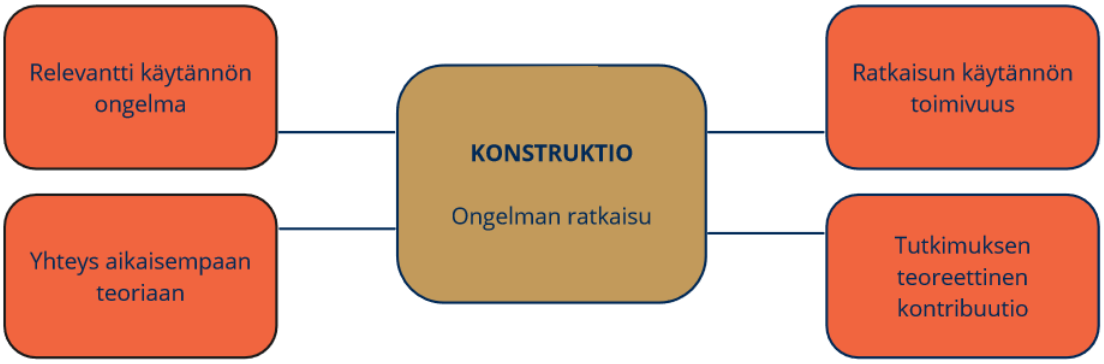
Vastauksia tutkimuskysymyksiin haetaan luomalla mittaristo avoimen lähdekoodin kotiautomaatiojärjestelmien vertailuun. Tutkielmassa käytetään konstruktiivista tutkimusmenetelmää, jossa ongelmaan pyritään löytämään ratkaisu teoriaan perustuvan ja testattavan konstruktion avulla. Mittariston testaaminen suoritetaan arvioimalla neljää avoimen lähdekoodin kotiautomaatiojärjestelmää, joiden joukosta löytyy mittariston avulla yksi selkeästi kaikki odotukset ja vaatimukset täyttävä järjestelmä. Testattavista järjestelmistä Home Assistant saa 4,9 pistettä. Testauksessa seuraavaksi tulleet Gladys Assistant (3,6 pistettä) ja OpenHAB (3,4 pistettä) jäävät selvästi taakse. Jeedom, neljäs testatuista järjestelmistä saa 2,6 pistettä.

Tässä pro gradu -tutkielmassa on seitsemän lukua. Tämän johdannon jälkeen luvussa 2 esitellään tutkielmassa käytetty tutkimusmenetelmä, jonka jälkeen siirrytään tutkimusaiheeseen liittyvään teoriaosuuteen. Luvussa 3 syvennytään ensin kotiautomaatiojärjestelmiin ja avoimen lähdekoodin erityispiirteisiin. Ohjelmistometriikkaan ja sen eri suuntauksiin perehdytään luvussa 4. Tutkielmassa luotava konstruktio eli mittaristo esitellään luvussa 5 ja se testataan luvussa 6. Lopuksi luvussa 7, yhteenveto ja johtopäätökset, pohditaan mittariston kytkeytymistä esitettyyn teoriaan ja sen tuottamaa arvoa.

2 Tutkimusmenetelmä

Tämä tutkielma on toteutettu käyttäen menetelmänä konstruktiivista tutkimusotetta. Konstruktiivinen tutkimusote kehitettiin alun perin liiketaloustieteiden käyttöön, mutta vastaavanlaisia tutkimusotteita käytetään myös useilla muilla tieteenaloilla. Tutkimusmenetelmän ydinpiirteet on esitetty kuvassa 2.1.

Lukan [65] mukaan konstruktiivinen tutkimusote tukeutuu olemassa olevaan teoriaan ja pyrkii ratkaisemaan merkittäviä käytännön ongelmia tuottaen innovatiivisia konstruktioita, ja näin tuottamaan uutta teoreettista tietoa. Konstruktioiksi Lukka lukee kaikki ihmisen luomat artefaktit, kuten mallit, kaaviot, suunnitelmat, organisaatiorakenteet, toimintastrategiat, kaupalliset tuotteet ja tietojärjestelmämallit, jotka eivät ole löydettyjä, vaan ne keksitään ja kehitetään.



Kuva 2.1: Konstruktiivisen tutkimusotteen ydinpiirteet (mukaillen Lukka [65])

Tietojärjestelmätieteissä paljon käytetty suunnittelututkimus, jossa myös pyritään ratkaisemaan käytännön ongelmia, on hyvin samantyyppinen konstruktiivisen tutkimusotteen kanssa. Suunnittelututkimuksessa puhutaan konstruktioiden sijaan artefakteista. Piirainen ja Gonzales [81] vertailivat suunnittelututkimusta ja konstruktiivista tutkimusotetta ja heidän mukaansa suunnittelututkimuksessa painotetaan yhteyttä olemassa oleviin ydinteorioihin, kun taas konstruktiivisessa tutkimusotteessa voidaan ehdottaa luovaakin teoriaan kytkettyä ratkaisua. Toinen selkeä ero heidän mielestään näiden kahden tutkimusmenetelmän välillä on se, että suunnittelututkimuksessa ei välttämättä tarvita ilmentymää, kun taas konstruktii-

visessa tutkimusotteessa painotetaan konstruktion relevanttiutta, soveltuvuutta ja ratkaisun toimivuuden testaamista käytännössä, eli konstruktiivinen tutkimusote sisältää aina konstruktion toteuttamisyrityksen [81]. Kasanen et al. [60] jakavat konstruktiivisen tutkimusotteen tutkimusprosessin kuuteen hyvin selkeään vaiheeseen:

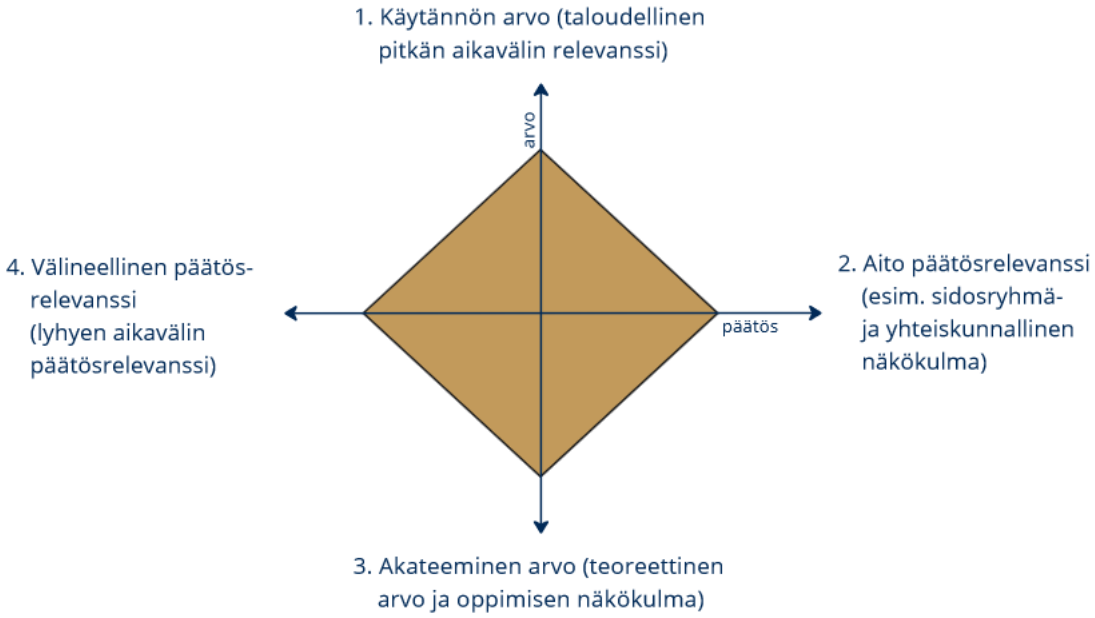
1. Merkittävän, tutkimuspotentiaalia sisältävän ongelman etsiminen
2. Yleisen ja syvällisen tietämyksen hankkiminen aiheesta
3. Ratkaisun eli konstruktion kehittäminen ongelmaan
4. Ratkaisun testaaminen käytännössä
5. Ratkaisun teoreettisen kytkennän ja uutuusarvon osoittaminen
6. Ratkaisun soveltamisalan pohdinta.

Tässä tutkielmassa tutkimuspotentiaalia sisältävä ongelma on kuvattu johdannossa. Teoriakytkentä alkaa tämän luvun jälkeen alkavissa teorialuvuissa, jonka jälkeen siirrytään ongelman ratkaisevan konstruktion, ohjelmistojen vertailuun tarkoitetun mittariston kehittämiseen. Ratkaisun testaaminen käytännössä tapahtuu empiriaosuudessa, jossa avoimen lähdekoodin kotiautomaatiojärjestelmät asennetaan laboratorioympäristössä yhden piirilevyn tietokoneelle. Lopuksi pohditaan erillisessä luvussa ratkaisun teoreettista kontribuutiota ja uutuusarvoa, sekä ratkaisun sovellusmahdollisuuksia. Soveltamisalan arvioinnissa Kasanen et al. puhuvat markkinatestistä, joka jaetaan kolmeen eri tasoon sen mukaan, kuinka laajasti konstruktio otetaan käyttöön:

- Heikko: vain kohdeorganisaatio on ottanut konstruktion käyttöön
- Keskivahva: konstruktio on otettu myös laajemmin käyttöön
- Vahva: konstruktiosta on todistetusti saatu taloudellista hyötyä.

Kasasen markkinatesti on hyvin liiketoimintalähtöinen. Siinä on lähtökohtaisesti olettamus siitä, että tutkimuksella on kohdeorganisaatio ja tavoitteena on kannattavuuden kasvu. Tutkimusprosessin aikana ei usein ole mahdollista tietää kuinka laajasti konstruktio otetaan käyttöön. Lukka [65] myös huomauttaa, että tutkijan ja kohdeyrityksen intressit saattavat mennä pahastikin ristiin, mikäli kohdeorganisaatio ei haluakaan julkaista tutkimusta arkaluontoisten tietojen paljastumisen pelossa.

Rautiainen et al. [84] ehdottivat markkinatestille vaihtoehtoista relevanssitestiä ja konstruoivat sen arviointiin työkaluksi diagrammin (kuva 2.2) nimeltä Relevance Diamond (relevanttiustimantti, oma suomennos), jossa merkityksellisyys voi olla päätöksentekoon liittyvää (engl., decision relevance) tai arvoperusteista (engl., value relevance). Relevanttiustimantissa arvoperustaisen vertikaalisen akselin yläpäässä on taloudellinen pitkän aikavälin merkitys, eli se mikä vastaa lähinnä Lukan esittämää markkinatestiä. Saman akselin toisessa päässä on akateeminen arvo. Akateemisen tutkimuksen kohdalla he ehdottivat, että heikko markkinatesti voitaisiin arvioida tutkimuksen alussa tai projektin aikana tutkimuksen mahdollisen potentiaalin perusteella. Keskivahvan markkinatestin saavuttaminen vaatisi tutkimuksen julkaisua ja vahvan markkinatestin kriteeristö täyttyisi, kun julkaisuun aletaan viittaamaan säännöllisesti. Timantin horisontaalisella akselilla reflektoidaan päätöksentekoon liittyvää relevanssia, joka voi vaihdella instrumentaalisesta lyhyen aikavälin päätösrelevanssista aina yhteiskunnallisesti merkittävään päätöksentekoon.



Kuva 2.2: Relevance diamond (mukaillen Rautiainen et al. [84])

3 Avoimen lähdekoodin kotiautomaatiojärjestelmät

Tässä luvussa taustoitetaan, mitä ovat avoimen lähdekoodin kotiautomaatiojärjestelmät. Ensimmäisessä alaluvussa 3.1 käydään läpi kotiautomaatiojärjestelmien kehitystä ensimmäisestä tunnetusta järjestelmästä aina nykyaikaisiin järjestelmiin asti. Toisen alaluvun 3.2 aluksi kerrotaan, miten määritellään avoimen lähdekoodin ohjelmistot ja miten niiden kehitys eroaa perinteisten kaupallisten ohjelmistojen kehityksestä. Sen jälkeen alaluvussa 3.3 esitellään neljä avoimen lähdekoodin kotiautomaatiojärjestelmää.

3.1 Kotiautomaatiojärjestelmä

Automaatiojärjestelmä voi olla yksittäinen laite, kuten liiketunnistuksen perusteella syttyvä valo, tai se voi olla kokonaista tehdasta ohjaava järjestelmä [105]. Lähes jokaisessa kodissa on automaatiota. Jotkut niistä voi perustua sensoriarvoihin, kuten ilmalämpöpumpun asettaminen kytkeytymään päälle sisälämpötilan noustessa yli asetetun raja-arvon tai hämäräkytkimen sytyttämä ulkovalo. Toisaalta automaatio voi perustua myös ohjelmointiin, kuten digiboksin ajastaminen tallentamaan lempisarja tai yksinkertaisimmillaan televisiolle annettu komento sammua kahden tunnin päästä.

Kotiautomaatiojärjestelmistä puhuttaessa käytetään myös termiä älykoti (engl., smart home), jonka sanastokeskuksen ylläpitämä TEPA-termipankki [89] määrittelee asukkaiden tarpeisiin suunnitelluksi, laitteita ja antureita yhdistäväksi tietoverkoksi, jota voidaan valvoa, käyttää tai ohjata etäältä. Tässä tutkielmassa käytetään nimitystä kotiautomaatiojärjestelmä, koska se kuvaa paremmin järjestelmän tarkoitusta ja toimintaa.

Ensimmäinen kotiautomaatiojärjestelmä ja ensimmäinen kotiautomaatioon liittyvä standardi esitellään alaluvussa 3.1.1. Seuraavassa alaluvussa 3.1.2 laitetaan esineiden internetin kehitys aikajanalle ja lopuksi alaluvussa 3.1.3 kerrotaan nykyaikaisten kotiautomaatiojärjestelmien ominaisuuksista, rakenteesta ja kommunikaatioratkaisuista.

3.1.1 Kotiautomaatiojärjestelmien historiaa

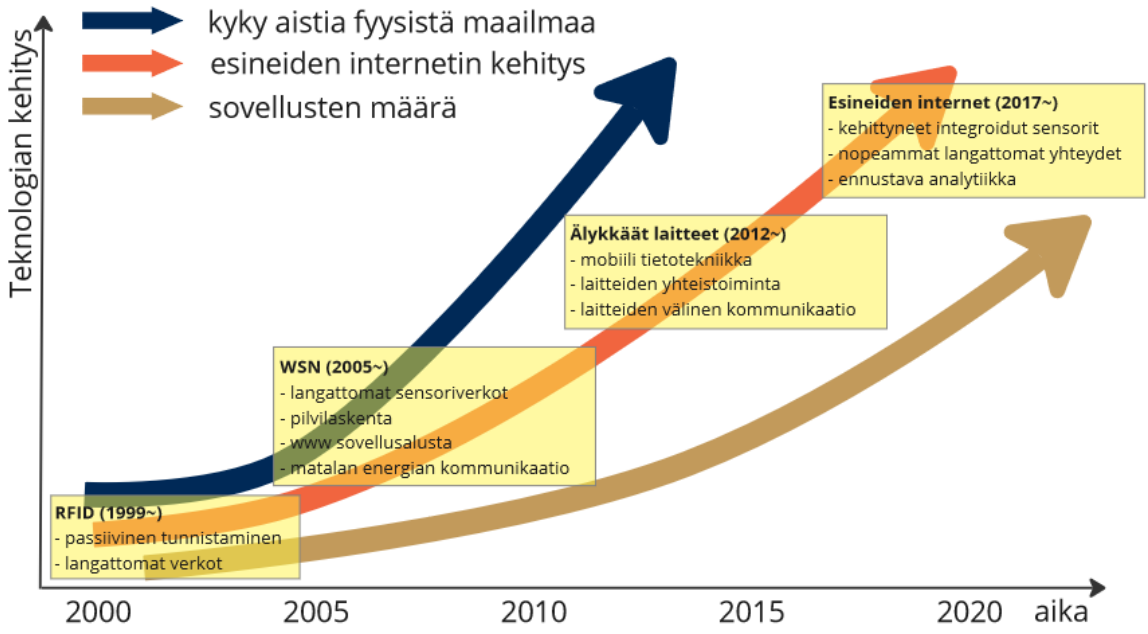
Vaikka kotiautomaatiojärjestelmiä saatetaan pitää tämän vuosituhannen keksintönä, on hyvä muistaa, että ensimmäinen kotiautomaatiojärjestelmä tehtiin jo vuonna 1966 [98]. Pittsburghissa asunut Westinghouse Electric Corporationin insinööri Jim Sutherland rakensi kotiinsa tietokoneen työpaikaltaan saamista osista, jotka puretiin Prodiac IV -nimisestä teollisen prosessin ohjauskoneesta. Sutherlandien kellariin asennettu tietokone, joka nykyisin sijaitsee tietokonehistorian museossa Kaliforniassa, sai nimen ECHO IV - The Electronic Computing Home Operator. Sen avulla Sutherlandit pystyivät huolehtimaan talouden kirjanpidosta ja kontrolloimaan muun muassa televisiota ja herätyskelloja [98]. Vuonna 1968 julkaistussa Popular Mechanics -lehden artikkelissa [49] kerrotaan Sutherlandien suunnitelmista saada ECHO IV kontrolloimaan ilmastointia säätilan perusteella ja laatimaan ostoslista keittiön kaapeista puuttuvien ruokatarvikkeiden mukaan.

Sutherlandin vaimo Ruth, joka oli kotitalouden opettaja, piti vuonna 1967 esitelmän Teksasissa ja kysyi konferenssiin osallistuneilta opettajilta mitä he toivoisivat kotitaloudessa olevalta tietokoneelta. Vastausten joukossa oli muun muassa lämpötilan, ilmankosteuden ja varashälyttimen kontrollointi, television ja valojen sammutus, ikkunoiden ja ovien avaaminen ja lukitseminen sekä automatisoitu nurmikon kastelu [98]. Moniin Sutherlandien visioimiin toimintoihin on nykyään saatavilla valmiita kaupallisia ratkaisuja, mutta yli 300 kiloinen ja lähes kaksi neliömetriä vievä ECHO IV ei koskaan edennyt kaupalliseksi tuotteeksi.

Vuonna 1975 Pico Electronics julkaisi X10 standardin, jota pidetään ensimmäisenä modernina kotiautomaatioteknologiana [14]. Sama teknologia on yhä käytössä, vaikka siinä on useita heikkouksia, kuten heikko häiriönsietokyky, sitä tukevien laitteiden vähyys sekä signaalikomentojen vähyys ja signaalin hitaus. X10 standardi perustuu signaalin lähetykseen rakennuksen sähköjohtoverkon välityksellä käyttäen lyhyitä radiotaajuuspurskeita. Yleisimpiä käyttötapauksia olivat valojen sytytys ja sammutus ja autotallin oven avaaminen ja sulkeminen. Vuosituhannen vaihteessa langattomien Wi-Fi-verkkojen myötä mahdollisuudet kasvoivat, mutta kotiautomaatiojärjestelmät eivät vieläkään yleistyneet. Vasta esineiden internetin (engl., Internet of Things, IoT) myötä kotiautomaatio alkoi vähitellen yleistymään.

3.1.2 Esineiden internet

Esineiden internet -termiä käytti ensimmäisen kerran Kevin Ashton työnantajalleen vuonna 1999 pitämänsä esityksen otsikkona [7]. Esityksen pääviestissä hän halusi korostaa, että internetin sisällön tuottaminen perustui lähes täysin ihmisen toimintaan, joko suoraan (kirjoittamalla) tai välillisesti (esimerkiksi skannaamalla viivakoodeja). Hänen ensimmäisessä viittauksessaan esineiden internetissä objektit olivat yksilöllisesti tunnistettavia ja radiotaajuisen tunnistustekniikan (engl,. radio frequency identification, RFID) avulla yhdistettyjä [64]. Vuosituhannen alussa RFID perustainen tunnistus lisääntyi etenkin logistiikassa, vähittäiskaupassa ja lääketeollisuudessa [64]. Kuvassa 3.1 on laitettu esineiden internetin teknologian kehitysvaiheita aikajanalle.



Kuva 3.1: Esineiden internetin kehitys (mukaillen Li et al. [64])

RFID:n jälkeen seuraava virstanpylväs teknologian kehityksessä oli langattomien sensoriverkkojen (engl., Wireless Sensor Network, WSN) kehityskulun alkaminen. Puhuttaessa langattomista sensoriverkoista tarkoitetaan usein laitteita, jotka eivät välttämättä ole yhteydessä internetiin ollenkaan, vaan paino on noodien eli laitteiden välisessä kommunikaatiossa, jonka mahdollisti etenkin langattomien verkkojen energiatehokkuuden paraneminen. Samoihin aikoihin ajoittuu ensimmäi-

set maininnat pilvilaskennasta. Myös World Wide Web kehittyi asiakaslähtöisempään suuntaan ja sitä alettiin käyttämään myös sovellusalustana (ns. Web 2.0 kehitysvaihe). Molemmat näistä kehitysvaiheista ovat luoneet perustan nykyaikaisille pilvipalveluille.

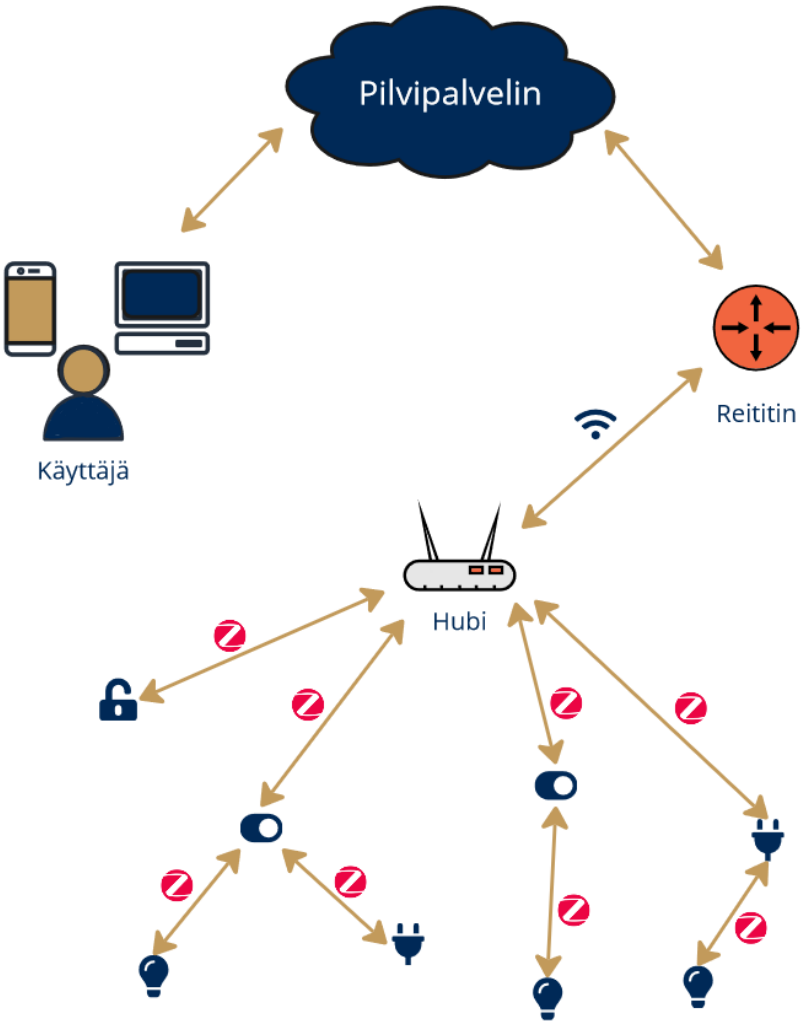
Viime vuosikymmenellä esineiden internetin kehitys otti isoja harppauksia. Laitteistojen ja mobiiliverkkojen samanaikainen kehitys mahdollisti älykkäiden laitteiden välisen kommunikaation ja yhteistoiminnan. Laitteisiin integroidut kehittyneet sensorit, nopeat langattomat yhteydet ja ennustava analytiikka ovat osa nykyaikaisia esineiden internetin ratkaisuja. Tämänhetkinen esineiden internetin tutkimus liittyy suurelta osin reunalaskentaan, kuudennen sukupolven mobiiliverkkojen ja lohkoketjujen tuomiin mahdollisuuksiin sekä koneoppimisen ja tekoälyn käyttöön [97].

3.1.3 Nykyiset kotiautomaatiojärjestelmät

Ashtonin ajatus esineiden internetistä perustui laitteiden kykyyn luoda dataa [7]. Laitteita, jotka havaitsevat tapahtuman tai muutoksen sen fyysisessä ympäristössä, kutsutaan esineiden internetin kontekstissa antureiksi tai sensoreiksi. Niiden avulla pystytään tekemään samankaltaisia havaintoja kuin ihmiset pystyvät tekemään näkö-, kuulo-, haju-, maku- ja tuntoaisteillaan. Kotiautomaatiojärjestelmissä paljon käytettyjä sensoreita ovat esimerkiksi liiketunnistin tai lämpötilasensori. Laitetta, joka esimerkiksi käynnistyy ohjelmoidun automaation tai sensoridatan perusteella, kutsutaan toimilaitteeksi eli aktuaattoriksi. Kotiautomaatiojärjestelmissä se voi olla esimerkiksi valaisin, sähköpistoke, ilmastointilaite tai elektroninen lukko. Nykyaikaiset kotiautomaatiojärjestelmät koostuvat edellä mainitun kaltaisista ohjelmoitavista esineiden internetin laitteista, joita voidaan tarvittaessa myös ohjata esimerkiksi etänä älypuhelimen sovelluksen avulla, kotona erilaisten sensorien avulla tai kirjautumalla sovellukseen kotiverkosta käsin.

Kotiautomaatiojärjestelmät vaativat yleensä sovelluksen tai alustan, johon laitteet ovat yhteydessä ja jossa automaatiot voidaan ohjelmoida. Usein kaupalliset järjestelmät vaativat oman erillisen keskittimen eli hubin, joka liitetään kodin langattomaan Wi-Fi-verkkoon tai suoraan reitittimeen Ethernet-kaapelilla. Tunnetuimpia kaupallisten järjestelmien valmistajia ovat Amazon, Aeotec, Apple, Google ja Hubitat. Niiden lisäksi useilla eri laitevalmistajilla ja jälleenmyyjillä, kuten Philips (Hue), IKEA (Trådfri) ja Lidl (Silvercrest Smart Home), on tarjolla oma keskittimensä. Wi-Fi-verkkoa käyttävien laitteiden ohjaamiseen riittää yleensä pelkkä mobiilisovellus.

Keskittimien ja laitteiden välisiä langattomia teknologioita on useita, joista tunnetuimpia ja suosituimpia ovat Wi-Fi-verkkojen lisäksi Bluetooth Low Energy (BLE), Zigbee ja Z-Wave. Näille verkoille on tyypillistä se, että ne ovat mesh- eli solmuverkkoja. Ne ovat itsemuodostuvia ja itsekorjautuvia ja laitteet voivat myös huolehtia datan reitittämisestä muille laitteille, kuten kuvassa 3.2 on esitetty.



Kuva 3.2: Esimerkki Zigbeetä käyttävän älykodin verkosta

Connectivity Standard Alliance, joka aikaisemmin tunnettiin nimellä Zigbee Alliance, julkaisi vuoden 2022 lokakuussa Matter-standardin, jonka toivotaan mahdollistavan eri valmistajien laitteiden yhteensopivuuden [12]. Matter-standardi käyttää IPv6-pohjaista Thread-verkkoprotokollaa, jonka ensimmäinen versio julkaistiin vuonna 2014 [103]. Thread-pohjaiset laitteet muodostavat itsekorjautuvan laiteverkon ja voivat keskustella toistensa kanssa keskenään myös ilman keskitintä.

3.2 Avoin lähdekoodi

Richard Stallman oli vuoteen 1984 asti töissä kuuluisassa MIT (Massachusetts Institute of Technology) yliopistossa. Hän oli jo vuosia tuskaillut kaupallisten tuotteiden ja niiden ohjelmistojen, kuten tulostimen ajureiden, toiminnallisuuteen ja koodiin, mutta hän ei voinut tehdä asialle mitään, sillä ohjelmistojen koodi ei ollut avoimesti saatavilla [11]. Yliopiston ottaessa käyttöön uuden kaupallisen käyttöjärjestelmän, hän erosi virastaan, jotta voisi olla varma siitä, että hänen ja ohjelmistoyhteisönsä kehittämä käyttöjärjestelmä GNU tulisi aina olemaan vapaa ohjelma. Hieman myöhemmin, lokakuussa 1985, Stallman perusti Free Software Foundationin tukemaan GNU:n kehitystä.

Ensimmäistä kertaa termiä "avoin lähdekoodi" käytti vuoden 1997 ja 1998 vaihteessa Foresight Instituutin Christine Peterson, koska termi "free software" koettiin harhaanjohtavaksi [11]. Avoin lähdekoodi tarkoittaa, että ohjelma on jaossa ja käytettävissä vapaasti ilman lisenssimaksua, ja sen lähdekoodi on kaikkien saatavilla ja muokattavissa omiin tarpeisiin [36]. Richard Stallman ei koskaan ole tunnustautunut osaksi avoimen lähdekoodin liikettä. Stallmanin kuuluisassa sanonnassa englanninkielistä sanaa "free" ei pitänyt yhdistää ilmaiseen vaan vapauteen ("I'm referring to freedom, not price. So think of free speech, not free beer") [11].

Tämän luvun ensimmäisessä alaluvussa 3.2.1 kerrotaan tarkemmin avoimen lähdekoodin ohjelmistojen kehityksestä, johon liittyy kiinteästi hakkerikulttuuri [11]. Seuraavassa alaluvussa 3.2.2 tarkastellaan avoimen lähdekoodin ohjelmistojen turvallisuutta ja yksityisyyden suojaa verrattuna kaupallisiin ohjelmistoihin.

3.2.1 Avoimen lähdekoodin ohjelmistojen kehitys

Hakkeri-termi viittaa alun perin taitavaan ja innokkaaseen tietokonealan harrastajaan tai ohjelmoijaan ja niin sanottu hakkerikulttuuri on luonut perustan nykyisille avoimen lähdekoodin järjestelmille [11]. Vuonna 1991 Stallman ja hänen ohjelmoijansa saivat GNU:n lähes valmiiksi, mutta kaikki osat yhteen sitova käyttöjärjestelmän ydin eli kernel puuttui. Samoihin aikoihin suomalainen Helsingin yliopiston opiskelija Linus Torvalds julkaisi Linux kernelin, johon hän integroi GNU työkalut. GNU-projektin oma kernel HURD ei ollut lähelläkään valmis Torvaldsin julkaistua Linux-kernelin. Stallman on ehdottanutkin, että pelkän Linuxin sijaan puhuttaisiin GNU-Linux-käyttöjärjestelmästä [11]. Stallmanin ja Torvaldsin kaltaisten hakkereiden innostuksen avulla on luotu paljon tänäkin päivänä käytettyjä ohjelmointikie-

liä ja ohjelmia, kuten vuonna 1987 julkaistu skriptimäinen ohjelmointikieli PERL, vuonna 1990 julkaistu Python ohjelmointikieli, vuonna 1994 julkaistu PHP ja vuonna 1995 julkaistu Apache palvelinohjelma.

Avoimen lähdekoodin ohjelmista on saatavilla yhteisön virallinen versio, jonka kehitykseen voi osallistua kuka tahansa [36]. Usein ohjelmiston käyttäjä päätyy kehittäjäksi silloin, kun huomaa avoimen lähdekoodin ohjelmassa virheen tai puutteen, jonka haluaisi korjata. Hansen et al. vertasivat sitä akateemiseen tutkimusmalliin, joka perustuu pitkälti hyväksyttyihin julkaisuihin. Pieni ydinryhmä on vastuussa kehityksestä ja sen laadun seurannasta. Yhteisöön kuuluvien kehittäjien maine kasvaa sen mukaan, mitä aktiivisemmin jäsen osallistuu kehitykseen. Kehitystyö tapahtuu pääsääntöisesti hajautettua versionhallintaa tukevilla Git-alustoilla [36].

Esineiden internetin sovellusten kehittäminen on haastavaa, koska niiden kehityksessä vaaditaan osaamista ohjelmiston lisäksi myös erilaisista laitteistoista, sensoreista ja aktuaattoreista [79]. Koska osaamista vaaditaan monelta alalta, kehittäjien määrä usein kasvaa. Kehittäjillä saattaa olla myös hyvin erilainen näkökanta esimerkiksi tietoturvan ja yksityisyyden suhteen.

3.2.2 Tietoturva ja yksityisyys

Avoimen lähdekoodin etuja turvallisuuden ja yksityisyyden kannalta on etenkin läpinäkyvyys ja luotettavuus [36]. Koska kuka tahansa voi tarkistaa avoimen lähdekoodin ohjelman koodin, tietoturvaan vaikuttavat ohjelmointivirheet löytyvät paljon helpommin verrattuna suljettuihin, omistusoikeuksien piiriin kuuluviin ohjelmistoihin, joissa vastaavat virheet paljastuvat usein vasta kun haavoittuvuutta aletaan käyttämään hyväksi. Richard Stallman sanoi, että suljettujen ohjelmistojen osalta on mahdotonta sanoa, onko ohjelmaan jätetty takaportti, jonka avulla voidaan tarkkailla ohjelman käyttöä käyttäjän tietämättä [11]. Juuri läpinäkyvyys luo edellytykset yksityisyyden suojalle, vaikka nykyään eri maiden asetukset ja lainsäädäntö voi asettaa ohjelmistovalmistajille vaatimuksia yksityisyyden suojaamiseksi. Esimerkiksi Euroopan unionin yleinen tietosuoja-asetus (General Data Protection Regulation, GDPR) sisältää 99 artiklaa ja 173 perustelukappaletta, joissa asetetaan vaatimukset henkilötietojen keräämistä, säilytystä ja hallinnointia varten.

Avoimen lähdekoodin ohjelmistojen kehittäjät ovat lähes poikkeuksetta järjestelmän käyttäjiä, jotka tunnetaan yhteisössä, mikä herättää luottamusta yhteisön sisällä. Yhteisö yleensä osallistuu laadunvarmistukseen testaamalla, mikä vähentää

kehittäjien painetta verrattuna markkinavetoisiin tuotteisiin. Hansen et al. kuitenkin mainitsevat, että ilman ammattimaista koodin tarkistusprosessia, ohjelmistoihin voi päätyä tietoturvaan ja yksityisyyteen vaikuttavia virheitä [36]. Päivitykset ja ohjelmavirheiden korjaus on usein nopeaa aktiivisissa yhteisöissä. Tämänlaisessa ketterässä ohjelmistokehityksessä on todettu koodin tarkistuksen lisäksi myös muita turvallisuuteen vaikuttavia pullonkauloja, kuten dokumenttien tuottaminen ja päivittyminen [85].

Yksityisyyden käsite jää epämääräiseksi, ellei sille anneta tiukkaa määritelmää [79]. Lakien, asetusten, ohjeistusten ja viitekehysten avulla voidaan auttaa kehittäjiä suunnittelemaan tietosuojatietoisia sovelluksia. Euroopan unionin yleinen tietosuoja-asetus GDPR on jo yksinään hyvin kattava [5]. Perera et. al näkevät kuitenkin tärkeimpänä sen, että ohjelmistokehittäjille kehittyisi yksityisyyttä koskeva ajattelutapa [79]. Mitä aikaisemmassa vaiheessa ohjelmistokehitystä tietoturvatoimenpiteet suoritetaan, sitä suurempi vaikutus niillä on [85].

3.3 Avoimen lähdekoodin kotiautomaatiojärjestelmiä

Tässä alaluvussa esitellään lyhyesti neljä avoimen lähdekoodin kotiautomaatiojärjestelmää. Valittuja järjestelmiä tullaan tarkastelemaan syvällisemmin luvussa 6, jolloin luvussa 5 luotua mittaristoa testataan asentamalla nämä järjestelmät Raspberry Pi yhden levyn tietokoneelle. Järjestelmien valinnassa kiinnitettiin huomiota erityisesti seuraaviin seikkoihin:

- Esiintyminen blogien ja artikkeleiden listauksissa
- Aika viimeisimmästä päivityksestä
- Asennusmahdollisuus Raspberry Pi:lle
- Integraatioiden ja yhteensopivuuksien määrä

Näissä esittelyissä pyritään eri lähteiden avulla selvittämään kunkin järjestelmän asennusten määrä ja löytämään tietoa järjestelmän kehityksestä vastaavasta yhtiöstä tai säätiöstä tai järjestelmän perustaja tai kehittäjäyhteisön vastuullinen jäsen. Lisäksi selvitetään avoimiin lähteisiin perustuen, onko kehityksestä vastaavilla jotain säännöllisiä tulonlähteitä projektista.

3.3.1 Gladys Assistant

Gladys Assistant on avoimen lähdekoodin kotiautomaatiojärjestelmä, joka mainostaa laittavansa yksityisyyden etusijalle. Ohjelmistoprojektin aloittanut ranskalainen Pierre-Gilles Leymarie julkaisi vuonna 2015 ensimmäisen kehittäjäyhteisölle tarkoitetun sivuston [63]. Gladys Assistant on toteutettu pääosin avoimeen lähdekoodiin perustuvan node.js JavaScript ajoympäristön avulla.

Leymarie alkoi kehittämään Gladys Assistantia vanhempiensa antamalle Raspberry Pi:lle vuonna 2013 [48]. Vuonna 2017 hän myi Starter Pack -pakettia, joka sisälsi videoita ja ohjeita järjestelmän käyttöön ja vuonna 2018 hän perusti maksullisen yhteisön järjestelmälle. Tällä hetkellä hän saa rahaa Gladys Assistant Plus-palvelusta, jonka avulla omaan instanssiin voi ottaa salatun etäyhteyden. Ohjelmiston sivuilla on julkaistu Plus-palvelun tilaajien määrän ja kuukausittain toistuva liikevaihto [28]. Tammikuussa 2024 Gladys Assistant toimi kotiautomaatiojärjestelmänä 711 kodissa ja Plus-palvelulla oli 90 tilaajaa ja kuukausittain toistuva liikevaihto oli 761 euroa. Leymarie ottaa vastaan myös lahjoituksia Buy Me a Coffee -palvelun kautta.

Gladys Assistantin asentaminen Raspberry Pi:lle on tehty helpoksi, sillä se löytyy uusimmasta Raspberry Pi Imager -ohjelmasta. Ohjelman sivusto ei listaa suurta määrää integraatioita, eli yhteensopivia laitteita, mutta integraatio Zigbee2MQTT -nimisen avoimen lähdekoodin ohjelman kanssa tarkoittaa, että sopivan Zigbee-yhdyskäytävän avulla Gladys Assistantin pitäisi pystyä kommunikoimaan yli 3 400 laitteen kanssa [109]. Kirjoitushetkellä viimeisin versio ohjelmasta on 4.36, joka on julkaistu 5.2.2024.

3.3.2 Home Assistant

Yksityisyys, paikallinen hallinta ja sopivuus Raspberry Pi:lle ovat vahvasti esillä Home Assistantin mainonnassa [41]. Hakipa millä hakukoneella tahansa listauksia avoimen lähdekoodin kotiautomaatiojärjestelmistä, Home Assistant on aina mainittu. Home Assistant Core on toteutettu Pythonilla. Itse käyttöjärjestelmä (engl., Home Assistant Operating System, HA OS) sisältää Coren lisäksi käyttöliittymän ja työkaluja, joiden avulla asennus esimerkiksi Raspberry Pi:lle onnistuu ilman erillisen käyttöjärjestelmän asentamista.

Home Assistantin asennusten määrä on kasvanut viimeisen 3 kuukauden aikana huimaa vauhtia: lokakuussa 2023 asennuksia oli noin 264 000 ja tammikuussa 2024

se oli jo yli 314 000 [42]. Home Assistantin voi asentaa Raspberry Pi Imagerin avulla ja se tarjoaa yli 2 600 integraatiota sisältäen lähes kaikki suurimmat valmistajat eri tuoteryhmistä. Viimeisin versio 2024.1.6 on julkaistu 31.1.2024.

Home Assistantin ensimmäiset kehittäjät ja ydintiimi perustivat vuonna 2018 Nabu Casa nimisen yhtiön, joka tarjoaa Home Assistant Cloud -pilvipalvelua. Tämä mahdollisti heille täysipäiväisen työskentelyn Home Assistantin parissa [71]. Nabu Casa on myös käyttänyt joukkorahoitusta laitteistojen, kuten Home Assistant Yellow, kehittämisessä [13]. Home Assistant Yellow on Raspberry Pi Compute Model 4:n päälle rakennettu valmis Home Assistant -ratkaisu [44]. Nabu Casa tarjoaa myös Home Assistantiin sopivaa SkyConnect Zigbee USB-tikkua, jonka voi tulevaisuudessa päivittää Thread -protokollan mukaiseksi tukemaan Matter-standardia [43]. Nabu Casan liikevaihto on 4 miljoonan dollarin luokkaa RocketReachin mukaan [86].

3.3.3 Jeedom

Jeedom on PHP ja Javascript -ohjelmointikielillä toteutettu kotiautomaatiojärjestelmä. Kuten Home Assistantin Yellow, Jeedom on myös saatavilla valmiina ratkaisuina nimeltä Jeedom Luna ja Jeedom Atlas. Jeedom Lunassa on sekä Zigbee, että Z-Wave-yhdyskäytävät. Jos järjestelmän haluaa ladata omalle Raspberry Pi:lle, tulee ensin ladata Raspberry Pi OS käyttäen Raspberry Pi Imageria ja sen jälkeen seurata sivuston ohjeita ohjelmiston lataamiseksi ja asentamiseksi [58]. Vaikka Jeedom on kehitetty avoimen lähdekoodin mukaan, useat moduulit ovat aina olleet maksullisia [33]. Jeedom Market -sivustolla oli vuoden helmikuussa 2024 Raspberry Pi:lle 551 moduulia, joista 337 oli ilmaisia. Maksullisten moduulien hinnat vaihtelivat kahdesta eurosta kymmeneen euroon.

Jeedomin kehitys alkoi vuonna 2013 ranskalaisen Loïc Gevreyn toimesta, mutta hän ei koskaan lähtenyt mukaan Jeedom SAS -nimiseen yhtiöön, joka perustettiin vuonna 2015 [34]. Hän on kuitenkin jatkanut projektissa aktiivisena kehittäjänä. Domadoo verkkokauppa, joka oli pitkään toiminut Jeedom tuotteiden myyjänä, osti vuoden 2023 syyskuussa Jeedom SAS -yhtiön [88], joka vuonna 2020 teki lähes puolen miljoonan liikevaihdolla noin 30 000 euron tuloksen [72]. Yrityskaupan alla yhtiö kertoi, että moduulien myynnistä saatava raha ei ole riittänyt infrastruktuurin ylläpitämiseen ja palkkojen maksuun ja ainoa ratkaisu on keskittyä enemmän ammattilais- ja yritysasiakkaisiin [34]. Yhteisössä on herännyt huoli siitä, että Jeedom muuttuu lopulta kokonaan kaupalliseksi järjestelmäksi. Jeedomin tiimin mu-

kaan ei ole syytä huoleen ja käyttäjien määrä on kasvussa. Jeedomin sivustolla olevan muutoslokin mukaan viimeisin versio Jeedomista on 4.4.1, jossa tärkein muutos on tuki PHP versiolle 8 [57].

3.3.4 OpenHAB

OpenHAB on avoimeen lähdekoodiin perustuva, pääosin Javalla koodattu teknologiariippumaton älykodin ohjausjärjestelmä, jonka kehityksen aloitti vuonna 2011 saksalainen Kai Kreuzer. OpenHAB on usein mukana samoissa vertailuissa, joihin on valittu myös Home Assistant. Hans-Jörg Merk, OpenHAB Foundation säätiön hallituksen jäsen on sanonut, että OpenHAB ei julkaise tai kerää, eikä tule julkaisemaan tai keräämään dataa käyttäjämääristä tai ohjelman latauksista [68]. Järjestelmän asentaminen Raspberry Pi:lle onnistuu Raspberry Pi Imagerin avulla ja viimeisin vakaa versio on 4.1.1, joka on julkaistu 7.1.2024.

OpenHAB puhuu integraatioiden sijaan sidoksista (engl., bindings), joita OpenHABilla on lähes 3 200 [74]. OpenHAB-sivustolla on myös linkki heidän myopenHAB nimiseen pilvipalveluun, jonka avulla omaa ohjausjärjestelmää on mahdollista käyttää myös verkon yli etänä. Ilmaisella pilvipalvelulla ei ole palvelutasosopimusta, ja sivuston mukaan sen tarkoitus on olla vain demo [76]. OpenHAB ottaa vastaan lahjoituksia, jotka ohjataan voittoa tavoittelemattomalle OpenHAB Foundation säätiölle, jonka jäseneksi yksityishenkilöt ja yritykset voivat liittyä kuukausimaksua vastaan [78].

4 Ohjelmistometriikka

"If you can't measure it, you can't improve it."
- Peter Drucker (1909-2005)

Ohjelmistomittareilla mitataan usein ohjelmistokehityksen kustannustehokkuutta, ohjelmiston monimutkaisuutta tai ohjelmiston laatua. Ohjelmistojen mittaamisen toivotaan tuottavan tietoa päätöksenteon tueksi ohjelmistojen kehityksessä [19].

Ohjelmistosuunnittelussa termiä "metriikka" on käytetty monella eri tapaa ja sillä on viitattu useisiin eri käsitteisiin, kuten mitattavaan määrään (mittaussuure), mittausmenettelyyn, mittaustuloksiin tai mittasuhteiden malleihin ja kohteiden mittaukseen [2]. Abran toteaa kirjassaan [2] sen olevan syy siihen, että tutkijoiden ja alan ammattilaisten hyväksymis- ja käyttöaste alan julkaisuihin on ollut alhainen.

Ohjelmistometriikka on kehittynyt eri ohjelmointikielien ja metodologioiden rinnalla jo 1960-luvulta lähtien. Tämän luvun ensimmäisessä alaluvussa 4.1 käydään läpi ohjelmistometriikan kehitysvaiheita aina 2000-luvulle asti, jolloin vielä pääosa ohjelmistokehityksestä perustui vesiputousmetodologian käyttöön [47].

Ketterät kehitysmenetelmät ja jatkuva ohjelmistokehittäminen ovat nopeuttaneet ohjelmistojen toimitusta. DevOps-nimi tulee englanninkielisistä sanoista *development* (suom. kehitys) ja *operations* (suom. tuotanto/toiminta). Sitä kuvaillaan prosessiksi tai kulttuuriksi, jossa yhdistetään kehitys- ja tuotantotoimintojen vuoropuhelua ja se on edustanut ohjelmistojen toimitussyklien kehitystä yli kahdenvuosikymmenen ajan nopeuttamalla toimitusta automatisoimalla ja integroimalla kehitys- ja toimintatiimien työtä [47]. Toisessa alaluvussa 4.2 tarkastellaan ohjelmistometriikkaa ketterien kehitysmenetelmien ja jatkuvan ohjelmistokehittämisen näkökulmasta.

Vaikka ohjelmistoa kehittävien yritysten tai sen kehityksestä vastaavien ohjelmoijien näkökulmasta moni mittari voi tuntua hyvin relevantilta, asiakkaan näkökulmasta se voi olla merkityksetön. Kolmannessa alaluvussa 4.3 tarkastellaankin ohjelmistojen ja niiden laadun mittausta asiakkaan kokeman arvon näkökulmasta.

4.1 Ohjelmistometriikan kehitysvaiheet

Koodirivien määrää (engl., Lines of Code, LOC) käytettiin ohjelman koon mittaamiseen jo 1960-luvun lopulla [18]. Sen avulla voidaan mitata ohjelmoijan tuotteliaisuutta (LOC per kuukausi) ja toisaalta myös ohjelmakoodin laatua suhteuttamalla ohjelmointivirheiden määrä tuhanteen riviin koodia (bugeja per KLOC). Vaikka LOC ei ota huomioon koodin kompleksisuutta, eikä se ole vertailukelpoinen eri ohjelmointikielien välillä, sen ja siitä johdettujen mittojen avulla voidaan mitata tehokkaasti ohjelmistokehityksen kustannustehokkuutta [18].

Ohjelman monimutkaisuuden mittaaminen yleistyi 1970-luvulla, jolloin McCabe ja Halstead julkaisivat omat kaavansa ohjelman kompleksisuuden mittaamiseen [18]. McCaben syklomaattinen kompleksisuus perustuu graafiteoreettiseen käsitteeseen ja siinä esiintyvä syklomaattinen luku ilmaisee lineaarisesti riippumattomien polkujen määrän ohjelmassa [17]. McCabe määritteli, että syklomaattisen luvun tulisi olla maksimissaan 10, muuten ohjelma tai moduuli on liian monimutkainen. Matemaattisesti voidaan tosin osoittaa, että syklomaattinen luku on yhtä suurempi kuin päättelyiden määrä ohjelmassa. Pelkkä syklomaattiseen lukuun luottaminen ei riitä, sillä on olemassa useita ohjelmia, joissa on paljon päättelyitä ja ne ovat silti helppoja ymmärtää, koodata ja ylläpitää [17].

Vuonna 1979 Allan Albrecht esitteli toimintopistemallin (engl., Functional Point Analysis, FPA), jonka avulla pyrittiin mittaamaan ohjelman kokoa käyttäjän näkökulmasta [2]. Toimintopisteiksi määriteltiin ulkoiset syötteet, tulosteet, kyselyt ja tiedostot, sekä sisäiset loogiset tiedostot [17]. Toimintopisteitä mitattaessa voi esiintyä subjektiivisuutta sen suhteen, mikä on mittaajan mielestä ohjelmiston kannalta relevantti piirre [37].

Ensimmäinen ohjelmistojen laadun mittaamiseen tarkoitettu standardi, ISO/IEC 9126 (Software engineering - Product quality), julkaistiin vuonna 1991. Se ja vuonna 1999 julkaistu ISO/IEC 14598 (Information technology - Software product evaluation) standardi on nykyään yhdistetty omaksi standardiperheekseen ISO/IEC 25000 (Systems and software engineering - Systems and software Quality Requirements and Evaluation), joka tunnetaan myös lyhenteellä SQuaRE [51].

Fenton ja Neil [18] totesivat vuonna 2000, että ohjelmistoteollisuuden mittaukset perustuivat silloin yhä pääosin vuoden 1970-luvun alun mittareihin. Heidän mielestään osasyynä oli se, että akateeminen tutkimus mittauksesta ei ollut vastannut teollisuuden tarpeita ja toisaalta tutkimus oli keskittynyt mittareihin, jotka soveltuivat vain pienille ohjelmille tai mittareilla mitattiin epäolennaisia ominaisuuksia.

4.2 Metriikka ketterissä menetelmissä

Ketterät ohjelmistokehitysmenetelmät, kuten Scrum, pyrkivät olemaan asiakaslähtöisiä, yksinkertaisia ja reagoimaan nopeasti muutoksiin palautteen ja trendien mukaan [61]. Ketteriin menetelmiin kuuluu mittareita, mutta ne poikkeavat perinteisistä mittareista. Kupiainen et al. [61] näkevät, että esimerkiksi perinteiseen kehittäjän tuottavuuteen painottuvan mittaamisen sijaan ketterässä ajattelutavassa pyritään mittareilla enemmänkin asettamaan tiimille sopivia työmääriä sprintin suunnittelun ohjaamiseksi, eikä niinkään tavoitteiden asettamiseksi. He tekivät systemaattisen kirjallisuuskatsauksen selvittääkseen ketterissä menetelmissä käytettäviä mittareita ja tutkivat niiden käytön motiiveja ja vaikutuksia. He löysivät hakukriteereillään 774 julkaisua, joista he valitsivat tutkimukseensa 30. Niiden yleisimmät viisi mittaria on esitetty taulukossa 4.1. Heidän tutkimuksensa mukaan nopeus, työmääräarvio ja asiakastyytyväisyys olivat tärkeimpiä mittareita, vaikkakin virheiden määrä oli asiakastyytyväisyyttä useammin mainittu.

Taulukko 4.1: Yleisimmät ketterien menetelmien mittarit [61], aineiston koko 30

Mittari	Tärkeys	Maininnat	Osuus
Nopeus	3	15	50%
Työmääräarvio	3	12	40%
Virheiden määrä	2	8	26%
Asiakastyytyväisyys	3	6	20%
Keskeneräisen työn määrä	1	6	20%

Parhaimmat DevOps-prosessit ja -kulttuurit pyrkivät yhdistämään ohjelmistokehityksen elinkaareen kehittäjien ja tuotantotoimintojen lisäksi muut sovellukseen sidoksissa olevat ryhmät, kuten esimerkiksi alustasta ja infrastruktuurista vastaavat suunnittelijat, tietoturva-asiantuntijat, hallinnon ja loppukäyttäjät [47]. Päivityksiä saatetaan julkaista jopa päivittäin ja näin pyritään vastaamaan ohjelmiston käyttäjien tarpeeseen saada uusia ominaisuuksia. Amaro et al. [6] julkaisivat vuonna 2023 laajan vertaisarvioidun systemaattisen kirjallisuuskatsauksen DevOps-mittareista. Heidän tutkimukseensa otettiin julkaistujen artikkeleiden lisäksi mukaan myös suppeaan käyttöön tarkoitettua kirjallisuutta (harmaa kirjallisuus) ja heidän alustava tietokantansa käsitti 1969 julkaisua, joista he valitsivat tutkimukseensa 114.

Taulukkoon 4.2 on poimittu heidän aineistostaan neljä yleisintä mittaria, joista jokainen mainittiin yli puolessa koko aineiston lähteistä. Selkeästi yleisin mittari oli keskimääräinen aika palautua virhetilanteesta. Toiseksi yleisin mittari oli muutosten keskimääräinen läpimenoaika ja kolmantena tuli päivitysten tiheys. Muutosvirheprosentti oli vasta neljänneksi yleisin mittari.

Taulukko 4.2: Yleisimmät DevOps-mittarit [6], aineiston koko 114

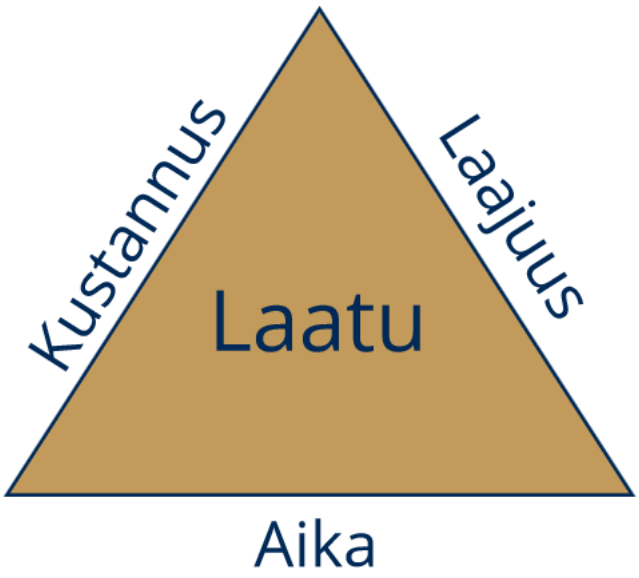
Mittari	Maininnat	Osuus
Keskimääräinen palautumis-/palautusaika (engl., Mean Time To Recover/Restore, MTTR)	96	84%
Muutosten keskimääräinen läpimenoaika (engl., Mean Lead-time for Changes, MLT)	91	80%
Päivitysten tiheys (engl., Deployment Frequency, DF)	88	77%
Muutosvirheprosentti (engl., Change Failure Rate, CFR)	72	63%

Bird et al. [8] tutkivat onko koodin omistajuuden ja koodin laadun välillä yhteyttä Microsoftin ohjelmistoprojekteissa ja he jakoivat kehittäjät kahteen ryhmään sen mukaan, kuinka paljon muutoksia he olivat tehneet ohjelmamoduuliin. Kehittäjän kontribuutio katsottiin vähäiseksi, jos hänen tekemänsä muutokset olivat alle 5% kaikista muutoksista ja vastaavasti kontribuutio oli merkittävä ja sitä kautta omistajuus vahvempi, jos kehittäjän panos vastasi yli 5% kaikista muutoksista. Mitä enemmän moduulilla oli kehittäjiä, joiden kontribuutio oli vähäistä, sitä suuremmalla todennäköisyydellä se sisälsi bugeja, eli ohjelmointivirheitä. Syyksi he arvioivat sen, että merkittävämmillä kehittäjillä on todennäköisesti enemmän ymmärrystä moduulin toiminnasta. Mitä suurempi ohjelmamoduulin kehitykseen eniten vaikuttaneen kehittäjän osuus oli kaikista kontribuutioista, sitä suuremmalla todennäköisyydellä moduulissa oli vähemmän virheitä [21].

Foucault et al. [20] tutkivat päteekö tämä avoimen lähdekoodin ohjelmistoissa, mutta he eivät päätyneet samaan tulokseen. Heidän hypoteesinsa oli, että avoimen lähdekoodin projektit ovat yleensä järjestäytyneet niin, että moduuleilla on usein ns. ydinkehittäjät ("heroes"), jotka vastaavat moduulin koodista ja siihen tehdyistä muutoksista [21].

4.3 Arvosuuntautuneet mittarit

Projektinhallinnan kolmio tunnetaan myös nimillä rautakolmio (engl., iron triangle) ja kolmirajoitteet (engl., triple constraints) [82]. Se on yli 50 vuotta vanha ja se on säilynyt enimmäkseen samanlaisena, vaikkakin sitä on aika ajoin kritisoitu voimakkaastikin. Pollack et al. [82] tarkastelivat artikkeleita 45 vuoden (1970-2015) ajalta ja totesivat, että aika ja kustannus ovat säilyttäneet paikkansa kolmiossa, mutta laadun tilalla on esiintynyt usein laajuus (engl., scope). Nykyään kolmio kuvataan usein niin, että laatu on kolmion keskellä, kuten kuvassa 4.1. Monet ohjelmistoja ja niiden kehitystyön onnistumista mittaavat mittaristot pohjautuvat näiden rajoitteiden mittaamiseen, mutta ketterän kehityksen projekteissa ne eivät riitä [59]. Kandengwa ja Khoza [59] kartoittivat Agile-metodologiaa käyttävien asiantuntijoiden mielipiteitä ja strategiset mittarit, jotka tuovat arvoa esiintyivät eniten vastauksissa. Yksi yleisimmistä vastauksista oli asiakastyytyväisyys, koska projektilla ei ole arvoa, jos sillä ei ole asiakkaita.

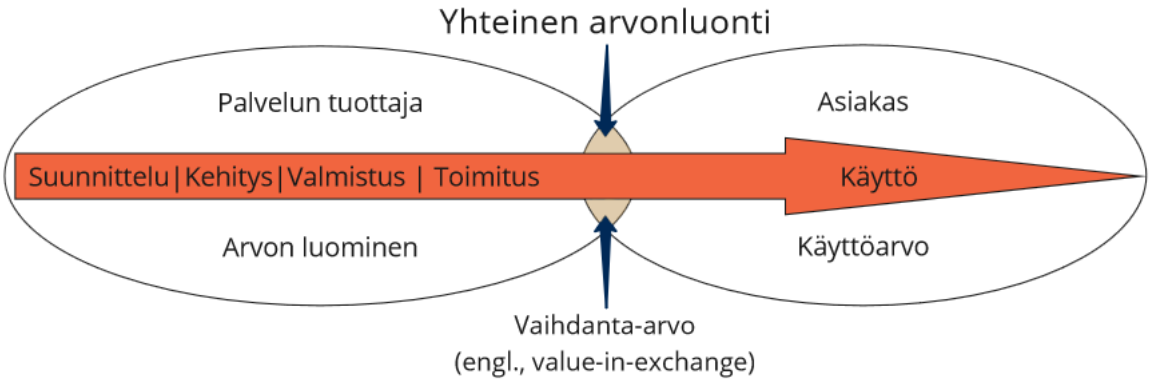


Kuva 4.1: Projektinhallinnan kolmio, rautakolmio, kolmirajoitteet

Tyrväinen et al. [104] toteavat, että ohjelmistonkehitysprosessia on hallittu perinteisesti kustannusperusteisesti mittaamalla ohjelmoijan ponnisteluja koodiriviä, toimintokohtaa tai vaatimusta kohden, kun taas nykyiset ohjelmistokehitysmenetelmät keskittyvät lean-ajattelun mukaisesti asiakasarvon luomiseen. He toteavat myös, että sovellusohjelmistojen ja pilvipalveluiden arvon mittaaminen on vaike-

aa ennen niiden käyttöönottoa, ja että nykyiset arvoteoriat eivät tarjoa yksiselitteistä tapaa arvioida asiakasarvoa. Ketterien menetelmien iteratiivinen lähestymistapa edistää vuorovaikutusta käyttäjien kanssa, koska sykliaika kehityksestä asiakaspalautteeseen lyhenee. Asiakaspalautteen avulla voidaan mitata ohjelmiston arvoa ja tuloksiin reagoiminen mahdollistaa yhä korkeamman arvon luomisen. Asiakkaan ohjelmistoista saamaa arvoa on vaikea mitata, koska se on subjektiivinen arvio, joka mitataan asiakkaan mielessä, ja siihen vaikuttaa etenkin se, mitä ohjelma tekee, eikä se mitä se on [39].

Grönroos ja Voima [32] määrittelivät arvonluomisen prosessina, jossa kaikki toimijat, niin asiakkaat kuin palveluntuottajat, luovat arvoa yhdessä (engl., value co-creation). Palvelusuuntautuneessa ajattelussa puhutaan nimenomaisesti palvelun käyttöarvosta, joka konkretisoituu palvelua kuluttaessa, mutta se täytyy luoda. Siinä välissä on jaettu alue (kts. kuva 4.2), jossa arvoa luodaan yhteisesti. Asiakkaan kokemaa arvoa ei voi määritellä etukäteen, vaan se muodostuu asiakkaan käyttäessä palvelua. Avoimen lähdekoodin ohjelmistojen osalta yhteinen arvonluonti konkretisoituu kehittäjien ja käyttäjien vuorovaikutuksessa esimerkiksi yhteisöjen keskustelualueilla.



Kuva 4.2: Yhteinen arvonluonti (mukaillen Grönroos ja Voima [32])

Arvopohjaisten mittareiden jaottelusta on useita esimerkkejä. Shanker [96] jaotteli vuonna 2012 avoimen lähdekoodin ohjelmistojen arvolupaukset viiteen tekijään: (1) toiminnallinen arvo, (2) kustannuksen/sijoituksen arvo, (3) toimittajasuhteen arvo (4) kehittäjäyhteisön tuoma arvo ja (5) brändin arvo. Haindl ja Plösch [35] tarkastelivat ohjelmistokehityksen laatumittareita laajemmasta näkökulmasta, sillä he tunnistivat, että monissa ryhmittelyissä keskityttiin usein vain yhteen kohderyhmään, joko kehittäjiin, liiketoimintaan tai asiakasarvoon. Heidän kyselytutkimuk-

seensa osallistui 41 erikokoisissa ohjelmistoalan yrityksissä työskentelevää asiantuntijaa. Heidän tutkimuksessaan mittarit ryhmiteltiin neljään ryhmään: (1) talous- ja markkinamittarit, (2) asiakastyytyväisyysmittarit, (3) arvonluontimittarit ja (4) arvolupausmittarit. Niistä ensimmäiseen ryhmään kuuluvia mittareita ovat esimerkiksi käyttäjien määrä, ohjelmiston tuotto tai markkinaosuus. Asiakastyytyväisyysindeksin lisäksi ohjelmiston arvostelut sovelluskaupoissa ja asiakaspysyvyys ja -vaihtuvuus olivat yleisimpiä mittareita toisessa ryhmässä. Arvonluomisen ja arvolupausten mittarit olivat molemmat jaoteltu neljään alaryhmään esimerkkimittareineen taulukon 4.3 mukaisesti.

Taulukko 4.3: Esimerkkejä arvomittareista [35]

Arvonluonti	Arvolupaukset
Jatkuva integrointi ja käyttöönotto + buildin kesto + käyttöönottojen määrä	Turvallisuus ja suorituskyky + vasteaika + haavoittuvuuksien määrä
Testauksen ja kehityksen tehokkuus + testien kattavuus + kehitysnopeus	Ominaisuuksien luotettavuus + saatavuus + keskimääräinen vikaväli
Resurssitehokkuus ja arkkitehtuuri + resurssien käyttö + ominaisuuksien monimutkaisuus	Käytettävyys + tehtävän valmistumisaika + käytettävyyshäiriöiden määrä
Ylläpidettävyys + tehtyjen muutosten määrä + tekninen velka	Toiminnallisuus ja sen parannus + toimitettujen tarinoiden määrä + ominaisuuskutsujen määrä päivässä

5 Mittaristo avoimen lähdekoodin kotiautomaatiojärjestelmän valintaan

Tässä tutkimuksessa luotava mittaristo on tarkoituksella tehty hyvin kevyeksi osittain tutkimuksen koon rajoittamiseksi ja toisaalta myös sen vuoksi, että eri järjestelmistä ja niiden kehittäjistä ja heidän taustallaan olevista yrityksistä ja yhteisöistä on vaikea saada vertailukelpoista tietoa. Kehitettävän mittariston avulla pyritään löytämään sellainen avoimen lähdekoodin kotiautomaatiojärjestelmä, jonka kehittäjien yhteisö on aktiivinen ja toimii ammattimaisesti ja joka parhaiten vastaa kuluttajien tarpeisiin ja mieltymyksiin, sekä ottaa myös huomioon kuluttajien huolenaiheet, kuten yksityisyydenturvan.

Avoimen lähdekoodin ohjelmistot eroavat kaupallisista ohjelmistoista monella tapaa. Usein avoimen lähdekoodin ohjelmistojen taustalla ei ole juridista yritystä, vaan ohjelmistoa kehitetään yhteisöllisesti. Ohjelmistojen käyttäjämääristä ja markkinaosuuksista on vaikea saada tarkkoja tietoja. Avoimen lähdekoodin ohjelmistojen kehittäjistä suurin osa on päätynyt kehittäjäksi käytettyään ohjelmistoa ensin itse [36]. Näistä syistä avoimen lähdekoodin ohjelmistojen osalta tarkastelemme kehittäjiä ja kuluttajia yhtenä kokonaisuutena, yhteisönä, jossa kehittäjät ja ohjelmiston käyttäjät luovat yhdessä arvoa.

Ennen mittariston esittelyä ensimmäisessä alaluvussa 5.1 tutustutaan lyhyesti aikaisempaan tutkimukseen aiheesta. Mittariston ensimmäinen mittausalue, yhteisö ja arvon luominen esitellään alaluvussa 5.2 ja alaluvussa 5.3 esitellään käyttöarvon ja käytönaikaisen arvon mittausalue. Lopuksi alaluvussa 5.4 mittaristoon lisätään vielä turvallisuutta ja yksityisyyttä koskeva mittausalue.

5.1 Aikaisempi tutkimus

Avoimen lähdekoodin ohjelmistojen laadun arvioimiseksi on kehitetty useita eri viitekehyksiä ja malleja [3, 4, 87, 90]. Vuonna 2022 Yilmaz ja Kolukisa Tarhan [106] totesivat, että monista malleista ja viitekehyksistä huolimatta, niiden käyttöönotto on ollut hyvin rajallista. He suorittivat systemaattisen kirjallisuuskatsauksen ja tarkastelivat 36 eri mallia tai viitekehystä vuosilta 2003-2020. Näistä valituista hie-

man yli puolet (58%) oli yleisluontoisia malleja avoimen lähdekoodin ohjelmistojen laadun arviointiin ilman viittausta mihinkään tiettyyn ominaisuuteen, ohjelmistoon tai käyttötarkoitukseen. Verrattuna tiettyyn tarkoitukseen suunniteltuihin malleihin, yleisluontoiset mallit ovat yleensä hyvin laajoja.

Roy et al. [87] kuvasivat menetelmän avoimen lähdekoodin ohjelmistojen laadun arviointiin IT-kouluttamattoman käyttäjän näkökulmasta. Heidän menetelmänsä on yhdistelmä ISO/IEC 25010:2010-standardin käytönaikaisen laatumallin osiosta, ohjelmistojen luotettavuuskäsitteistä sekä avoimen lähdekoodin arviointimenetelmästä OpenBRR. Heidän mallissaan arviointi tapahtuu viidessä vaiheessa:

1. Kontekstin määrittely
2. Käytönaikaisen laadun arviointi
3. Ohjelman luotettavuuden arviointi toiminnan oikeellisuuden perusteella
4. Ohjelmistokohtaisten laatuominaisuuksien arviointi ehdotetun mallin mukaan
5. Laadun kokonaisarviointi

Avoimen lähdekoodin kotiautomaatiojärjestelmiä käsitteleviä artikkeleita ja vertailuja löytyi vain muutamia. Dominiguez-Bolano et al. [15] tarkastelivat laajemmin eri IoT-alustojen, mukaan lukien avoimen lähdekoodin kotiautomaatiojärjestelmien, ominaisuuksia. Heidän katsauksessaan ei pisteytetty eri alustoja, vaan listattiin niiden lisenssi, pääkäyttötarkoitus, käytetty ohjelmointikieli, tuettujen integraatioiden määrä, tuetut tietokannat, menetelmät automaatioiden ohjelmointiin sekä mahdollisuudet analysoida ja visualisoida dataa.

Borissova et al. [10] käyttivät puolestaan monitavoitteista hyötyteoriaa (engl., Multi-Attribute Utility Theory, MAUT) laittaessaan avoimen lähdekoodin kotiautomaatiojärjestelmiä paremmuusjärjestykseen. He käyttivät kriteereinään järjestelmän asennuksen helppoutta, joustavuutta ja käyttöliittymää sekä käyttäjien yhteisöä, kehityksen vauhtia, integraatioiden määrää, tuettuja protokollia ja käytettyä ohjelmointikieltä. MAUT-pohjainen malli mahdollistaa eri kriteerien painottamisen, jolloin lopputulos riippuu valitsijan painotuksesta.

Setz et al. [94] julkaisivat vuonna 2021 kattavan vertailun avoimen lähdekoodin kotiautomaatiojärjestelmistä. Heidän tutkimuksessaan oli kolme osaa, joista ensimmäisessä kartoitettiin hakukoneiden avulla olemassa olevia avoimen lähdekoodin kotiautomaatiojärjestelmiä. He löysivät 20 eri järjestelmää, joista he valitsivat

tarkempaan tarkasteluun neljä. Näiden neljän valinnassa he käyttivät kriteereinä muutosten (engl., commits) määrää, viimeisimmän muutoksen ajankohtaa, ohjelman saamia arvosteluja (GitHub tähdet), dokumentaatiota ja edistäjien (engl., contributors) määrää.

Toisessa osassa näiden neljän järjestelmän ominaisuuksia tarkasteltiin tarkemmin 17 eri käyttötapauksen avulla. Käyttötapausten kuvaukset he valitsivat Abbas et al. [1] tutkimuksen aineistosta, joka sisälsi hyvin yksityiskohtaisia käyttötapauksia. Kaikkia käyttötapauksia ei testattu käytännössä, vaan järjestelmien kyky toteuttaa ne arvioitiin dokumentaatioon ja järjestelmän eri toiminallisuuksiin perustuen. Käyttötapauksista suurin osa liittyi kodin turvallisuuteen ja mukavuuteen.

Kolmannessa osassa he vielä vertasivat valittuja neljää järjestelmää 34 eri kriteeriin, kuten esimerkiksi ohjelmiston suosio, tuki, laitteistovaatimukset, dokumentoinnin laatu ja lisäosien (integraatioiden) lukumäärä. He myös vertailivat tarvittavien hiiren painallusten määrä tietyn automaation ohjelmoimiseksi. Heidän vertailussaan ei tarkasteltu yksityisyysasetuksia tai yksityisyyden suojaa missään osiossa.

5.2 Mittausalue Y: Yhteisö

Avoimen lähdekoodin kehityksessä aktiivinen ja laaja yhteisö on merkki siitä, että kehitystyölle on tekijöitä ja sen jatkuvuus on taattu. Kehittäjien lisäksi yhteisöön kuuluu käyttäjiä, jotka voivat toimia tukifoorumina tai ylläpitää dokumentaatiota. Yhteisön koko ei yksinään riitä, vaan yhteisön aktiivisuus on myös merkittävä tekijä [96]. Rekisteröityjen jäsenten määrän kasvulla voidaan arvioida järjestelmien kiinnostavuutta. Myös viestien ja uusien viestiketjujen määrä kertoo aktiivisuudesta, mutta niiden vertaileminen yhteisöpalstojen välillä voi aiheuttaa vääristymiä, koska eri palstoilla on eri keskustelualueita ja sääntöjä esimerkiksi myynti-ilmoitusten tai aiheen ulkopuolisten viestiketjujen suhteen.

Versionhallinta on usein hoidettu hajautetusti käyttäen Git-versionhallintajärjestelmää. GitHub on sivusto, jonne on tallennettu yli 100 miljoonaa ohjelmistokehitysprojektia, jotka käyttävät Git-versionhallintaa. GitHub-käyttäjät voivat antaa ohjelmistokehitysprojekteille tähden, jos kokevat sen kiinnostavaksi tai hyödylliseksi. Usein kehittäjät tekevät sen myös osoittaakseen arvostusta projektia kohtaan [9]. Yli neljännes kehittäjistä pitää tähtien määrää tärkeänä kriteerinä valitessaan projektia, jota voisivat alkaa edistämään.

5.2.1 Mittari Y1: Kasvava yhteisö

Tässä mittarissa yhteisön kasvua mitataan vertaamalla viimeisen 30 päivän aikana rekisteröityneiden käyttäjien määrää keskiarvoon, joka saadaan jakamalla yhteisön rekisteröityneiden käyttäjien määrä yhteisön iällä kuukausissa (mittari Y1, kaava 5.1). Tuloksena saatu suhdeluku kertoo, onko yhteisön kasvu keskiarvoa hitaampaa vai nopeampaa.

$$Y1 = \frac{\text{30 pv aikana rekisteröityneet}}{\text{yhteisön keskimääräinen kasvu kuukaudessa}} \tag{5.1}$$

Vertailtaessa eri avoimen lähdekoodin kotiautomaatiojärjestelmiä keskenään, tämän mittarin tuloksena saatu suhdeluku muutetaan pisteiksi seuraavasti:

$$pisteet(Y1) = \begin{cases} 5, \text{ kun } Y1 > 2,0 \\ 4, \text{ kun } 1,5 < Y1 \leq 2,0 \\ 3, \text{ kun } 1,0 < Y1 \leq 1,5 \\ 2, \text{ kun } 0,8 < Y1 \leq 1,0 \\ 1, \text{ kun } 0,5 < Y1 \leq 0,8 \\ 0, \text{ kun } Y1 \leq 0,5 \end{cases}$$

5.2.2 Mittari Y2: Aktiivinen yhteisö

Käyttäjien aktiivisuutta mitataan suhteuttamalla viimeisein 30 päivän aikana kirjautuneet käyttäjät (engl., Monthly Active Users, MAU) kaikkien rekisteröityneiden käyttäjien määrään (mittari Y2, kaava 5.2).

$$Y2 = \frac{\text{30 pv aikana kirjautuneet käyttäjät}}{\text{kaikki rekisteröityneet}} \cdot 100\% \tag{5.2}$$

Tuloksena saatu aktiivisuutta viimeisen 30 päivän aikana kuvaava prosenttiluku muutetaan pisteiksi seuraavasti:

$$pisteet(Y2) = \begin{cases} 5, \text{ kun } Y2 > 15,0\% \\ 4, \text{ kun } 12,0\% < Y2 \leq 15,0\% \\ 3, \text{ kun } 8,0\% < Y2 \leq 12,0\% \\ 2, \text{ kun } 5,0\% < Y2 \leq 8,0\% \\ 1, \text{ kun } 2,0\% < Y2 \leq 5,0\% \\ 0, \text{ kun } Y2 \leq 2,0\% \end{cases}$$

5.2.3 Mittari Y3: Projektin kiinnostavuus ja suosio (Stargazers)

Kehittäjien määrän sijaan tässä mittarissa keskitytään mittaamaan vain kehitysprojektin kiinnostusta Git-versionhallintajärjestelmän tähtien perusteella käyttäen Jarczyk et al. [53] luomaa Stargazers-mittaria, jota myös Setz et al. [94] käyttivät omassa vertailussaan. Myös tässä mittarissa pisteet annetaan suoraan Stargazers-kaavan 5.3 perusteella.

$$pisteet(Y3) = \log_{10}(x + 10) \tag{5.3}$$

5.3 Mittausalue K: Käyttöarvo ja käytönaikainen laatu

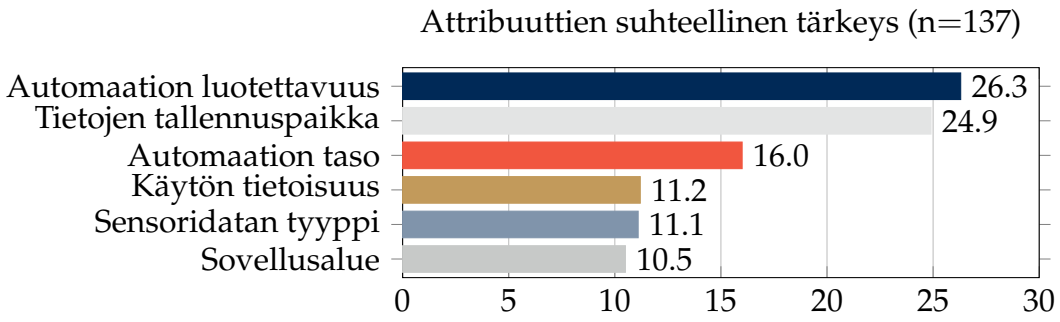
Ohjelmiston käyttöarvon määrittelevät sen käyttäjät. Kun tätä mittaristoa testataan seuraavassa luvussa 6, käyttöarvo määritellään Schomakers et al. [92] tutkimukseen pohjautuen. Tutkimuksessaan he käyttivät adaptiivista valintoihin perustuvaa conjoint-analyysia (engl., Adaptive Choice-Based Conjoint, ACBC), jonka avulla he selvittivät kotiautomaatiojärjestelmien tärkeimpien tuoteominaisuuksien merkitystä kuluttajille.

Conjoint-analyysi on tilastollisen analyysin muoto, jonka avulla voidaan tutkia muun muassa sitä, kuinka tärkeinä vastaajat pitävät tuotteiden tai palveluiden eri ominaisuuksia [83]. Vastaajien mieltymykset puretaan tuoteominaisuuksien osahyötyarvoiksi tai osittaishyödyiksi (engl., part-worth utilities). Vastaajalle voidaan esimerkiksi näyttää korttia, jossa tuotteella on kolme eri tuoteominaisuuksien kombinaatiota, joista vastaajan pitää valita mieluisin. Vastaajan valitsema kombinaatio saa yhden pisteen. Jotta näistä tiedoista saadaan vertailukelpoista dataa, käytetään lineaarista regressiota selvittämään kunkin tuoteominaisuuden tason suhteellista merkitystä sen ominaisuusryhmässä [62]. Näitä osahyötyarvoja tarkastelemalla voidaan selvittää minkä ominaisuuden muutos vaikuttaa selkeimmin kombinaation valintaan. Mitä suurempi ero suurimman ja pienimmän arvon välillä on, sitä suurempi on sen tuoteominaisuuden merkitys vastaajalle [62].

Tuoteominaisuuksien ja niiden eri tasojen määrän kasvaessa suureksi, kasvaa myös vaihtoehtojen kokonaislukumäärä niin suureksi, ettei niitä pysty käsittelemään kyselytutkimuksen avulla. Esimerkiksi Schomakers et al. tutkimuksessa on kuusi ominaisuutta, joista viidellä on neljä eri tasoa ja yhdellä on kolme tasoa, mikä tekee yhteensä 3 072 eri kombinaatioita, jolloin vaihtoehtoja täytyy karsia.

Kyselyohjelmistoja valmistava Sawtooth Software on erikoistunut conjoint-analyysiin ja kehittänyt conjoint-analyysiä yhdessä tutkijoiden kanssa [31]. Sawtooth Software kehitti jo 1980-luvun lopulla adaptiivisen conjoint-analyysin, jossa vastaajat laittavat ensin attribuutit tärkeysjärjestykseen ja vasta sen jälkeen heille esitetään etukäteen valittuja kombinaatioita [31]. Nykyään ACBC on Sawtooth Softwaren edistynein conjoint-analyysijärjestelmä, jossa yhdistyy valintaperusteinen conjoint-analyysi, tekoäly sekä dynaaminen kombinaatioiden muodostus, jonka avulla pyritään pitämään kyselyyn vastaaminen mielenkiintoisena [91].

Schomakers et al. [92] tekemään tutkimukseen osallistui 137 saksalaista iältään 18-64 vuotta. Vastaajista 64,5% oli miehiä, ja yli puolella vastaajista oli yliopistotutkinto. Heidän valitsemistaan kuudesta attribuutista eli ominaisuudesta kaksi, sensoridatan tyyppi ja tiedon tallennuspaikka, liittyivät pääosin yksityisyyteen. Seuraavat kaksi, automaation luotettavuus ja käytön tietoisuus oli luokiteltu luottamukseen liittyviksi. Loput kaksi attribuuttia olivat automaation taso ja sovellusalue. Kuvassa 5.1 on esitetty Schomakers et al. tutkimuksen attribuuttien suhteellinen tärkeys toisiin attribuutteihin verrattuna prosentteina. Vastaajien mielestä kaksi selkeästi tärkeintä attribuuttia olivat automaation luotettavuus (26,3%) ja tietojen tallennuspaikka (24,9%). Kolmanneksi tärkein attribuutti oli automaation taso (16,0%). Kolme muuta attribuuttia saivat kukin hieman yli 10% osuuden.



Kuva 5.1: Schomakers et al. [92] tutkimuksen attribuuttien tärkeys prosentteina

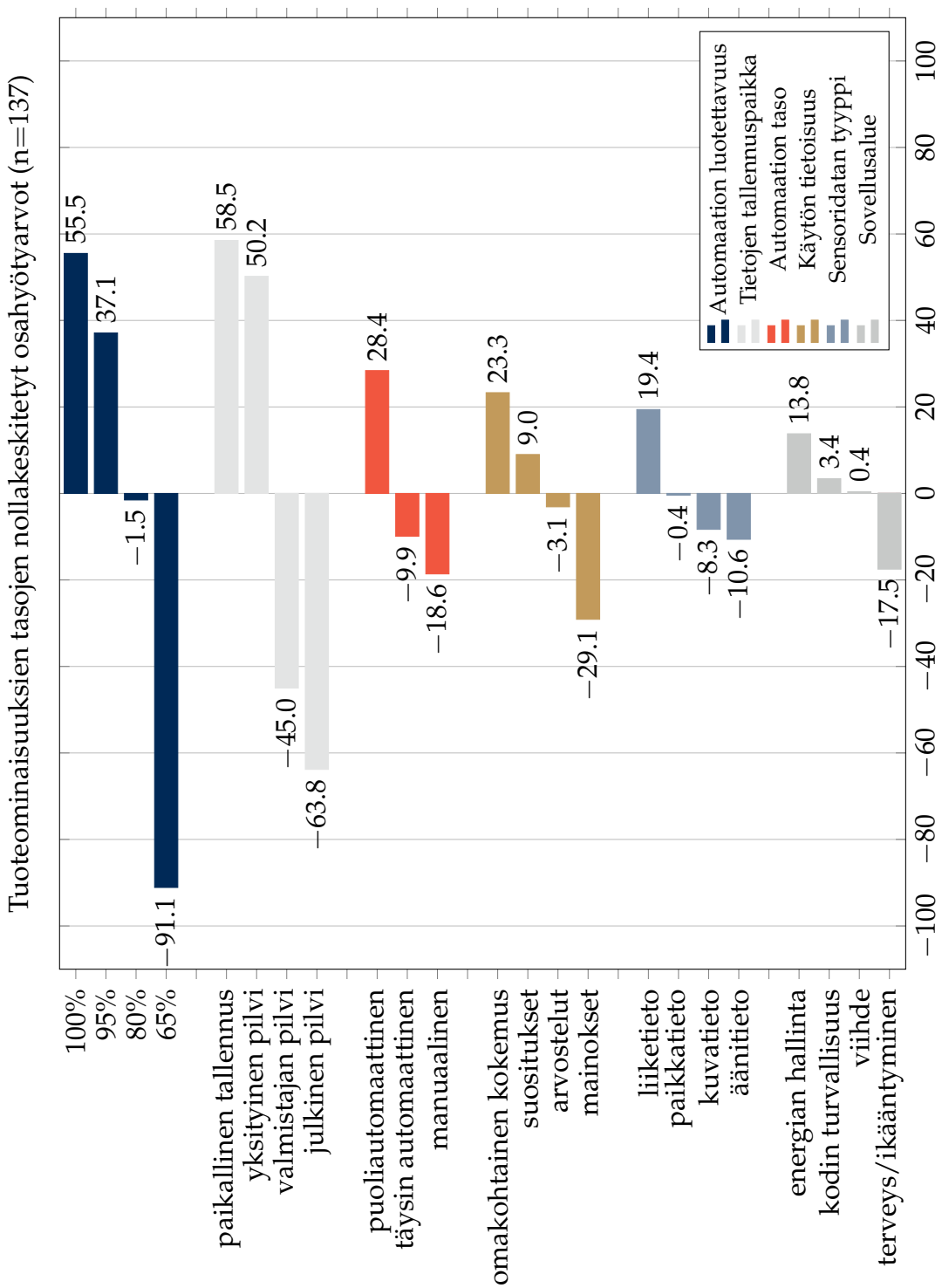
Vaikka suurella osalla vastaajista oli huoli omasta yksityisyydestään, monella oli suunnitelmissa hankkia, tai oli jo käytössä oma kotiautomaatiojärjestelmä. Yksityisyyden osalta vastaajat suosivat datan paikallista tallentamista tai yksityistä pilvipalvelua viranomaisten tai valmistajien pilvipalvelujen sijaan. Toisena yksityisyyteen liittyvänä attribuuttina tutkittiin sensoridatan tyyppiä, joista mieluisimpana pidettiin liikkeentunnistusta ja vähiten mieluisana äänentunnistusta.

Luottamuksen osalta täydellinen toimintavarmuus nousi tärkeimmäksi ominaisuudeksi tutkimuksessa. Omakohtaiset kokemukset olivat tärkein tapa kasvattaa luottamusta järjestelmään. Vastaajista vain kahdella ei ollut minkäänlaista kokemusta tai ennakkotietoa kotiautomaatiojärjestelmistä. Toiseksi tärkein luottamusta lisäävä tekijä oli ystävien ja tuttavien suositukset ja vasta sen jälkeen sivustoilla annetut julkiset arvostelut. Kaikkein vähiten luotettiin mainoksiin.

Koska kyse on automaatiojärjestelmästä, manuaalinen järjestelmä ei miellyttänyt vastaajia, mutta toisaalta myös täyttä automaatiotakaan ei haluttu. Täysi automaatio vaatisikin usein enemmän sensoridataa. Yksityisyyden tunne perustuu osittain kontrolliin, joten puoliautomaattinen järjestelmä oli mieluisin vastaajien mielestä.

Sovellusalueet oli jaettu seuraaviin ryhmiin: energian hallinta, kodin turvallisuus, viihde sekä terveys/ikääntyminen. Energian hallinta oli suosituin sovellusalue ja toisena tuli kodin turvallisuus. Sovellusalueella oli vähäisin merkitys mieltymysten suhteen.

Conjoint-analyysin tuoteominaisuuksien tasojen osahyötyarvot on esitetty kuvassa 5.2. Siinä jokaista tuoteominaisuuden tasoa on verrattu kaikkien vastaajien osahyötyarvojen keskiarvoon. Koska eri tuoteominaisuuksien osahyötyjen keskiarvot vaihtelevat ja tulokset on haluttu esittää vertailun helpottamiseksi samassa kaaviossa, on keskimääräistä tasoa vastaava pistemäärä muutettu nollaksi [62]. Esimerkiksi toiseksi tärkeimmän tuoteominaisuuden, tietojen tallennuspaikan, tasojen nollakeskitetyistä osahyötyarvoista voidaan päätellä, että tuoteominaisuuden ollessa paikallinen tallennus tai yksityinen pilvi, kombinaatio tulee todennäköisemmin valituksi, kuin sen ollessa valmistajan pilvi tai julkinen pilvipalvelu.



Kuva 5.2: Kotiautomaatiojärjestelmän tärkeimmät ominaisuudet suosituimmuusjärjestyksessä [92]

ISO 25010-standardi [95] esittelee käytönaikaisen laadun mallin, jossa on viisi pääkohtaa: (1) vaikuttavuus, (2) tehokkuus, (3) tyytyväisyys, (4) riskittömyys ja (5) käyttöalueen kattavuus. ISO 25022-standardi [52] puolestaan antaa esimerkkejä käytönaikaisen laadun mittaamisesta. Vaikuttavuutta mitattaessa tarkastellaan kuinka hyvin käyttäjät suoriutuvat määritellyistä tehtävistä ja tehokkuutta voidaan arvioida esimerkiksi tehtäviin käytetyn ajan perusteella. Tyytyväisyyttä mitataan usein kyselyiden avulla tai keräämällä dataa siitä, mitkä ohjelman osat ovat suosituimpia. Riskittömyyden mittauksessa arvioidaan laadun vaikutusta talouteen, maineeseen, ympäristöön ja resursseihin, kuten esimerkiksi henkilöstöön. Käyttöalueen kattavuuden kohdalla arvioidaan kuinka hyvin edellä mainitut kohdat täyttyvät määritetyn käyttöalueen kontekstissa.

5.3.1 Mittari K1: Käyttöarvo

Tässä mittarissa mitataan käyttöarvoa sen perusteella, miten kotiautomaatiojärjestelmä täyttää sille ennalta määritellyt vaatimukset. ISO/IEC standardissa 25022 [52] painotetaan, että käytönaikaisen laadun vertaileminen eri järjestelmien välillä tulee tapahtua samassa määritellyssä kontekstissa. Tässä mittarissa määritellyt vaatimukset muodostavat siis kontekstin, johon perustuu käytönaikaisen laadun mittaaminen seuraavassa mittarissa K2. Määritellyt vaatimukset jaetaan tehtäviksi, joista jokainen tehtävä tulee suorittaa käyttäen testattavan järjestelmän ja laboratorion laitteiston dokumentaatiota sekä testattavan järjestelmän virallista yhteisöpalstaa.

Tehtävien suorittamiseen käytetty aika mitataan ja tehtävillä on tehtäväkohtaiset aikarajat. Tehtävän suoritusnopeudella ei ole vaikutusta pisteisiin, mutta aikarajan täyttyessä tehtävä arvioidaan sen hetkisen edistymisen perusteella. Aikarajat tulee asettaa sellaisiksi, että niiden puitteissa tehtävän pystyy suorittamaan, jos toiminnallisuus on järjestelmässä ja se on hyvin dokumentoitu.

Vaatimusten toteutumista arvioidaan kolmiportaisella asteikolla. Jos vaatimus toteutuu pääosin, eli tehtävä saadaan suoritettua sille määritetyssä aikarajassa, tuloskorttiin merkitään 5 pistettä. Jos vaatimus toteutuu vain osin, eli toiminnallisuus ei täyty täysin, koska toiminnallisuus puuttuu tai annettu aikaraja tuli täyteen, tuloksena on 3 pistettä. Mikäli vaatimuksen toteutuminen jää hyvin vajavaiseksi tai se ei toteudu ollenkaan, tuloksena on 1 piste. Tulokset kerätään taulukon 5.1 mukaiselle tuloskortille, jonka avulla lasketaan mittarin lopulliset pisteet laskemalla tehtäväkohtaisten pisteiden keskiarvo kaavan 5.4 mukaisesti.

Taulukko 5.1: Mittarin K1 tuloskortti

Vaatimus, ja siitä johdettu tehtävä	Ajankäyttö		Täyttyykö vaatimus?		
	aika	max	1, ei	3, osin	5, kyllä
t_1					
...					
t_n					

$$pisteet(K1) = \bar{t} = \frac{t_1 + \cdots + t_n}{n} \tag{5.4}$$

5.3.2 Mittari K2: Käytönaikainen laatu

Tässä mittarissa mitataan käytönaikaista laatua käyttäen hyödyksi mittarissa K1 luotua kontekstia, joka jakautui tehtäviin. Kun aikaisemmassa mittarissa K1 mitattiin suoriutuuko testaaja tehtävästä annetussa aikarajassa ja kuinka hyvin, tässä käytönaikaista laatua mittaavassa mittarissa testaajan tulee jokaisen tehtävän jälkeen vastata väittämään *"tehtävän suorittaminen oli helppoa"* käyttäen Likert-asteikkoa. Rensis Likertin vuonna 1932 kehittämää asteikkoa käytetään usein mittaamaan asenteita [67]. Yleisesti käytetyssä viisiportaisessa asteikossa vaihtoehdot ja pisteet määräytyvät seuraavasti:

5. täysin samaa mieltä
4. jokseenkin samaa mieltä
3. ei samaa eikä eri mieltä
2. jokseenkin eri mieltä
1. täysin eri mieltä
0. toiminnallisuus puuttuu

Tehtävien aikana testaaja käyttää järjestelmän dokumentaatiota ja käyttöliittymää, joten testaajan tulee vastata myös dokumentaatioon ja käyttöliittymään liittyviin väittämiin käyttäen samaa asteikkoa. Vastaukset väittämiin tallennetaan taulukon 5.2 mukaiselle tuloskortille, jonka avulla lasketaan mittarin K2 lopulliset pisteet laskemalla eri väittämien keskiarvo kaavan 5.5 mukaisesti.

Taulukko 5.2: Mittarin K2 tuloskortti

Tehtävien ja niiden suoritukseen liittyvien järjestelmän osien ja toimintojen arviointi	Arviointi					
	0	1	2	3	4	5
t_1 suorittaminen oli helppoa						
...						
t_n suorittaminen oli helppoa						
v_1 Dokumentointi on kattava						
v_2 Dokumentti on helppolukuinen						
v_3 Käyttöliittymä on intuitiivinen						

$$pisteet(K2) = \bar{tv} = \frac{t_1 + \cdots + t_n + v_1 + v_2 + v_3}{n + 3} \tag{5.5}$$

5.4 Mittausalue T: Tietoturva ja yksityisyys

Schuster ja Habibipour [93] kysyivät 251 ruotsalaisen huolenaiheita ja suhtautumista kotona käytettävien IoT-laitteiden yksityisyyteen liittyvissä kysymyksissä. Vastaajat olivat 18-82 -vuotiaita mediaanin ja keskiarvon ollessa 44 vuotta. Vastausten perusteella IoT-laitteiden turvallisuus ja yksityisyyden suoja huolestutti vastaajia, ja he pelkäsivät tietojen vuotamista tai väärinkäyttöä. Suuri osa vastaajista ei myöskään uskonut olemassa olevien tuotteiden olevan turvallisia, eivätkä vastaajat luottaneet valmistajien kykyyn huolehtia yksityisyyden suojasta. Vain 8% vastaajista piti älykodin tuotteiden turvallisuutta hyvänä ja 35% mielestä turvallisuus- ja yksityisyysseikat rajoittivat heidän haluaan ostaa tai käyttää tuotteita.

Vastaajia pyydettiin myös laittamaan kirjallisuudessa esitettyjä yksityisyyden suojaamisen tarkoitettuja toimenpiteitä tärkeysjärjestykseen. Vastaajien mielestä selkeästi tärkein toimenpide oli pyytää käyttäjältä suostumus tietojen lähettämiseen kolmansille osapuolille. Toiseksi tärkeimpänä pidettiin oletusarvoisen tietosuojan (engl., Privacy by Design, PbD) noudattamista, joka tarkoittaa, että järjestelmässä on oletusarvoisesti tiukimmat mahdolliset yksityisyysasetukset. Salauksen käyttäminen oli vasta kolmantena ja neljänneksi tärkeimpänä toimenpiteenä pidettiin etukäteisen suostumuksen (opt-in) ja jälkikäteen tapahtuvan kieltämisen (opt-out) periaatteen toteutumista tietojen keräämisen ja käytön yhteydessä. Viidenneksi tärkein seikka oli valmistajan turvallisuusstrategia tuotteen elinkaaren ajalle.

Vähiten merkittävinä huolenaiheina pidettiin kaksivaiheista tunnistautumista sekä lakien ja asetusten toteuttamista, vaikka esimerkiksi Euroopan unionin yleinen tietosuoja-asetus GDPR velvoittaa tuotteiden ja sovellusten tuottajia toteuttamaan sisäänrakennetun ja oletusarvoisen tietosuojan [16]. Schuster ja Habibipour [93] totesivatkin, että tiedon ja kokemuksen välillä on ilmeinen ero, sillä tietämättömät ihmiset liioittelivat lähes kaikkia riskejä ja määrittelivät kaikki toimenpiteet tärkeiksi.

Taulukko 5.3: Toimenpiteet tärkeysjärjestyksessä [93].

Toimenpide	Tärkeä(%)	Ei tärkeä(%)
Suostumus tietojen lähettämiseen	91	2
Oletusarvoinen tietosuoja	85	3
Salaus	84	5
Datan keräämisen suostumus/kielto	84	6
Valmistajan turvallisuusstrategia	80	4
Haittaohjelmien torjunta	73	11
Vastuunjako	72	5
Sertifioinnit ja merkinnät	69	9
Tietoturvapäivitykset	67	13
Hyökkäyksestä toipuminen ("self-healing")	65	13
Tunkeilijan havaitseminen	63	14
Lait ja asetukset (mm. GDPR)	63	10
Kaksivaiheinen tunnistautuminen	61	17

Morrison et al. [70] tekivät systemaattisen kirjallisuuskartoituksen ohjelmistojen tietoturvamittareista ohjelmistokehityksen eri vaiheissa. He löysivät 4 818 artikkelia, joista he valitsivat 71 artikkelia. Niissä oli yhteensä 324 uniikkia tietoturvamittaria. Heidän arvionsa mukaan 85% mittareista oli artikkelin kirjoittajien ehdottamia ja niistä noin 60% oli arvioitu joko kirjoittajien tai muiden toimesta. Mittarit painottuivat toteutus- ja käyttövaiheisiin. Ohjelmistokehityksen vaatimusten määrittelylle, suunnittelulle ja testausvaiheelle oli suhteessa paljon vähemmän mittareita.

Yksi yleisimmin käytetyistä tietoturvaheikkouksien raportointijärjestelmistä on CVE (Common Vulnerabilities and Exposures). CVE järjestelmään ilmoitettujen haavoittuvuuksien määrä on hyvin yleinen mittari ja siitä oli myös johdettu haavoit-

tuvuuksien tiheys suhteuttamalla haavoittuvuuksien määrä koodirivien määrään (LOC). Morrison et al. [70] totesivat, että haavoittuvuuksia laskettaessa artikkelit käyttivät myös muita raportointijärjestelmiä tai laskivat vain haavoittuvuudet, joiden vakavuus oli arvioitu CVSS (Common Vulnerability Scoring System) asteikolla yli määritellyn raja-arvon. Eri järjestelmiä ja niiden tietoturvaa verrattaessa on otettava huomioon useita eri seikkoja. Mitä pidempään ohjelmisto on ollut saatavilla, sitä suuremmalla todennäköisyydellä sen koodista on jo löytynyt useampia haavoittuvuuksia. Ohjelmiston ominaisuudet, laajuus ja käytetty ohjelmointikieli vaikuttavat myös lähdekoodin kokoon, minkä vuoksi haavoittuvuuksien suhteuttaminen koodirivien määrään ei ole vertailukelpoinen eri ohjelmien välillä.

Toinen yleinen mittari on MTTR (Mean Time to Remediate), jolla mitataan keskimääräistä aikaa haavoittuvuuden julkaisun ja korjaavan päivityksen julkaisemisen välillä. Sen osalta haasteena on saada selville, milloin haavoittuvuus on tullut kehittäjien ja yhteisön tietoon. CVE haavoittuvuudet lisätään järjestelmään silloin, kun ne saavat ID numeron, mutta sen perusteella ei voi päätellä mitään siitä, milloin haavoittuvuus on löydetty, ilmoitettu tai julkaistu [100]. CVE haavoittuvuuksien ilmoitusprosessi etenee normaalisti niin, että haavoittuvuus ilmoitetaan ensin CVE ohjelmaan kuuluvan ohjelmiston kehittäjille ja he raportoivat sen CVE järjestelmään ja hakevat haavoittuvuudelle ID numeroa [101]. Prosessi mahdollistaa sen, että kehittäjille jää aikaa valmistella ja julkaista korjauspäivitys ennen haavoittuvuuden julkaisemista. Avoimen lähdekoodin ohjelmistot toimivat usein vapaaehtoisvoimin, joten resurssit ovat rajalliset, mutta selkeä tietoturvapolitiikka ja haavoittuvuuksien raportointiprosessi voivat auttaa saamaan tietoa haavoittuvuuksista.

Hatzivasilis et al. [38] ehdottivat menetelmää nimeltä SPD (Security, Privacy and Dependability), jossa yleistä turvallisuutta mitataan ohjelman hyökkäyspinta-alan mukaan ja yksityisyyttä ja käyttövarmuutta niihin liittyvien määräysten ja standardien mukaan. Hyökkäyspinta-alan mittausmenetelmänä he käyttivät Manadhatan ja Wingin [66] luomaa mittaria nimeltä ASM (Attack Surface Metric), joka perustuu Howard et al. [46] luomaan hyökkäyspinta-alan osamäärään RASQ (Relative Attack Surface Quotient). Manadhatan ja Wing toteavat, että heidän ASM mittausmenetelmä vaatii järjestelmän lähdekoodin hyökkäyspinta-alan laskemiseksi ja vaikka avoimen lähdekoodin järjestelmien osalta koodi on saatavilla, sen koko voi joskus olla kohtuuttoman suuri. He ehdottivatkin mittausmenetelmälle laajennusta, jossa määriteltäisiin systemaattinen tapa arvioida järjestelmän hyökkäyspinta-ala ilman lähdekoodia.

5.4.1 Mittari T1: Tietoturva

Jos järjestelmällä ei ole määriteltyä tietoturvapolitiikka (engl., security policy), on oletettavaa, ettei tietoturvaan liittyviä asioita ole mietitty huolella. Osallistuminen haavoittuvuuksien raportointiohjelmiin ja helposti löydettävät korjaukset haavoittuvuuksiin luovat luottamusta siihen, että haavoittuvuuksiin reagoidaan.

Mittari T1 sisältää neljä eri vaatimusta, joiden toteutumista arvioidaan kolmiportaisella asteikolla. Jos vaatimus toteutuu pääosin, tuloksena on 5 pistettä. Jos vaatimus toteutuu vain osin, tuloksena on 3 pistettä. Mikäli vaatimuksen toteutuminen jää hyvin vajavaiseksi tai se ei toteudu ollenkaan, tuloksena on 1 piste.

- v_1 Tietoturvapolitiikka on olemassa ja se on helposti löydettävissä
- v_2 Tietoturvapolitiikka on kattava ja selkeä, ja se sisältää ohjeet haavoittuvuuksien raportointiin
- v_3 Ohjelmisto osallistuu johonkin CVE raportointiohjelmaan
- v_4 Raportoidut haavoittuvuudet ja korjaukset niihin on helposti löydettävissä

Mittarin lopputuloksena ilmoitetaan laskemalla vaatimusten pisteiden keskiarvon kaavan 5.6 mukaisesti.

$$pisteet(T1) = \bar{v} = \frac{v_1 + \cdots + v_4}{4} \tag{5.6}$$

5.4.2 Mittari T2: Yksityisyys

Kotiautomaatiojärjestelmien leviämisen yhtenä suurimmista hidasteista on pidetty pelkoa yksityisyyden suojan murenemisesta. Schusterin ja Habibipourin [93] tutkimuksen suostumus tietojen lähettämiseen ja oletusarvoinen tietosuoja olivat tärkeimmät toimenpiteet yksityisyyden suojan varjelemisessa. Avoimen lähdekoodin kotiautomaatiojärjestelmien perusversiot käyttävät usein paikallista tallennusta, joten tietojen lähettämisen sijaan, tässä mittarissa keskitytään oletusarvoiseen tietosuojaan ja siihen mitä tietoja järjestelmä kerää, onko vastaaminen pakollista ja mitkä ovat järjestelmän oletusasetukset tietosuojan osalta.

- v_1 Järjestelmä kysyy vain olennaisia tietoja tai tietojen syöttäminen on vapaaehtoista
- v_2 Järjestelmän oletusasetukset ovat tietosuojan kannalta tiukimmat

Tämän mittarin vaatimusten toteutumista arvioidaan samalla kolmiportaisella asteikolla kuin edeltävässä tietoturvaa mittaavassa mittarissa T1 (kts. 5.4.1). Mittarin lopputuloksena ilmoitetaan vaatimusten pisteiden keskiarvo kaavan 5.7 mukaisesti.

$$pisteet(T2) = \bar{v} = \frac{v_1 + v_2}{2} \tag{5.7}$$

6 Testaus

Tässä luvussa testataan edellisessä luvussa luotu mittaristo käyttäen otantana alaluvuissa 3.3.1 - 3.3.4 esiteltyjä avoimen lähdekoodin kotiautomaatiojärjestelmiä. Kasanen et al. [60] pitivät konstruktiivisen tutkimuksen yhtenä tärkeimmistä ominaispiirteistä markkinatestiä, jossa toteutetun konstruktion eli ratkaisun toimivuus testataan käytännössä.

Ennen varsinaista testausta, ensimmäisessä alaluvussa 6.1 määritellään testaukselle konteksti ja esitellään testauksessa käytetty laitteisto. Toisessa alaluvussa 6.2 suoritetaan yhteisöön ja arvonluomiseen liittyvät mittaukset. Kolmannessa alaluvussa 6.3 mitataan järjestelmien käyttöarvoa ja käytönaikaista arvoa järjestelmä kerrallaan käyttäen aiemmin määritellyssä kontekstissa ja käyttäen esiteltyä laitteistoa. Yksityisyyden ja tietoturvan mittaukset ovat neljännessä alaluvussa 6.4 ja alaluvussa 6.5 kootaan testaustulokset yhteen.

6.1 Konteksti ja käytetty laitteisto

Tämän testin kontekstin määrittelyssä käytetään Schomakers et al. [92] tutkimusta, joka esiteltiin luvussa 5.3. Heidän tutkimuksensa perusteella käyttäjät odottivat kotiautomaatiojärjestelmän automaatioilta täydellistä luotettavuutta, tiedot tulisi tallentaa paikallisesti, automaation taso olisi puoliautomaattinen ja perustua liikkeentunnistukseen ja tärkein sovellusalue käyttäjien mielestä oli energiankulutuksen hallinta.

Käytönaikaista laatua voidaan mitata tarkkailemalla ohjelman käyttäjiä tai simuloimalla käyttöä laboratoriossa [52]. Tämän testin suorittamiseksi hankittiin laboratoriolle kuvassa 6.1 oleva laitteisto, joka koostui koteloidusta Raspberry Pi 4B yhden piirilevyn tietokoneesta ja 16 GB MicroSD-muistikortista. Raspberryn sisäisen Bluetooth radion lisäksi tarvitaan Zigbee-yhdyskäytävä, joka liitettiin USB-jatkojohdolla häiriöiden välttämiseksi [50]. Sensoreina ja toimilaitteina käytettiin monikäyttöistä RuuviTag-sensoria ja IKEA Trådfri -tuoteperheen tuotteita. Taulukossa 6.1 on tarkemmat tiedot laitteistosta mukaan lukien edullisin hinta EU-alueen verkkokaupoista tilattuna.

Kuva 6.1: Laboratorion laitteisto

Taulukko 6.1: Laboratoriolaitteet ja hinnat EU-alueen verkkokaupoissa 16.3.2024

Tuote	Hinta
Raspberry Pi 4 Model B - 2 GB	54,80€
Raspberry Pi USB-C Power Supply - Multi 5V 3A	10,58€
Official Raspberry Pi 4 B Case - Black/Grey	10,58€
Sandisk EDGE 16GB Micro SD Class A1	6,50€
USB 2.0, A - A jatkojohto	2,90€
Verkkokaapeli, CAT 5e UTP, 0,5 m	1,50€
Phoscon ConBee II Zigbee USB-yhdyskäytävä	49,90€
IKEA Trådfri Led-lamppu E27 806 lm	8,99€
IKEA Trådfri langaton liiketunnistin	7,99€
IKEA Trådfri Kaukosäädin+pistorasia-setti	19,99€
RuuviTag BLE lämpötila, kosteus, ilmanpaine ja liiketunnistin	39,99€
Yhteensä	213,72€

Alaluvussa 6.3 suoritettavassa järjestelmien käyttöarvon ja käytönaikaisen laadun mittauksessa kukin avoimen lähdekoodin kotiautomaatiojärjestelmä asennetaan laboratorion tietokoneelle ja pyritään luomaan seuraavat automaatiot käyttäen apuna ensisijaisesti vain ohjelmiston tai laitteiston dokumentaatiota ja järjestelmän virallista yhteisöpalstaa:

1. Lampun automaatio: lamppu syttyy palamaan, kun liiketunnistin havaitsee liikettä, ja se sammuu 30 sekunnin päästä siitä, kun liikettä ei enää havaita
2. Kytkimen automaatio
    - Kytkimen ON-painalluksen jälkeen pistorasia antaa virtaa 1 minuutin ajan
    - Kytkimen OFF-painallus kytkee pistorasian pois päältä
3. Pistorasian automaatio
    - Jos pistorasia ei ole päällä, se alkaa antamaan virtaa lämpötilan laskiessa alle 10 °C
    - Jos pistorasia on päällä, se ei enää anna virtaa, kun lämpötilan nousee yli 18 °C

Koska laitteistoja on vain yksi, asennukset ja testaus tehdään yhdelle kotiautomaatiojärjestelmälle kerrallaan. Testaus on jaettu seuraaviin vaiheittaisiin tehtäviin, joiden suorittamiselle on asetettu aikarajat:

1. Käyttöjärjestelmän asentaminen, aikaraja 2 tuntia
2. RuuviTag-sensorin integroiminen, aikaraja 2 tuntia
3. Zigbee-yhdyskäytävän integroiminen, aikaraja 2 tuntia
4. IKEA Trådfri -tuotteiden integroiminen, aikaraja 2 tuntia
5. Lampun automaation ohjelmointi, aikaraja 1 tunti
6. Kytkimen automaation ohjelmointi, aikaraja 1 tunti
7. Pistorasian automaation ohjelmointi, aikaraja 2 tuntia

6.2 Mittausalue Y

Valituista neljästä avoimen lähdekoodin kotiautomaatiojärjestelmistä Gladys Assistant ja Jeedom ovat ranskalaista alkuperää, josta selvimpänä merkkinä on ranskankielinen yhteisöpalsta. Jeedomin yhteisöpalstalla on erillinen englanninkielinen alue, jossa on viiden vuoden aikana aloitettu vain kahdeksan viestiketjua. Gladys Assistant puolestaan perusti erillisen englanninkielisen yhteisön kaksi vuotta sitten. Gladys Assistantin strategia näyttää toimivan, sillä englanninkielinen yhteisö kasvaa selvästi eniten ja toisaalta Jeedomin kasvu on vertailuryhmän pienin.

Home Assistantin kasvu on myös suurta ottaen huomioon sen, että se on ainoa vertailun kotiautomaatiojärjestelmistä, jolla on tarjolla myös virallinen palvelin Discord keskustelualustalla. Discord suunniteltiin alkujaan videopeliyhteisöille viestintäsovellukseksi, mutta sen käyttö on laajentunut huomattavasti. Home Assistantin Discord-palvelimella on kanavia, joilla voi pyytää apua ja tukea eri ongelmiin. Home Assistantin Discord-palvelimella on 27.3.2024 lähes 133 500 jäsentä, joista kirjoittamishetkellä online-tilassa oli lähes 16 000 jäsentä.

Mittausten perusteena olevat tiedot on esitetty taulukossa 6.2. Kaikki yhteisöpalstat käyttivät samaa avoimen lähdekoodin alustaa (Discourse) yhteisöpalstoilleen, joten tilastotietojen laskentatapa järjestelmien välillä on yhteneväinen. Yhteisöpalstojen perustamiskuukausi on pyritty selvittämään mahdollisimman tarkasti käyttäen lähteinä kotiautomaatiojärjestelmän sivustoa ja siellä olleita uutisia ja blogeja. Jos niiden avulla ei ole saatu selville perustamisajankohtaa on aika pyritty selvittämään etsimällä kotiautomaatiojärjestelmän perustajan yhteisöpalstalle liittymispäivämäärä tai palstalle tehdyn ensimmäisen viestin päivämäärä.

Taulukossa 6.3 on pisteet yhteisöpalstojen kasvun ja aktiivisuuden mittareille Y1 ja Y2. Pisteytyksessä Gladys Assistantin kohdalla otetaan huomioon ranskankielisen ja englanninkielisen yhteisöpalstan tulokset niiden kokoon suhteutettuna.

Git-versionhallintajärjestelmistä kaksi suurinta ovat GitHub ja Gitlab. Kaikki vertailun järjestelmät käyttävät GitHubia. Taulukossa 6.4 on esitetty eri kehitysprojektien core-tietovarastojen tähdet ja laskettu niiden perusteella Stargazers-luku, jonka perusteella määräytyy mittarin Y3 pisteet.

Taulukko 6.2: Yhteisöpalstojen viimeisen 30 päivän tilastot ja kaikki rekisteröityneet käyttäjät (24.03.2024) [26, 27, 40, 55, 73] sekä yhteisön keskimääräinen kasvu

	Gladys Assistant		HA OS	Jeedom	OpenHAB
	ranska	englanti	englanti	ranska	englanti
Ketjuja	27	3	3 500	761	291
Viestejä	338	29	48 100	11 700	3 900
Rekisteröityneet	20	12	5 200	157	245
Akt. käyttäjiä	95	17	40 300	2 900	2 500
Rek. käyttäjiä yhteensä	719	91	226 000	19 800	43 600
Perustettu	4/2017	4/2022	3/2016	8/2019	7/2015
Ikä kuukausissa	83 kk	23 kk	96 kk	55 kk	104 kk
Kasvu/kk ka.	8,6 / kk	4,0 / kk	2 400 / kk	360 / kk	420 / kk

Taulukko 6.3: Yhteisöpalstojen kasvun (Y1) ja aktiivisuuden (Y2) mittaustulokset

	Gladys Assistant			HA OS	Jeedom	OpenHAB
	fr	en	fr+en[a]	englanti	ranska	englanti
Y1 (kasvusuhde)	2,3	3,0	2,4	2,2	0,4	0,6
Y1 Pisteet			5	5	0	1
Y2 (aktiivisuus-%)	13,2 %	18,7 %	13,8 %	17,8 %	14,6 %	5,7 %
Y2 Pisteet			4	5	4	2

[a] suhteutettuna rekisteröityjen käyttäjien lukumäärään

Taulukko 6.4: Kehitysprojektien core-tietovaraston tähdet (26.03.2024) [22, 23, 24, 25], Stargazers-luku ja kiinnostavuuden (Y3) mittaustulos

	Gladys Assistant	HA OS	Jeedom	OpenHAB
Tietovarasto	/Gladys	/core	/core	/openhab-core
Tähdet	2 481	68 085	383	3 433
Stargazers-luku	3,4	4,8	2,6	3,5
Y3 Pisteet	3,4	4,8	2,6	3,5

6.3 Mittausalue K

Tässä testauksen empiirisessä osiossa vertailun neljän kotiautomaatiojärjestelmän käyttöarvoa ja käytönaikaista arvoa mitataan käyttäen laboratorion laitteistoa. Mittausalueen molempien mittareiden mittaukset suoritetaan samalla kertaa, yksi järjestelmä kerrallaan. Ensimmäisessä alaluvussa 6.3.1 suoritetaan mittaukset Gladys Assistantin osalta, toisessa alaluvussa 6.3.2 on vuorossa Home Assistant, jonka jälkeen alaluvussa 6.3.3 seuraa Jeedom. Lopuksi alaluvussa 6.3.4 tehdään mittaukset vielä OpenHAB-järjestelmän osalta.

6.3.1 Gladys Assistant

Tehtävä 1: Käyttöjärjestelmän asennus

Gladys Assistantin sivustolta löytyy dokumentaatio helposti `Docs` -valikosta, jonka aloitussivu kertoo, että käyttäjä on löytänyt oikean oppaan aloittelijoille. Lyhyen laitehankintoihin ja suunnitteluun liittyvän pohjustuksen jälkeen sivun lopussa on linkki asennussivustolle.

Gladys Assistantin asennustiedosto Raspberry Pi 4:lle on kooltaan 505 MB. Sen siirtämiseksi MicroSD-kortille kehotetaan käyttämään Balena Etcher -ohjelmaa. Balenan sivustolla on kolme lataustiedostoa Windowsille, yksi MacOS:lle ja kaksi Linuxille. Aloittelijalle saattaa tulla vaikeus valita oikea Windows tiedosto. Valitsimme vaihtoehdon `Windows (x86|x64) (Portable)`, jotta vältymme ylimääräisiltä asennuksilta. Etcherin lataustiedoston koko on 133 MB ja Gladys Assistant lataustiedoston asennus onnistuu sen avulla helposti ilman erillisiä ohjeita. Jälkeen päin selviää, että Gladys Assistantin asentamiseen olisi voinut käyttää myös Raspberry Pi Imager -ohjelmaa.

Kun MicroSD-kortti on laitettu Raspberry Pi -tietokoneeseen, joka on yhdistetty kotireitittimeen, kestää hetken aikaa ennen kuin Gladys Assistant on käynnistynyt ja sitä pääsee käyttämään osoitteessa `http://gladys.local`. Ensimmäiseksi pitää luoda järjestelmälle käyttäjä. Nimitietoina vaaditaan niin etunimi kuin sukunimikin, mutta samalla painotetaan, että tiedot tallennetaan vain paikallisesti. Myös sähköposti kysytään, sillä se toimii käyttäjätunnuksena. Kenttään pitää laittaa sähköpostiosoitteen vaatimukset täyttävä tieto, mutta asennus onnistuu myös käyttämällä sähköpostiosoitetta, jota ei ole olemassa. Silloin unohtuneen salasanan vaihtaminen sähköpostilinkin avulla ei tietenkään toimi.

Muissa asetuksissa kysytään vain käytettävät mittajärjestelmät lämpötilan ja välimatkojen osalta. Talon tai asunnon sijainnin voi määritellä kartalta vapaasti, eli selain ei esitä paikannuspyyntöä. Samalla voi määritellä taloon tai asuntoon huoneita.

Navigointi Gladys Assistantissa on helppoa. Ylhäällä on kuusi valikkoa: `Home`, `Chat`, `Integrations`, `Calendar`, `Maps` ja `Scenes`. Dokumentaatiosta löytyy kahta eniten konfiguraatiota vaativaa valikkoa (`Integrations` ja `Scenes`) vastaavat kansiot, joka tekee olon turvalliseksi. Käyttöliittymä on selkeä ja pelkistetty.

Tehtävä 2: RuuviTagin integroiminen

Asennuksen jälkeen Gladys Assistantin opas ei enää anna suosituksia seuraavista toimenpiteistä. Koska seuraava tehtävä on yhdistää RuuviTag, joka käyttää BLE teknologiaa, `Integrations` -valikko tuntuu luonnollisimmalta paikalta aloittaa. Sieltä löytyykin kohta `Bluetooth`, jossa voi yrittää etsiä ja lisätä Bluetooth-laitteita. Valinnan `Bluetooth discover` takaa löytyy `Scan` -nappi, jota painamalla löytyy testiympäristöstä 41 laitetta, joista vain kolmella on selkeä tunniste ja muista näkyy vain MAC-osoite.

RuuviTagin MAC-osoite selvisi puhelimeen asennetusta Ruuvi Station sovelluksesta. Sama MAC-osoite löytyi Gladys Assistantin listauksesta, mutta skannauksesta oli luultavasti kulunut sen verran aikaa, että liittäminen Gladys Assistantiin ei ensin onnistunut `Connect in Gladys` -napista. Uusi yritys avasi näkymän, jossa laite määritellään tarkemmin. Huoneen lisäksi on mahdollista määrittää laite sellaiseksi, joka ilmaisee läsnäoloa ("Use this device as presence sensor"). RuuviTagin asetusten tallennus ei näytä onnistuvan ilman, että se asetetaan tällaiseksi sensoriksi. Myöhemmin selviää, että näillä asetuksilla RuuviTagia ei voi käyttää lämpötilatietoja vaativissa automaatioissa, joten RuuviTag-integraatio puuttuu.

Tehtävä 3: Zigbee USB:n integroiminen

Zigbee USB-yhdyskäytävän asennukseen ei ensin löydy ohjetta. Pienellä selailulla `Integrations`-valikosta löytyy kohta `Zigbee2mqtt`, jonka `Settings`-valikosta löytyy paikka USB-yhdyskäytävän asentamiselle. Ohjeet löytyvät dokumentoinnin vastaavasta kohtaa. Vaikka ohjeissa esitellään eri valmistajan USB-yhdyskäytävän asennus, Phoscon ConBee II on myös listattuna ohjelman valikossa ja sen asentaminen sujuu ongelmitta.

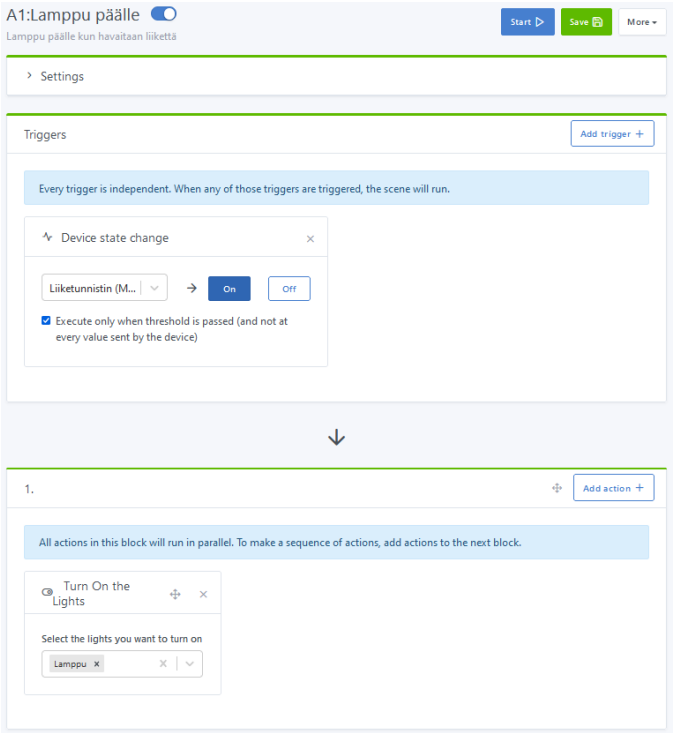
Tehtävä 4: IKEA Trådfri integroiminen

Samalla sivulla Zigbee USB-yhdyskäytävän asennuksen kanssa on ohjeet, miten lisätä Zigbee-laitteita järjestelmään. `Zigbee2MQTT`-valikon `Discover`-sivulla pitää ensin sallia laitteiden liittyminen ("Permit joining") Zigbee-konfiguraatioon, jonka jälkeen kehotetaan painamaan `Scan`-nappia laitteiden löytämiseksi. `Scan`-nappia ei löydy sivustolta, eikä ohjesivukaan kerro missä sen pitäisi olla. Englanninkieliseltä yhteisöpalstalta löytyy viestiketju, jossa kerrotaan, että nappi on poistettu ja dokumentti on päivitetty. Järjestelmässä asetussivulla viitataan yhä nappiin, jota ei ole, eikä Zigbee-laitteet ilmesty sivulle, vaikka ne aktivoidaan paritusta varten. Edellä mainitun viestiketjun aloittaja sanoo tehneensä paritukset suoraan Zigbee2mqtt-sivulla. `Zigbee2MQTT`-valikon `Setup`-sivulta löytyykin linkki, jonka avulla pääsee Zigbee2MQTT käyttöliittymään. Siellä IKEA Trådfri -tuotteiden lisääminen onnistuu viemällä laitteen USB-tikun lähelle ja aktivoimalla parituksen tuotteen mukana tulleiden ohjeiden mukaan. Pistorasia tosin yhdistyi vasta resetoinnin jälkeen, mikä luultavasti johtuu siitä, että sen mukana toimitettu kaukosäädin oli jo valmiiksi paritettu sen kanssa.

Tehtävä 5: Lampun automaation ohjelmointi

Lampun automaation ohjelmointi alkaa helposti. Ylävalikossa oleva `Scenes`-teksti vie valikkoon, jossa voi luoda uuden automaation klikkaamalla nappia `New +`, jonka jälkeen automaatiolle tulee valita nimi ja valita sille ikoni. Monista ikoneista huolimatta tälle automaatiolle oli vaikea löytää sitä kuvaava ikoni. Itse automaation luomiseksi täytyy ensin luoda sille laukaiseva tekijä `Add trigger +` -nappia painamalla ja sille seuraus `Add action +` -nappia painamalla. Molempien luominen on hyvin intuitiivista. Ainoastaan valintaruutu "Execute only when threshold is passed (and not at every value sent by the device)" herättää kysymyksiä.

Englanninkieliseltä yhteisöpalstalla kysytään, miksi käytössä on vain binäärinen tieto siitä, havaitseeko liiketunnistin liikettä [29]. IKEA liiketunnistin lähettää occupancy-tietoa, eli tietoa siitä, onko joku ollut havaintoalueella määritellyn ajan (occupancy_timeout) sisällä [107]. Lampun automaation ohjelmointi liiketunnistukseen perustuen siis onnistuu, mutta vain osin, koska emme pysty vaikuttamaan katkaisuaikaan. Käytettäessä kuvan 6.2 mukaista automaatiota, laboratoriolaitteiston lamppu sammui 3 minuutin kuluttua siitä, kun liikettä oli ensimmäisen kerran havaittu, riippumatta raja-arvon valintaruudun arvosta.



Kuva 6.2: Gladys Assistant ja lampun automaatio

Tehtävä 6: Kytkimen automaation ohjelmointi

Edellisen tehtävän jälkeen tämän automaation ohjelmointi sujui helposti käyttäen kahta erillistä automaatiota. Ensimmäisessä automaatiossa pistorasia laitetaan päälle, kun kytkimen I-nappia painetaan. Sitä seuraa toinen toiminto, jossa odotetaan 1 minuutti. Odotusajan lisäämiseen on valmis valinta, jonka avulla se oli helppo ohjelmoida. Lopuksi odotusajan jälkeen ohjelmoidaan kolmas toiminto katkaisemaan virrat pistorasiasta. Toisessa automaatiossa kytkimen O-napin painallus kytkee pistorasian pois päältä.

Tehtävä 7: Pistorasian automaation ohjelmointi

Koska RuuviTag-sensorin integroiminen ei onnistunut, tätä automaatiota ei voinut suorittaa. Sensorin valmistaja tarjoaa erillistä Ruuvi Gateway -nimistä yhdyskäytävää sensorien liittämiseen, mutta hakukoneiden avulla löytyy myös useita tapoja yhdistää Ruuvi-sensorit MQTT-palvelimeen, jollaisen voi asentaa Gladys Assistantiin integraatiosivulla.

Yhteenveto

Gladys Assistantin asennus oli helppoa, vaikka se olisi voinut olla vieläkin helpompaa, jos dokumentaatiossa olisi ohjattu käyttämään Raspberry Pi Imageria. Missään vaiheessa asennuksen aikana ei kuitenkaan tarvinnut käyttää komentorivitulkkia. Zigbee-yhdyskäytävän asennus oli helppo tehtävä, mutta IKEA-tuotteiden asennuksessa tuhrautui ylimääräistä aikaa, sillä asennussivulla viitattiin nappiin, jota ei ollut olemassa.

RuuviTag-integraation puuttuminen jäi harmittamaan, sillä sen vuoksi pistorasian automaatiota ei voinut suorittaa. Kytkimen automaatio puolestaan sujui helposti. Automaatioiden ohjelmointi ei vaatinut koodaamista ja oli hyvin yksikertaista. Liiketunnistimen ohjelmointi sammumaan haluttuna aikana ei onnistunut, mutta siinä rajoitteena oli liiketunnistimen tapa ilmoittaa läsnäolosta kolmen minuutin ajan.

Gladys Assistantin kotisivuilla olevaa dokumentaatiota oli helppo lukea, mutta toisaalta ohjeet olivat usein lyhyitä ja teknisiä. Käyttöliittymässä ei ollut mitään ylimääräistä ja navigointi järjestelmässä oli helppoa.

Taulukossa 6.5 on Gladys Assistantin mittarin K1 tuloskortti. Gladys Assistantin pisteiksi tuli K1-mittarilla 3,6. Käytönaikaisen laadun, eli mittarin K2 osalta Gladys Assistantin pisteiksi tuli 3,3. Taulukossa 6.6 on Gladys Assistantin mittarin K2 tuloskortti.

Taulukko 6.5: Mittarin K1 tuloskortti: Gladys Assistant

Vaatimus, ja siitä johdettu tehtävä	Ajankäyttö aika	Ajankäyttö max	Täyttyykö vaatimus? 1, ei	3, osin	5, kyllä
t_1 Käyttöjärjestelmän asennus	30	120			x
t_2 RuuviTagin integroiminen	120	120	x		
t_3 Zigbee USB:n integroiminen	10	120			x
t_4 IKEA Trådfri integroiminen	80	120			x
t_5 Lampun automaation ohjelmointi	60	60		x	
t_6 Kytkimen automaation ohjelmointi	20	60			x
t_7 Pistorasian automaation ohjelmointi	120	120	x		

Taulukko 6.6: Mittarin K2 tuloskortti: Gladys Assistant

Tehtävien ja niiden suoritukseen liittyvien järjestelmän osien ja toimintojen arviointi	Arviointi					
	0	1	2	3	4	5
t_1 Käyttöjärjestelmän asennus oli helppoa					x	
t_2 RuuviTagin integroiminen oli helppoa	x					
t_3 Zigbee USB:n integroiminen oli helppoa						x
t_4 IKEA Trådfri integroiminen oli helppoa				x		
t_5 Lampun automaation ohjelmointi oli helppoa				x		
t_6 Kytkimen automaation ohjelmointi oli helppoa						x
t_7 Pistorasian automaation ohjelmointi oli helppoa	x					
v_1 Dokumentointi on kattava				x		
v_2 Dokumentti on helppolukuinen					x	
v_3 Käyttöliittymä on intuitiivinen						x

6.3.2 Home Assistant

Tehtävä 1: Käyttöjärjestelmän asennus

Home Assistantin dokumentaatio alkaa esittelemällä eri laitteita ja niille sopivia Home Assistantin asennustapoja. HA OS asennus Raspberry Pi:lle on arvioitu helpoksi ja sitä tukee hyvin yksityiskohtaiset ohjeet Raspberry Pi Imagerin käytöstä lähtien. Home Assistant on myös valmiina todella nopeasti sen jälkeen, kun Raspberry Pi on liitetty kotiverkkoon.

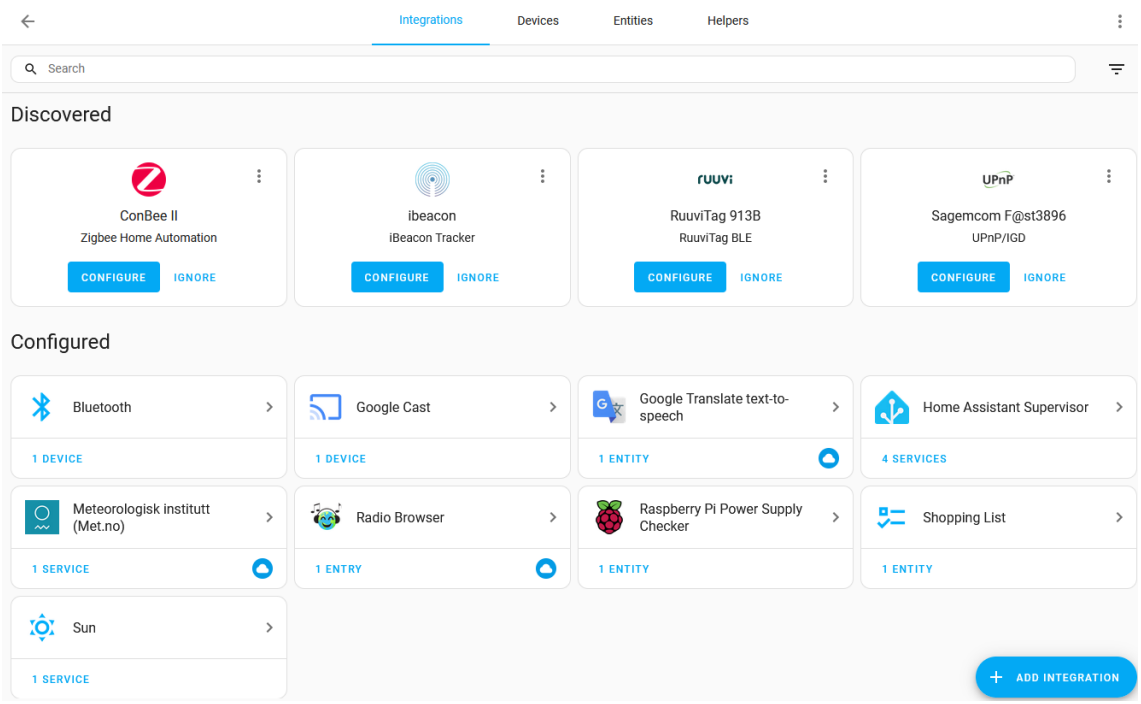
Ensimmäisenä tehtävä on luoda järjestelmälle käyttäjä. Home Assistantin kielen voi vaihtaa jo tässä vaiheessa suomeksi, mutta dokumentaation seuraaminen on helpompaa, kun jättää valinnaksi englannin. Järjestelmä kysyy nimeä, käyttäjätunnusta ja salasanaa kahteen kertaan. Nimeä kirjoittaessa järjestelmä ehdottaa käyttäjätunnukseksi nimen ensimmäistä osaa pienellä alkukirjaimella. Käyttöpaikan sijainnin voi antaa osoitteena tai valita kartalta. Käyttötarkoitus selvitetään samalla ja sijainnin jälkeen kysytään vielä maatietoa, jonka perusteella järjestelmä näyttää esimerkiksi mittayksiköt.

Seuraavassa vaiheessa kysytään tietoja, jotka käyttäjä haluaa jakaa. Neljä eri vaihtoehtoa on kaikki oletuksena valitsematta, kuten oletusarvoinen tietosuoja edellyttää. Niin järjestelmä kuin dokumentointi kertovat tarkkaan mihin tietoja käytetään.

Lopuksi Home Assistant tekee skannauksen ja näyttää mitä kaikkia signaaleja se on jo löytänyt. Dokumentaatiossa seuraava vaihe on terminologian ja konseptin lyhyt esittely.

Tehtävä 2: RuuviTagin integroiminen

Navigoimme asetusten kohtaan laitteet ja palvelut, jossa jo näkyy kuvan 6.3 mukaisesti, että Home Assistant on löytänyt RuuviTag-signaalin ja tunnistanut myös Raspberryyn liitetyn ConBee II USB-tikun.



Kuva 6.3: Home Assistant ja löydetyt integraatiot

Määrittely tapahtuu yksinkertaisesti vain painamalla `Configure` -nappia, jonka jälkeen järjestelmä kysyy tilaa/huonetta, jossa laite on ja sen jälkeen RuuviTag on yhdistetty ja eri sensoriarvot on näkyvissä ja käytettävissä.

Tehtävä 3: Zigbee USB:n integroiminen

Zigbee-yhdyskäytävän integraatio on lähes yhtä helppoa kuin RuuviTagin integraatio. `Configure` -napin jälkeen järjestelmä kysyy verkkoasetuksia radiolle. Valitsemme vaihtoehdon "Erase network settings and create a new network", koska USB-

tikulla voi olla tallennettuna verkkoasetuksia aikaisemmasta testauksesta. Konfiguraatiossa ConBee II USB-tikusta tehdään Zigbee-koordinaattori, joka koordinoi Zigbee-laitteiden viestinvälitystä käyttäen ZHA (Zigbee Home Automation) integraatiota.

Tehtävä 4: IKEA Trådfri integroiminen

Zigbee-laitteiden yhdistäminen Home Assistantissa tapahtui menemällä asetuksissa `Devices & services` -valikkoon ja siellä välilehdelle `Devices`. Siellä valitaan laite `Zigbee Coordinator`, jonka mallina näkyy ConBee II. Aukeavalla näkymällä on `Device info` -niminen laatikko, jossa on linkki "Add devices via this device", jota painamalla siirrytään sivulle, jossa Zigbee-laitteiden paritus tapahtuu.

IKEA Trådfri -laitteista kaikki muut paitsi lamppu yhdistyivät heti. Lampun ohjeissa mainittu virran katkaiseminen kuusi kertaa ei saanut sitä paritustilaan useista yrityksistä huolimatta. Puolen tunnin dokumentaation tutkimisen jälkeen paritus kuitenkin onnistui ilman, että mitään mainittavaa tehtiin eri tavalla kuin aikaisemmin.

Tehtävä 5: Lampun automaation ohjelmointi

Automaatioita pääsee luomaan valitsemalla `Automations & scenes` asetuksista. `Create automation` -nappia painamalla aukeaa ponnahdusikkuna, josta voi valita uuden automaation luomisen lisäksi valmiin suunnitelman (engl., blueprint), jonka avulla automaation luominen on helpompaa. Tarjolla on ”Motion-activated Light”-suunnitelma, jossa pitää vain valita sensoriksi liiketunnistin ja aktuaattoriksi lamppu. Suunnitelmassa on myös liukukytkin, jossa voi määritellä kuinka kauan valo palaa liiketunnistuksen jälkeen. Kuten Gladys Assistantissa, tässä asetettu aika lisätään liiketunnistimen määrittämään kolmen minuutin aikarajaan, jonka jälkeen liiketunnistin vasta muuttaa läsnäolosta ilmoittavan binääriarvon nollaksi.

Tehtävä 6: Kytkimen automaation ohjelmointi

Kytkimen automaation luominen on helppoa ilman valmista suunnitelmaa. Automaation laukaisevan tekijän jälkeen on mahdollista antaa valinnainen ehto, jonka jälkeen voi määritellä tapahtumien ketjun. Kun kytkimen I-nappia on painettu, pistorasia menee päälle, josta alkaa minuutin viive, jonka jälkeen pistorasia sammuu. Erillinen automaatio kytkimen O-napille on vielä yksinkertaisempi.

Tehtävä 7: Pistorasian automaation ohjelmointi

Pistorasian ohjelmointi onnistuu myös ilman valmista suunnitelmaa. Lämpötilan, joka on RuuviTag-laitteen entiteetti, laskiessa alle arvon 10, tarkistetaan, onko pistorasia päällä. Jos pistorasia on päällä, se kytketään pois päältä. Vastaava automaatio ylärajalle on helppo tehdä kopioimalla ensimmäinen automaatio ja muuttamalla arvoja ja tapahtumia vastaamaan tehtävän kuvausta.

Yhteenveto

Home Assistantilla kaikki kontekstissa määritellyt tehtävät onnistuivat nopeasti ja helposti. Ainoat pisteiden vähennykset liittyvät laitteistoon. Ei ole ihme, että Home Assistant on suosituin avoimen lähdekoodin kotiautomaatiojärjestelmä. Dokumentaatioon on käytetty aikaa ja siinä on otettu huomioon tietotekniikkaa vähemmän tuntevat käyttäjätkin.

Home Assistantin käyttöliittymää on helppo käyttää ja se on selkeä ja se on käännetty usealle eri kielelle. RuuviTagin ja Zigbee-yhdyskäytävän integraatio kävi sanamukaisesti napin painalluksella ja automaatioiden teko sujui ilman ohjelmointia. Järjestelmä on niin huolellisesti suunniteltu, että sitä voisi luulla kaupalliseksi järjestelmäksi.

Taulukossa 6.7 on Home Assistantin mittarin K1 tuloskortti, jossa Home Assistantin pisteiksi tuli 4,7. Käytönaikaisen laadun mittarin K2 osalta Home Assistantin pisteiksi tuli 4,9 taulukon 6.8 tuloskortin mukaisesti.

Taulukko 6.7: Mittarin K1 tuloskortti: Home Assistant

Vaatimus, ja siitä johdettu tehtävä	Ajankäyttö		Täyttyykö vaatimus?		
	aika	max	1, ei	3, osin	5, kyllä
t_1 Käyttöjärjestelmän asennus	20	120			x
t_2 RuuviTagin integroiminen	10	120			x
t_3 Zigbee USB:n integroiminen	10	120			x
t_4 IKEA Trådfri integroiminen	50	120			x
t_5 Lampun automaation ohjelmointi	10	60		x	
t_6 Kytkimen automaation ohjelmointi	10	60			x
t_7 Pistorasian automaation ohjelmointi	30	120			x

Taulukko 6.8: Mittarin K2 tuloskortti: Home Assistant

Tehtävien ja niiden suoritukseen liittyvien järjestelmän osien ja toimintojen arviointi	Arviointi					
	0	1	2	3	4	5
t_1 Käyttöjärjestelmän asennus oli helppoa						x
t_2 RuuviTagin integroiminen oli helppoa						x
t_3 Zigbee USB:n integroiminen oli helppoa						x
t_4 IKEA Trådfri integroiminen oli helppoa					x	
t_5 Lampun automaation ohjelmointi oli helppoa						x
t_6 Kytkimen automaation ohjelmointi oli helppoa						x
t_7 Pistorasian automaation ohjelmointi oli helppoa						x
v_1 Dokumentointi on kattava						x
v_2 Dokumentti on helppolukuinen						x
v_3 Käyttöliittymä on intuitiivinen						x

6.3.3 Jeedom

Tehtävä 1: Käyttöjärjestelmän asennus

Jeedomin dokumentaatiossa esitellään ensin Jeedomin oma valmis Atlas-niminen ratkaisu, jonka jälkeen on listaus Raspberry Pi:hin pohjautuvasta tee-se-itse-laitteistosta. Ennen Jeedomin asennusta pitää MicroSD-kortille asentaa Debian 11 (Bullseye) versio käyttäen Raspberry Pi Imager ohjelmaa. Dokumentin mukaan Jeedom ei ole vielä yhteensopiva Debian 12 (Bookworm) version kanssa, mutta sen eteen tehdään töitä. Jotta laitteeseen voidaan ottaa SSH-yhteys, pyydetään asennuksen jälkeen luomaan MicroSD-kortin juureen tiedosto nimeltä "ssh", sillä tietoturvasyistä Debianissa SSH-yhteys ei ole enää aktivoitu standardina. Vaikka dokumentti on englanninkielinen, on kuvakaappaukset ranskankielisestä Windows-järjestelmästä, mikä vaikeuttaa ohjeiden sisäistämistä.

Vaikka ohjeiden mukainen tiedosto on luotu, SSH-yhteyden ottaminen ei silti onnistu. Päädymme etsimään hakukoneella tietoa, ja löydämme ohjeen, jossa kerrotaan kuinka Raspberry Pi Imagerissa voi tehdä muutoksia Debian asennukseen. Asennamme käyttöjärjestelmän uudestaan niin, että siinä on SSH aktivoitu ja luomme samalla käyttäjätunnuksen ja salasanan, jolla voidaan kirjautua. Ohjeissa viitataan yhä oletuskäyttäjätunnukseen ja -salasanaan (pi/raspberry), jotka on myös

poistettu tietoturvasyistä. Luomme käyttäjän "pi", jotta mahdollisesti kovakoodatut käyttäjänimen tarkistukset eivät aiheuta ongelmia. Kun SSH-yhteys on luotu, pitää asentaa itse Jeedom dokumentissa annettujen ohjeiden mukaan. Ohjeissa sanotaan, että asennus voi kestää internet-yhteydestä riippuen jopa 90 minuuttia, mutta saamme onneksi asennuksen valmiiksi vähän yli 15 minuutissa.

Kun kirjaudumme Jeedomiin käyttäen sisäverkon IP-osoitetta, joudumme heti käyttämään käännösohjelmaa, sillä aloitussivu on ranskankielinen ja kielivalintaa ei ole tarjolla. Dokumentaation ja käännösohjelmien avulla asennus kuitenkin etenee. Kun Jeedom käyttäjä ja Jeedom Marketplace-käyttäjä on luotu ja ne on linkattu keskenään, yritämme löytää paikan vaihtaa kieliasetuksen englanniksi. Vaikka se löytyykin, on silti moni ponnahdusikkuna yhä ranskaksi, sekä myös pakolliset hyväksyntää vaativat käyttöehdot ovat ranskaksi. Jos Jeedom aikoo vakavasti tavoitella suurempaa markkina-asemaa Ranskan ulkopuolelle, käyttöehdot ja ohjeet on hyvä kääntää ainakin englanniksi. Etenemme ohjeissa ja vaihdamme järjestelmän pääkäyttäjän oletussalasanan ja luomme ensimmäisen objektin dokumentin mukaisesti. Objektin tarkoitus jää epäselväksi ja sitä ei ehdi selvittää, sillä tehtävän aikaraja tulee vastaan.

Tehtävä 2: RuuviTagin integroiminen

RuuviTagin integroimiseksi etsimme Marketplacelta hakusanalla "bluetooth" ja sieltä löytyy Bluetooth Advertisement -niminen liitännäinen (engl., plugin), jonka pitäisi olla yhteensopiva RuuviTag-sensorin kanssa [54]. Asennus sujuu helposti ja teemme liitännäisen dokumentaation mukaiset konfiguroinnit eli annamme laitteelle nimen. Luomme myös tässä vaiheessa ensimmäisen objektin, johon RuuviTag linkataan. Kojelaudalla (engl., dashboard) arvot eivät kuitenkaan päivity. Ranskankielisellä yhteisöpalstalla sanotaan, että se johtuu puuttuvasta rajapinnasta ja tarjolla on Gladys Assistantin tapaan vain tiedot läsnäolon ja signaalin voimakkuuden osalta [56].

Etsimme vielä hakusanalla "BLE" ja asennamme 4 € maksavan MQTT Discovery liitännäisen, jonka dokumentaatiossa [69] mainitaan Theengs Gateway, jonka pitäisi olla yhteensopiva RuuviTagin kanssa. Jälleen hidasteeksi muodostuu kieli. Vaikka liitännäisen sanotaan olevan ranskaksi ja englanniksi, käännös on jäänyt suurelta osin tekemättä. Tämän vuoksi tuhraantuu aikaa, ja tehtävän aikaraja tulee vastaan. Koska integrointia ei saatu onnistumaan annetussa ajassa, vaatimus ei täyty.

Tehtävä 3: Zigbee USB:n integroiminen

Etsimme Marketplacesta hakusanalla "zigbee" ja virallisia liitännäisiä löytyy JeeZigbee ja Deconz. Deconz on tehty varta vasten Phosconin ConBee Zigbee USB-tikuille. JeeZigbeen sanotaan tukevan useampaa USB-tikkua ja siinä on erikseen maininta siitä, että Deconz-liitännäistä ei tarvita sen lisäksi. Molemmat näistä liitännäisistä on maksullisia ja molempien hinta on 6 €. Koska JeeZigbeellä on parempi arvostelu, ja se on uudempi, valitsemme sen. Asennus vie Marketplacen puolelle, jossa taas on tarjolla ranskankielinen sivu ehtoineen. Maksu onnistuu luottokortilla.

JeeZigbeen dokumentaatiota tutkiessa huomaamme, että se vaatii MQTT Manager liitännäisen ja lisäksi pitää asentaa Zigbee2MQTT, johon löytyy ohjeet ohjelmiston sivuilta [108]. Zigbee2MQTT:n konfigurointitiedostoon `configuration.yaml` tulee vaihtaa mqtt-osiossa server, user ja password tiedot vastaamaan omaa asennusta:

```
mqtt:
  base_topic: zigbee2mqtt
  server: mqtt://192.168.1.100:1883
  user: jeedom
  password: abCDEfGHIjKlmNOP1qrsTUV2Wxyz34567890
```

IP-osoite on sama kuin Jeedomilla ja portti on standardina 1883. Käyttäjätiedot löytää MGTT Manager -liitännäisen `Authentication` kentästä.

Tämän jälkeen JeeZigbeen konfiguraatiossa tulee valita kohtaan `Controller port` USB-portti, jossa ConBee on tikku, sekä `Controller type` kohtaan tulee valita vaihtoehto "Deconz/Conbee". Tämän jälkeen voi aloittaa IKEA Trådfri -tuotteiden integroimisen Zigbee2MQTT-ohjelman käyttöliittymässä, johon pääsee selaimella käyttäen porttia 8080 (esimerkin tapauksessa: http://192.168.1.100:8080).

Tehtävä 4: IKEA Trådfri integroiminen

Zigbee2MQTT käyttöliittymä on tuttu jo Gladys Assistantin testistä. Myös nyt IKEA Trådfri -tuotteiden lisääminen onnistuu helposti. Laitteet näkyvät saman tien myös Jeedomissa JeeZigbeen `Equipment` -valikossa. Siellä niille voi antaa selväkieliset nimet, yhdistää asennuksen yhteydessä luotuun objektiin ja määrittää näkyväksi kojelaudalla.

Tehtävä 5: Lampun automaation ohjelmointi

Jeedomissa automaatioiden ohjelmoiminen tapahtuu `Tools` -valikon `Scenarios` -toiminnossa. Jeedomin automaatioiden ohjelmointi ei vaadi ohjelmointitaitoja ja lampun automaatio saadaan toimimaan liiketunnistimen perusteella ilman ongelmia. Lampun sammuttaminen onnistuu vasta kolmen minuutin kuluttua samasta syystä kuin Gladys Assistantin kanssa mainittiin: IKEA liiketunnistin ilmoittaa läsnäolosta binääriarvolla määritellyn ajan, joka standardina on kolme minuuttia.

Tehtävä 6: Kytkimen automaation ohjelmointi

Myös kytkimen automaation ohjelmointi sujuu ongelmitta. Minuutin aikakatkaisun ohjelmoimiseen löytyi ohjeet dokumentaatiosta ja järjestelmä ohjasi tekemistä.

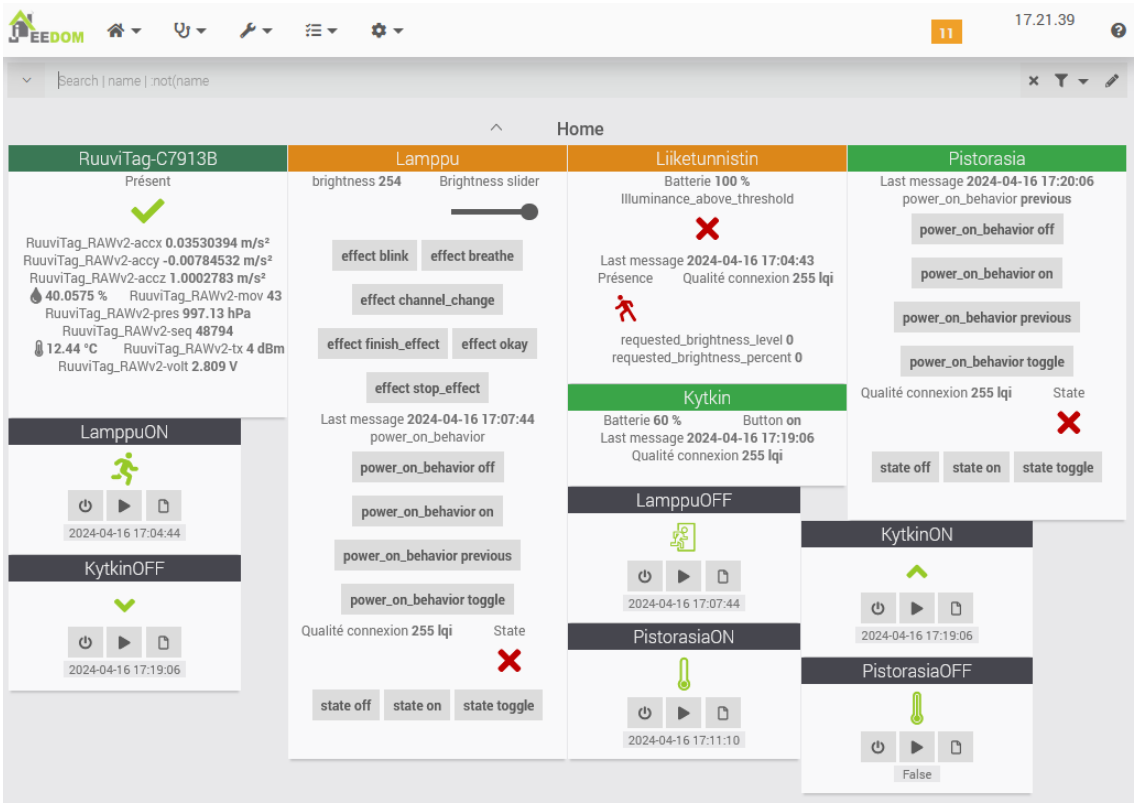
Tehtävä 7: Pistorasian automaation ohjelmointi

Pistorasian ohjelmointi jää tekemättä, koska RuuviTagin integrointi ei onnistunut määritellyssä ajassa. Myöhemmin integraatio saatiin toimimaan asentamalla vielä Theengs Gateway [102] Jeedomin kanssa samalle laitteelle. Integraatio oli kuitenkin niin hankala toteuttaa, ettei sitä voi suositella.

Yhteenveto

Jeedom oli turhauttava kokemus suurimmalta osin vajavaisen käännöstyön takia. Myös asennusohjeissa ollut vanhentunut tieto hidastutti asennusta. Laitteiden integroimiseen tarvittiin maksullisia lisäosia, joissa toistui samat ongelmat kuin järjestelmässäkin. Lopulta kun kaikki osat olivat paikallaan, automaatioiden tekeminen oli yksinkertaista. Jos Jeedom on matkalla kaupalliseksi järjestelmäksi, vaatii se vielä paljon parantamista. Jeedomissa kuvan 6.4 mukainen kojelauta rakentui itsestään laitteiden ja automaatioiden lisäyksen yhteydessä. Käyttöliittymään heijastui myös kieliongelmat.

Taulukossa 6.9 on Jeedomin mittarin K1 tuloskortti. Mittarin K1 tulokseksi Jeedom sai 3,6. Käytönaikaisen laadun osalta Jeedomin jäi 2,9 pisteeseen, pisteiden jakautuessa taulukon 6.6 mukaisesti.



Kuva 6.4: Jeedomin kojelauta

Taulukko 6.9: Mittarin K1 tuloskortti: Jeedom

Vaatimus, ja siitä johdettu tehtävä	Ajankäyttö		Täyttyykö vaatimus?		
	aika	max	1, ei	3, osin	5, kyllä
t_1 Käyttöjärjestelmän asennus	120	120			x
t_2 RuuviTagin integroiminen	>120	120	x		
t_3 Zigbee USB:n integroiminen	120	120			x
t_4 IKEA Trådfri integroiminen	20	120			x
t_5 Lampun automaation ohjelmointi	30	60		x	
t_6 Kytkimen automaation ohjelmointi	20	60			x
t_7 Pistorasian automaation ohjelmointi	>120	120	x		

Taulukko 6.10: Mittarin K2 tuloskortti: Jeedom

Tehtävien ja niiden suoritukseen liittyvien järjestelmän osien ja toimintojen arviointi	Arviointi					
	0	1	2	3	4	5
t_1 Käyttöjärjestelmän asennus oli helppoa						x
t_2 RuuviTagin integroiminen oli helppoa		x				
t_3 Zigbee USB:n integroiminen oli helppoa		x				
t_4 IKEA Trådfri integroiminen oli helppoa						x
t_5 Lampun automaation ohjelmointi oli helppoa						x
t_6 Kytkimen automaation ohjelmointi oli helppoa						x
t_7 Pistorasian automaation ohjelmointi oli helppoa	x					
v_1 Dokumentointi on kattava				x		
v_2 Dokumentti on helppolukuinen		x				
v_3 Käyttöliittymä on intuitiivinen				x		

6.3.4 OpenHAB

Tehtävä 1: Käyttöjärjestelmän asennus

OpenHABin dokumentaatio alkaa lyhyellä selvityksellä järjestelmän terminologiasta ja konfiguraatioista. Järjestelmän voi asentaa tiedostopohjaisena tai käyttöliittymäpohjaisena. Dokumentaatio on vain käyttöliittymäpohjaiselle versiolle, joten asennamme ja konfiguroimme OpenHABin käyttämään käyttöliittymää. Asennus onnistuu helposti Raspberry Pi Imagerin avulla. Kun muistikortti on laitettu Raspberryyn ja tietokone on käynnistetty, kestää pitkään, ennen kuin dokumentissa kerrotusta osoitteesta `http://openhabian:8080` aukeaa käyttöliittymä.

Asennus alkaa pääkäyttäjän luomisella. Pääkäyttäjälle pyydetään antamaan käyttäjätunnus ja syöttämään haluttu salasana kahdesti. Sen jälkeen määritellään järjestelmälle kieli ja annetaan muut alueasetuksen ja aikavyöhyke. Seuraavaksi järjestelmä kysyy, halutaanko kodin sijainti jakaa. Sijainti mahdollistaa auringonnousun ja -laskun, sekä sään kaltaisten sijainnista riippuvaisten tietojen hyödyntämisen. Sijainnin voi määritellä antamalla koordinaatit, merkitsemällä kodin kartalta tai käyttämällä laitteen sijaintia. Lopuksi järjestelmä ehdottaa vielä neljän lisäosan asentamista. Niiden osalta on linkki dokumentaatioon.

Tehtävä 2: RuuviTagin integroiminen

Aloitamme tutkimaan `Add-On Store` valikkoa. Koska RuuviTag käyttää Bluetooth-teknologiaa, asennamme Bluetooth Binding -lisäosan. Tämän jälkeen navigoidaan asetuksissa olevaan `Things` valikkoon, jossa on postilaatikko, jossa odottaa tehtävä asentaa Bluetooth-rajapinta (`Bluetooth Interface`).

Oikeassa alareunassa on sininen ympyrä, jonka sisällä on plusmerkki. Sitä painamalla voi asentaa lisää laitteiden ilmentymiä. Ensimmäiseksi pitää valita mihin sidokseen laite kytkeytyy, eli valitsemme Bluetooth-sidoksen. Sieltä löytyy `Scan` -nappi, joka löytää toistakymmentä Bluetooth-laitetta, muttei RuuviTagia. Teemme myöhemmin uuden skannauksen ja sen jälkeen listalla on yli 50 Bluetooth-signaalia, mukaan lukien RuuviTag, joka on tunnistettu MAC-osoitteen perusteella, samoin kuin useimmat muut löytyneistä radioista.

Hetkeen päästä RuuviTag häviää skannaustuloksista ja jäljellä on vain puolet alkuperäisistä laitteista. Vaikka teemme useita uusintaskannauksia listalla näkyy vain toistakymmentä geneeristä signaalia. Sivun lopussa on kohta `RuuviTag Smart Beacon`, jossa voi asentaa RuuviTagin manuaalisesti. Sen tehtyämme valitsemme `Things` -sivulta ja navigoimme `Channels` -välilehdelle, jossa luodaan lämpötilalle kanava ilmentymän ja itse laitteen välille. Kanavaa luotaessa järjestelmä ehdottaa myös itse laitteen (`Item`) luomista. Monikäyttöisen RuuviTag-sensorin osalta `Item` tulee luoda jokaiselle sensorille erikseen. Koska aikaa on kulunut jo paljon, joudumme vain luomaan lämpötilalle kanavan ja laitteen. Olisi ollut mielenkiintoista nähdä miten ohjattu RuuviTag luominen onnistuu.

Tehtävä 3: Zigbee USB:n integroiminen

`Add-on Store` -osiosta löytyy Zigbeelle oma sidos `ZigBee Binding`. Sen asetusten kautta ei selviä miten ConBee II -tikku saataisiin asennettua. Dokumentaatiosta löytyy syy, `ZigBee Binding` ei tue ConBee II -tikkua. Kun etsimme dokumentaatiosta hakusanalla "conbee", löytyy lisäosa nimeltä `deCONZ Binding`. Bluetooth-lisäosan asennuksen kaavaa noudattaen asennetaan seuraavaksi Zigbee-rajapinta, jonka jälkeen siirrytään `Things` -valikon sinisen plusmerkin kautta konfiguroimaan laitetta. Konfiguraatiossa pyydetään deCONZ-ohjelmiston IP-osoitetta ja porttia, eikä USB-porttia. Dokumentaatiosta selviää, että deCONZ-sidos vaatii, että Raspberry Pi:lle tai muulle verkossa olevalle laitteelle tulee asentaa erillinen yhdyskäytävä, johon löytyy ohjeet ConBeen valmistajalta [80].

Päätämme asentaa yhdyskäytävän samalle Raspberry Pi:lle, jossa on OpenHAB, koska käytössä ei ole muita laitteita. Dokumentaatiosta [75] löytyy oletuskäyttäjä ja -salasana, joiden avulla voi kirjautua OpenHabiin SSH-yhteydellä. Koska deCONZ-yhdyskäytävän ohjelmisto käyttää samaa porttia kuin OpenHAB, se tulee konfiguroida käyttämään eri porttia. Yhteisöpalstan ohjeiden [77] avulla deCONZ-yhdyskäytävän portiksi vaihdetaan 8081.

Lopuksi deCONZ-yhdyskäytävä pitää konfiguroida sen omassa käyttöliittymässä luomalla salasana. OpenHABin puolella deCONZ-sidoksen asetuksissa pitää yhdistää sidos ja deCONZ-yhdyskäytävä. Jotta ne voivat yhdistyä, täytyy yhdyskäytävän asetuksissa painaa `Authenticate app` -nappia, ennen kuin tallentaa sidoksessa yhdyskäytävän IP-osoitteen ja portin. Lopulta OpenHAB näyttää löytävän yhdyskäytävän ja yhdistäminen onnistuu.

Tehtävä 4: IKEA Trådfri integroiminen

IKEA Trådfri -tuotteet pitää ensin parittaa deCONZ-yhdyskäytävässä, mikä onnistuu helposti, mutta tuotteet eivät kuitenkaan ilmesty OpenHABiin. Dokumentaation ja yhteisöpalstan avulla selviää, että deCONZ-yhdyskäytävän lisäksi on OpenHABiin asennettava deCONZ-siltaus. Senkään asennus ei heti auttanut. Vasta kun siltauksessakin vaihtaa portiksi deCONZ-yhdyskäytävän käyttämän 8081, siltaus näkyy online-tilassa. IKEA Trådfri -tuotteet pitää sen jälkeen asentaa uudestaan ennen kuin ne näkyvät OpenHABissa. Yksi yhtenäinen ohje olisi säästänyt paljon aikaa tässäkin vaiheessa.

Tehtävä 5: Lampun automaation ohjelmointi

Lampun automaation ohjelmointi ei heti onnistu OpenHABissa, joten ajan säästämiseksi tutkimme, onnistuisiko se deCONZ-yhdyskäytävä puolella. Siellä automaation ohjelmointi tapahtuu helposti. Vaikka yhdyskäytävän ohjelmoinnissa on mahdollisuus antaa kaksi toimintoa ja määritellä niiden välillä oleva aika, 30 sekunnin väliaika sammutuksen kanssa ei toimi, vaan aikakatkaisu tapahtuu kolmen minuutin ja 30 sekunnin päästä eli tämä arvo lisätään IKEA liiketunnistimen kolmen minuutin "occupancy_timeout" arvon päälle. Kun yritämme hakea apua dokumentaatiosta ja yhteisöpalstalta, ohjeet viittaavat usein tiedostopohjaisen version ohjeisiin ja sisältävät YAML-merkintäkieltä.

Tehtävä 6: Kytkimen automaation ohjelmointi

Myös kytkimen ohjelmointi OpenHABissa ei meinaa onnistua, joten päädymme senkin osalta ohjelmoimaan automaation yhdyskäytävän puolella. Liitetyn kytkimen löytää deCONZ-yhdyskäytävän laiteryhmän `Switch Editor` -kohdasta, jossa voi määritellä mitä tapahtuu kytkimen eri napeista, mutta siellä ei voi määritellä mitään raja-aikaa tai edes eri toimintoja pitkälle napin painallukselle. `Automation` kohta, missä lampun automaatio toteutettiin, tarjoaa laukaisevaksi laitteeksi vain liiketunnistinta, joten siellä ei voi käyttää kytkintä. Myös tämän automaation ohjelmoinnissa tulee aikaraja vastaan ja joudumme tyytymään puolittaiseen ratkaisuun.

Tehtävä 7: Pistorasian automaation ohjelmointi

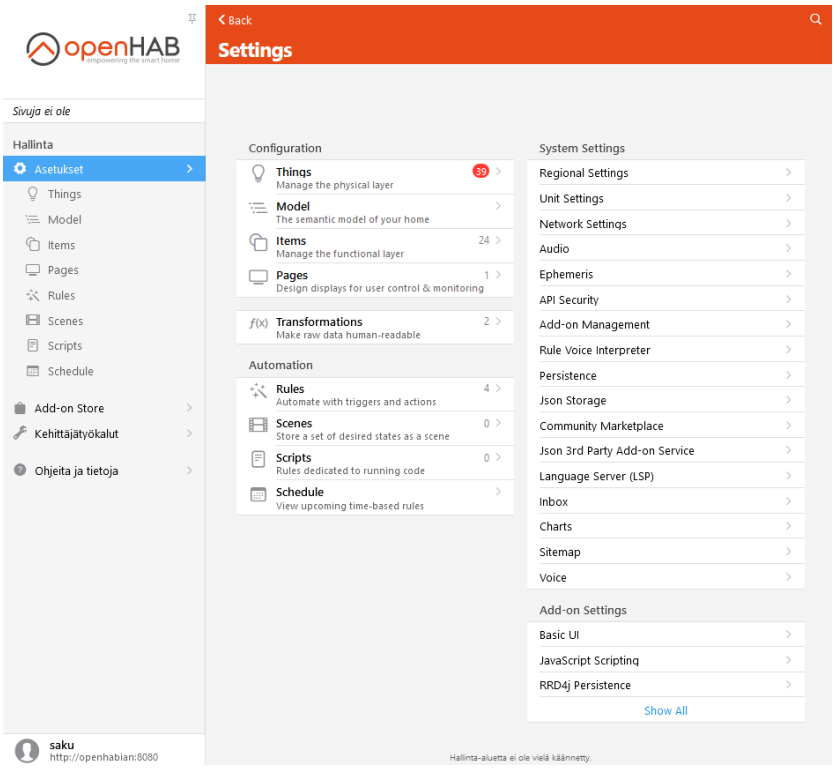
Koska pistorasian ohjelmoinnissa tarvitaan RuuviTagia, on se tehtävä OpenHABin puolella. Emme saa selkeää käsitystä siitä, miksi IKEA Trådfri -tuotteet eivät reagoi OpenHABin puolella tehtyihin automaatioihin. Dokumentaatio on selkeä ja automaatioita testaamalla toimilaitteet saadaan kytkeytymään päälle ja pois, mutta automaation laukaisu liiketunnistimen, kytkimen napin tai lämpötilan muutoksen perusteella ei toimi. Yhteisöpalstalla viitataan aina tiedostopohjaiseen konfigurointiin ja YAML-merkintäkieleen. Kahden tunnin jälkeen tehtävä jää kesken.

Yhteenveto

OpenHABin helpon asennuksen jälkeen seuraavat tehtävät vievät kaikki maksimiajan. Vaikka RuuviTagille oli olemassa automaattinen asennus, RuuviTag jostain syystä hävisi skannaustuloksista. Laboratorion käyttämä Phoscon ConBee II -tikku vaati OpenHABissa erillisen yhdyskäytävän asentamisen, joka osoittautui vaativaksi tehtäväksi, mutta se saatiin tehtyä juuri ja juuri aikarajoituksen puitteissa.

IKEA tuotteiden liittäminen ei myöskään ollut yksinkertaista, joka saattoi johtaa siihen, ettei tuotteet toimineet OpenHABin automaatioissa. Automaatioista lampun automaatio ja kytkimen automaatio saatiin kuitenkin osittain tehtyä deCONZ-yhdyskäytävän puolella. OpenHAB dokumentaatio on erittäin kattava ja helppolukuinen. Kuvassa 6.5 esitetyn käyttöliittymän navigoinnissa ei ollut ongelmia, mutta erilaiset konfiguroinnit sekoittavat helposti aloittelevan käyttäjän ajatukset.

Taulukoissa 6.11 ja 6.12 on OpenHABin mittareiden K1 ja K2 tuloskortit. OpenHABin pisteiksi tuli K1-mittarilla 3,9 ja K2-mittarilla 2,5.



Kuva 6.5: OpenHABin konfigurointi asetukset

Taulukko 6.11: Mittarin K1 tuloskortti: OpenHAB

Vaatimus, ja siitä johdettu tehtävä	Ajankäyttö		Täyttyykö vaatimus?		
	aika	max	1, ei	3, osin	5, kyllä
t_1 Käyttöjärjestelmän asennus	40	120			x
t_2 RuuviTagin integroiminen	120	120			x
t_3 Zigbee USB:n integroiminen	120	120			x
t_4 IKEA Trådfri integroiminen	120	120			x
t_5 Lampun automaation ohjelmointi	60	60		x	
t_6 Kytkimen automaation ohjelmointi	60	60		x	
t_7 Pistorasian automaation ohjelmointi	>120	120	x		

Taulukko 6.12: Mittarin K2 tuloskortti: OpenHAB

Tehtävien ja niiden suoritukseen liittyvien järjestelmän osien ja toimintojen arviointi	Arviointi					
	0	1	2	3	4	5
t_1 Käyttöjärjestelmän asennus oli helppoa						x
t_2 RuuviTagin integroiminen oli helppoa			x			
t_3 Zigbee USB:n integroiminen oli helppoa		x				
t_4 IKEA Trådfri integroiminen oli helppoa				x		
t_5 Lampun automaation ohjelmointi oli helppoa		x				
t_6 Kytkimen automaation ohjelmointi oli helppoa		x				
t_7 Pistorasian automaation ohjelmointi oli helppoa	x					
v_1 Dokumentointi on kattava						x
v_2 Dokumentti on helppolukuinen						x
v_3 Käyttöliittymä on intuitiivinen			x			

6.4 Mittausalue T

Gladys Assistantin ja OpenHABin sivustoilta ei löydy tietoturvapolitiikka, mutta molempien GitHub sivustolla (Gladys Assistantin tietovarastossa /Gladys ja OpenHABin tietovarastossa /.github) on tiedosto SECURITY.md, joissa on vain lyhyt ohje haavoittuvuuksien ilmoittamiseen. Vaikka ohjeet haavoittuvuuksien raportointiin löytyy, T1-mittarin kahden ensimmäisen vaatimuksen toteutuminen jää hyvin vajavaiseksi. Jeedomilla ei ole minkäänlaista tietoturvapolitiikkaa. GitHubissa SECURITY.md tiedostoa ei ole, eikä ohjeita haavoittuvuuksien raportointiin löydy mistään.

Gladys Assistantille, Jeedomille ja OpenHABille kaikille löytyy kaksi CVE haavoittuvuutta (Gladys Assistant: CVE-2023-43256 ja CVE-2023-47440, Jeedom: CVE-2020-9036 ja CVE-2021-42557 sekä OpenHAB: CVE-2020-5242 ja CVE-2021-21266), joiden perusteella ne osallistuvat CVE raportointiohjelmaan. Minkään edellä mainitun järjestelmän kotisivuilta tai GitHubista ei löytynyt tietoa siitä, missä versiossa näihin haavoittuvuuksiin on tullut mahdollisesti korjaus, joten mittarin T1 vaatimus neljä ei toteudu. Mittarin T1 pisteet Gladys Assistantille, Jeedomille ja OpenHABille jäävät siten 2,0 pisteeseen.

Gladys Assistant ja Jeedom kysyvät nimitietoja ja sähköpostiosoitetta pakollisina tietoina järjestelmän pääkäyttäjää luodessa. Vaikka Gladys Assistantissa tarkoi-

tuksena on luultavasti vain tarjota tapa palauttaa unohtunut salasana, olisi toivottavaa, että näiden tietojen syöttäminen on vapaaehtoista ja syy niiden pyytämiselle kerrottaisiin. Jeedomissa samoja tietoja kysytään luotaessa käyttäjää Marketplacen puolelle. Oletusasetusten osalta Gladys Assistant saa täydet pisteet, mutta Jeedom vain kolme pistettä, koska Jeedomilta alkaa myös tulemaan roskapostiksi luokiteltua mainospostia, minkä kieltäminen ei ollut mahdollista käyttäjää luodessa.

Home Assistant ja OpenHAB eivät kysy mitään ylimääräisiä tietoja. Yksityisyyteen liittyvien tietojen, kuten sijainnin määrittely on vapaaehtoista ja asennuksen yhteydessä selitetään mihin tietojen pyytäminen perustuu. Home Assistantilla on myös erillinen yksityisyyttä koskeva sivusto, jossa käydään läpi mitä tietoja he keräävät ja mihin niitä käytetään. Home Assistant ja OpenHAB molemmat saavat mittarista T2 täydet 5,0 pistettä.

Home Assistantilta löytyy kattava tietoturvaa käsittelevä sivusto, jossa luetellaan kaikki CVE haavoittuvuudet ja missä versiossa niihin on tehty korjaus. Yhteensä Home Assistantille löytyy 14 CVE raportoitua haavoittuvuutta, joista 10 on viime vuodelta. Samalla sivulla on ohjeet haavoittuvuuksien raportointiin. Home Assistant ostaa tietoturva-auditointeja myös palveluina ja raportoi niiden löydöksistä avoimesti [45]. Home Assistant saa ainoana T1-mittarista täydet 5,0 pistettä. Taulukossa 6.13 on esitetty mittausalueen T tulokset kootusti.

Taulukko 6.13: Vertailtavien kotiautomaatiojärjestelmien mittareiden T1 ja T2 mittaustulokset

	Gladys Assistant	HA OS	Jeedom	OpenHAB
T1, v_1: Tietoturvapolitiikka on olemassa ja se on helposti löydettävissä	1	5	1	1
T1, v_2: Tietoturvapolitiikka on kattava ja selkeä, ja se sisältää ohjeet haavoittuvuuksien raportointiin	1	5	1	1
T1, v_3: Ohjelmisto osallistuu johonkin CVE raportointiohjelmaan	5	5	5	5
T1, v_4: Raportoidut haavoittuvuudet ja korjaukset niihin on helposti löydettävissä	1	5	1	1
T1 pisteet	2,0	5,0	2,0	2,0
T2, v_1: Järjestelmä kysyy vain olennaisia tietoja tai tietojen syöttäminen on vapaaehtoista	3	5	3	5
T2, v_2: Järjestelmän oletusasetukset ovat tietosuojan kannalta tiukimmat	5	5	3	5
T2 pisteet	4,0	5,0	3,0	5,0

6.5 Testitulosten yhteenveto

Tässä testissä testattiin neljä avoimen lähdekoodin kotiautomaatiojärjestelmää (Gladys Assistant, Home Assistant, Jeedom ja OpenHAB) käyttäen mittaristona luvussa 5 luotua mittaristoa, joka jakautuu kolmeen eri mittausalueeseen, joista jokaisessa on 2-3 mittaria.

Mittausalue Y mittasi yhteisön kasvua (mittari Y1) ja aktiivisuutta (mittari Y2), sekä projektin kiinnostavuutta ja suosiota (mittari Y3). Mittausalue K koostui kahdesta mittarista, joista ensimmäinen (mittari K1) mittasi käyttöarvoa ja toinen (mittari K2) käytönaikaista laatua. Turvallisuus (mittari T1) ja yksityisyys (mittari T2) muodistivat mittausalueen T. Mittaustulokset mittareittain on esitetty taulukossa 6.14. Lisäksi on laskettu kaikkien mittaustulosten keskiarvo.

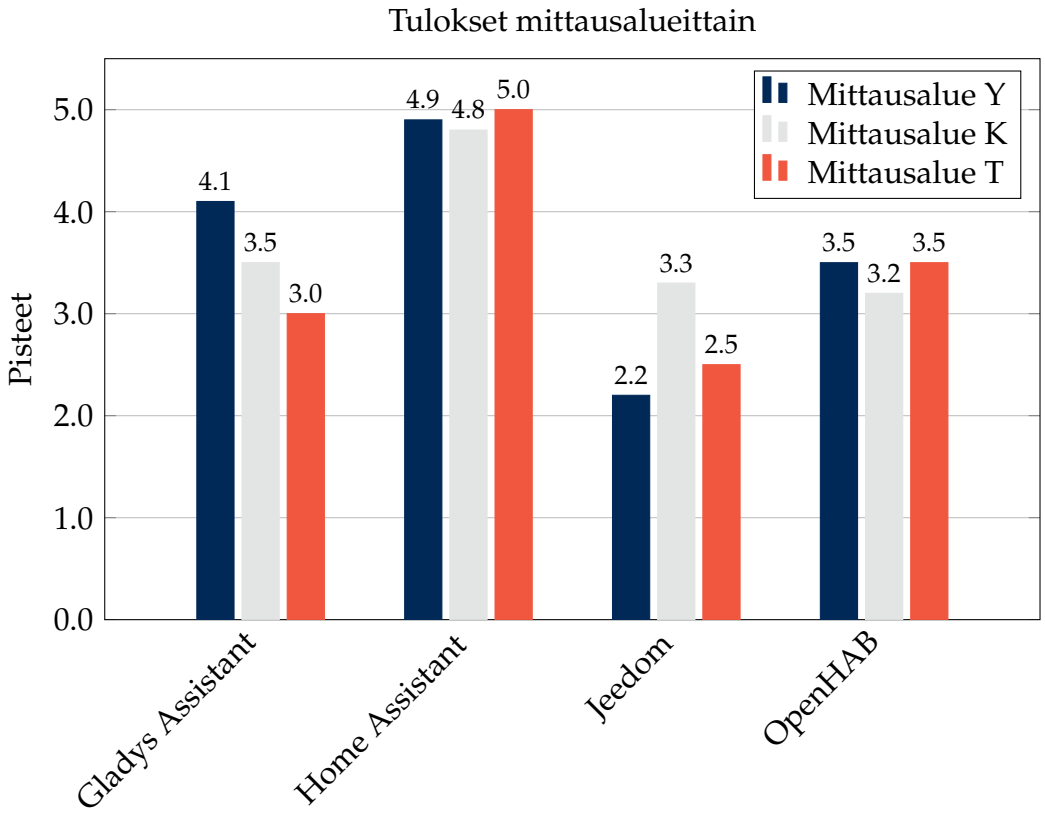
Taulukko 6.14: Mittaustulosten yhteenveto

	Gladys Assistant	Home Assistant	Jeedom	OpenHAB
Y1: Kasvava yhteisö	5,0	5,0	0,0	5,0
Y2: Aktiivinen yhteisö	4,0	5,0	4,0	2,1
Y3: Projektin kiinnostavuus	3,4	4,8	2,6	3,5
K1: Käyttöarvo	3,6	4,7	3,6	3,9
K2: Käytönaikainen laatu	3,3	4,9	2,9	2,5
T1: Tietoturva	2,0	5,0	2,0	2,0
T2: Yksityisyys	4,0	5,0	3,0	5,0
Yhteensä	**3,6**	**4,9**	**2,6**	**3,4**

Parhaimman tuloksen sai Home Assistant keskiarvolla 4,9. Vertailun toiseksi parhaaksi keskiarvolla 3,6 yltänyt Gladys Assistant on vertailtavista järjestelmistä selkeästi pienin, mutta kiinnostus siihen on kasvussa englanninkielisen yhteisöpalstan perustamisen myötä. Se pystyi jättämään taakseen OpenHABin, joka sai keskiarvoksi 3,4. OpenHAB on usein esitetty varteenotettavimpana kilpailijana Home Assistantille, mutta se ei sovellu aloittelijoille. OpenHABin kompastukseksi koituikin etenkin käytönaikainen laatu. OpenHAB on myös pyrkinyt luomaan maineen turvallisuutta ja yksityisyyttä vaalivana järjestelmänä, joten turvallisuuspolitiikan puuttuminen jäi ihmetyttämään.

Home Assistant on kasvanut jo niin suureksi, että sen taustalla on neljän miljoonan dollarin liikevaihtoa pyörittävä yritys Nabu Casa, jonka työntekijöiksi kehittäjien ydintiimi on siirtynyt. Home Assistant on kaikilla osa-alueilla paras ja etenkin dokumentoinnin ja käytönaikaisen laadun osalta se on ylivoimainen. Ammattimainen suhtautuminen turvallisuuteen ja yksityisyyteen toi täydet pisteet mittausalueelta T. Eri mittausalueiden mittareiden keskiarvot järjestelmittäin on esitetty kuvassa 6.6.

Vertailun viimeiseksi keskiarvolla 2,6 jäi Jeedom, jonka kansainvälistymisen ja kasvun esteenä on pitäytyminen ranskan kielessä. Jeedomilla ei ole erillistä englanninkielistä yhteisöpalstaa ja vaikka ohjelmiston asetuksista löytyy myös englanti, käännöstyö on jäänyt pahasti kesken.



Kuva 6.6: Mittausalueiden keskiarvot järjestelmittäin

7 Yhteenveto ja johtopäätökset

Tässä pro gradu -tutkielmassa arvioitiin avoimen lähdekoodin kotiautomaatiojärjestelmien valmiutta vastata kuluttajien tarpeisiin. Avoimen lähdekoodin kotiautomaatiojärjestelmiä on useita, joten sopivan järjestelmän valinta voi olla vaikeaa. Vaikka kotiautomaatiojärjestelmien historia alkaa jo 1960-luvulta, on suurimpana esteenä niiden leviämiselle pidetty kuluttajien huolta järjestelmien turvallisuudesta ja omasta yksityisyydestään. Yksityisyyden suojaa ei ollut painotettu aikaisemmissa avoimen lähdekoodin kotiautomaatiojärjestelmien vertailuissa.

Tutkimusmenetelmänä käytetty konstruktiivinen tutkimusote pyrkii löytämään ratkaisun käytännön ongelmaan tai kysymykseen tuottamalla konstruktion, joka on innovatiivinen ja perustuu olemassa olevaan teoriaan [65]. Tutkielman alussa käytiin läpi kotiautomaatiojärjestelmien ominaispiirteitä ja historiaa, sekä avoimen lähdekoodin ohjelmistojen etuja. Ennen varsinaiseen konstruktioon siirtymistä käsiteltiin myös sitä, miten ohjelmistojen laadun mittaaminen on muuttunut viime vuosikymmenten aikana kohti arvosuuntautunutta ajattelua.

Tässä tutkimuksessa konstruktio oli mittaristo avoimen lähdekoodin kotiautomaatiojärjestelmien vertailuun. Avoimen lähdekoodin ohjelmistojen mittaukseen ei ole olemassa yhtä hyväksyttyä viitekehystä tai mallia [106], joten useimmiten mittaristot, kuten myös tämä mittaristo, suunnitellaan tiettyyn tarkoitukseen. Mittaristossa pyrittiin kuitenkin käyttämään hyödyksi aikaisemmin luotuja mittareita.

Avoimen lähdekoodin ohjelmistojen etuna on turvallisuuden kannalta läpinäkyvyys ja luotettavuus [36]. Kehittäjien ja järjestelmän käyttäjien yhteinen arvonluonti näkyy selvimmin yhteisöjen keskustelualueilla, joten yhteisö ja etenkin sen kasvu ja aktiivisuus ovat tärkeitä avoimen lähdekoodin ohjelmistojen kehitykselle. Yhteisön merkitys oli nähtävissä mittaustuloksista, sillä kahden mittauksissa parhaiten menestyneen järjestelmän yhteisöt olivat aktiivisia ja kasvoivat nopeiten.

Tutkimuksessa luodussa mittaristossa painotettiin yhteisöissä tapahtuvan arvonluomisen, tietoturvan ja yksityisyyden lisäksi ominaisuuksia, joita kuluttajat odottavat kotiautomaatiojärjestelmiltä. Tämä tieto perustui Schomakers et al. [92] tutkimukseen, jonka perusteella muodostettiin konteksti käyttöarvon ja käytönaikaisen laadun mittaamiselle. Käytönaikaisessa laadussa pyrittiin painottamaan helppout-

ta, koska tämän tutkimuksen tarkoituksena oli löytää järjestelmä, jonka asennus ja konfigurointi onnistuisi myös tietotekniikkaa vähemmän tuntevalta käyttäjältä. Borissova et al. [10] tutkimuksessa, jossa OpenHAB sijoittui parhaaksi, arviointi perustui osaavan asiantuntijan antamiin pisteisiin ja painotuksiin. Setz et al. [94] laajassa vertailussa Home Assistant menestyi parhaiten OpenHABin tullessa toiseksi. Kummankaan edellä mainitun tutkimuksen otokseen ei kuulunut Gladys Assistant eikä Jeedom.

Konstruktiivisessa tutkimusotteessa painotetaan konstruktion testaamista käytännössä [81]. Mittaristo testattiin tutkimalla neljän avoimen lähdekoodin kotiautomaatiojärjestelmän yhteisöä ja niiden dokumentaatiota ja tietoturvapolitiikkaa. Empiirisessä osuudessa valitut järjestelmät asennettiin yhden piirilevyn tietokoneelle käyttöarvon, käytönaikaisen laadun ja yksityisyyden suojan arvioimiseksi. Käyttöarvon mittaukseen vaikutti myös laboratorion laitteisto, joka yritettiin koota laitteista, jotka ovat laajasti saatavilla ja yhteensopivia mahdollisimman monen järjestelmän kanssa. Kaikkien järjestelmien dokumentaation alussa pyrittiin kiinnittämään lukijan huomio sensorien ja aktuaattorien valintaan. Kotiautomaatiossa on käytössä myös useita eri teknologioita, joista Matter/Thread on uusin.

Kaikkein selkein osa-alue, jossa vertailun voittaja erottui, oli tietoturva. Morrison et al. [70] löysivät satoja uniikkeja tietoturvamittareita systemaattisessa kirjallisuuskartoituksessaan. Tämän tutkielman mittaristossa pyrittiin arvioimaan järjestelmän ja sitä kehittävän yhteisön suhtautumista tietoturvaan heidän tietoturvapolitiikan ja avoimuuden perusteella. Home Assistant oli ainoa järjestelmä, jolla oli selkeä ja kattava tietoturvapolitiikka. Vaikka sille oli eniten julkaistuja haavoittuvuuksia, jokaiseen haavoittuvuuteen oli julkaistu korjaus ja korjauksen sisältävä versiotaso oli selvästä löydettävissä.

Testauksen luotettavuuden kannalta olisi ollut hyvä tehdä testaus useamman järjestelmän osalta ja teettää mittausalueen K testit henkilöllä, jolla ei ole kokemusta kotiautomaatiosta. Se ei kuitenkaan ollut mahdollista tutkimuksen laajuuden puitteissa, joten testauksessa pyrittiin asettumaan kotiautomaatiosta kiinnostuneen, mutta tietotekniikka vain vähän tuntevan henkilön asemaan ja käyttämään testattaessa hyväksi vain järjestelmän dokumentaatiota ja yhteisöpalstaa.

Mittaustulosten perusteella erottui selkeästi yksi kaikilla mittausalueilla pärjäävä ja vaatimuksiin sopiva järjestelmä, Home Assistant, jossa järjestelmän asentaminen, sensorien ja aktuaattoreiden integrointi ja automaatioiden luominen onnistui seuraamalla järjestelmän dokumentaatiota. Se sai pisteitä 4,9. Toiseksi tulleen Gla-

dys Assistantin pieni kehittäjäyhteisö on alkanut laajentua englanninkielisen yhteisöpalstan avauduttua. Se sai vertailussa 3,6 pistettä jättäen taakseen OpenHAB-in, joka sai 3,4 pistettä. Neljäs testatuista järjestelmistä, Jeedom, jäi 2,6 pisteeseen. Vastauksena tutkimuskysymykseen voidaankin todeta, että tutkimuksessa luodulla mittaristolla voidaan laittaa järjestelmät paremmuusjärjestykseen ja löytää avoimen lähdekoodin kotiautomaatiojärjestelmä, jolla on aktiivinen ja kasvava yhteisö, joka on helppokäyttöinen ja vastaa kuluttajien vaatimuksia, sekä ottaa turvallisuuden ja yksityisyydensuojan huomioon.

Mielenkiintoisia jatkokehitysideoita heräsi tutkielman aikana useita. Esimerkiksi Borissova et al. [10] käyttämän MAUT-pohjaisen mallin yhdistäminen mittaristoon mahdollistaisi eri mittausalueiden painotuksen. Jatkotutkimusta kaipaa myös ohjelmistojen tietoturvan vertailukelpoinen mittaaminen. Siihen on ehdotettu useita eri tapoja, mutta yhtä systemaattista, yleisesti hyväksyttyä tapaa ei ole olemassa.

Lähteet

[1] ABBAS, T., KHAN, V.-J., JA MARKOPOULOS, P. Investigating the Crowds Creativity for Creating On-Demand IoT Scenarios. *International Journal of HumanComputer Interaction 36*, 11 (2020), 1022–1049.

[2] ABRAN, A. *Software Metrics and Software Metrology*. John Wiley Sons, Inc. IEEE Computer Society Press, New Jersey, 2010.

[3] ADEWUMI, A., MISRA, S., OMOREGBE, N., JA SANZ, L. F. FOSSES: Framework for open-source software evaluation and selection. *Software: Practice and Experience 49*, 5 (2019), 780–812.

[4] AHMAD, N., JA LAPLANTE, P. A. A systematic approach to evaluating open source software. Kirjassa *Strategic Adoption of Technological Innovations*. IGI Global, 2013, ss. 50–69.

[5] ALJERAISY, A., BARATI, M., RANA, O., JA PERERA, C. Privacy Laws and Privacy by Design Schemes for the Internet of Things: A Developers Perspective. *ACM Computing Surveys 54*, 5 (2021), 1–38.

[6] AMARO, R., PEREIRA, R., JA DA SILVA, M. M. Capabilities and metrics in DevOps: A design science study. *Information Management 60*, 5 (2023), 103809.

[7] ASHTON, K. That internet of things thing. *RFID journal 22*, 7 (2009), 97–114.

[8] BIRD, C., NAGAPPAN, N., MURPHY, B., GALL, H., JA DEVANBU, P. Don't Touch My Code! Examining the Effects of Ownership on Software Quality. Julkaisusarjassa *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering* (New York, NY, USA, 2011), Association for Computing Machinery, 4–14.

[9] BORGES, H., JA VALENTE, M. T. Whats in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform. *Journal of Systems and Software 146* (2018), 112–129.

[10] BORISSOVA, D., DANEV, V., GARVANOVA, M., RADOSLAV, Y., JA GARVANOV, I. Identification of the Important Parameters for Ranking of Open-Source Home Automation Platforms for IoT Management. Julkaisusarjassa *Advances in Systems Engineering* (Cham, 2022), Springer International Publishing, 310–319.

[11] BRETTHAUER, D. Open source software: A history. *Information Technology and Libraries 21*, 1 (maaliskuu 2002), 3–10.

[12] CONNECTIVITY STANDARDS ALLIANCE. Matter Arrives Bringing A More Interoperable, Simple And Secure Internet Of Things to Life. URL `https://csa-iot.org/newsroom/matter-arrives/`, viitattu 29.9.2023.

[13] CROWD SUPPLY. Home Assistant Yellow. URL `https://www.crowdsupply.com/nabu-casa/home-assistant-yellow`, viitattu 22.1.2024.

[14] DENNIS, A. K. *Raspberry Pi home automation with Arduino: automate your home with a set of exciting projects for the Raspberry Pi*. Packt Publishing, Birmingham, UK, 2013.

[15] DOMÍNGUEZ-BOLAÑO, T., CAMPOS, O., BARRAL, V., ESCUDERO, C. J., JA GARCÍA-NAYA, J. A. An overview of IoT architectures, technologies, and existing open-source projects. *Internet of Things 20* (2022), 100626.

[16] EUROPEAN DATA PROTECTION BOARD. Ohjeet 4/2019 25 artiklan mukaisesta sisäänrakennetusta ja oletusarvoisesta tietosuojasta Versio 2.0. URL `https://edpb.europa.eu/system/files/2021-04/edpb_guidelines_201904_dataprotection_by_design_and_by_default_v2.0_fi.pdf`, viitattu 20.7.2023.

[17] FENTON, N., JA BIEMAN, J. *Software Metrics: A Rigorous and Practical Approach, Third Edition*, 3rd ed. CRC Press, Inc., USA, 2014.

[18] FENTON, N. E., JA NEIL, M. Software metrics: successes, failures and new directions. *Journal of Systems and Software 47*, 2 (1999), 149–157.

[19] FENTON, N. E., JA NEIL, M. Software Metrics: Roadmap. Julkaisusarjassa *Proceedings of the Conference on The Future of Software Engineering* (New York, NY, USA, 2000), Association for Computing Machinery, 357–370.

[20] FOUCAULT, M., FALLERI, J.-R., JA BLANC, X. Code Ownership in Open-Source Software. Julkaisusarjassa *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (New York, NY, USA, 2014), Association for Computing Machinery.

[21] FOUCAULT, M., TEYTON, C., LO, D., BLANC, X., JA FALLERI, J.-R. On the usefulness of ownership metrics in open-source software projects. *Information and Software Technology 64* (2015), 102–112.

[22] GITHUB. Gladys Assistant. URL `https://github.com/GladysAssistant`, viitattu 26.3.2024.

[23] GITHUB. Home Assistant. URL `https://github.com/home-assistant`, viitattu 26.3.2024.

[24] GITHUB. Jeedom. URL `https://github.com/jeedom`, viitattu 26.3.2024.

[25] GITHUB. openHAB. URL `https://github.com/openhab`, viitattu 26.3.2024.

[26] GLADYS ASSISTANT. About Communauté Gladys Assistant. URL `https://community.gladysassistant.com/about`, viitattu 24.3.2024.

[27] GLADYS ASSISTANT. About Gladys Community (EN). URL `https://en-community.gladysassistant.com/about`, viitattu 24.3.2024.

[28] GLADYS ASSISTANT. Open metrics. URL `https://gladysassistant.com/open/`, viitattu 31.1.2024.

[29] GLADYS ASSISTANT. Why are device exposes only partially supported? URL `https://en-community.gladysassistant.com/t/why-are-device-exposes-only-partially-supported/119`, viitattu 8.4.2024.

[30] GOLDMAN, D. CNN Business: Your Samsung TV is eavesdropping on your private conversations. URL `https://money.cnn.com/2015/02/09/technology/security/samsung-smart-tv-privacy/index.html`, viitattu 24.4.2024.

[31] GREEN, P. E., KRIEGER, A. M., AGARWAL, M. K., JA JOHNSON, R. M. Adaptive Conjoint Analysis: Some Caveats and Suggestions; Comment. *JMR, Journal of Marketing Research 28*, 2 (toukokuu 1991), 215.

[32] GRÖNROOS, C., JA VOIMA, P. Critical service logic: making sense of value creation and co-creation. *Journal of the Academy of Marketing Science 41*, 2 (2012), 133–150.

[33] HAADE - HOME ASSISTANCE ALSACE HOME AUTOMATION ELECTRONICS. Jeedom. URL https://haade.fr/en/category/jeedom, viitattu 31.1.2024.

[34] HAADE - HOME ASSISTANCE ALSACE HOME AUTOMATION ELECTRONICS. Jeedom Sas is over. URL https://haade.fr/en/blog/jeedom-is-over-welcome-domadoo-suite-aventure, viitattu 31.1.2024.

[35] HAINDL, P., JA PLÖSCH, R. Value-oriented quality metrics in software development: Practical relevance from a software engineering perspective. *IET Software 16*, 2 (2021), 167–184.

[36] HANSEN, M., KÖHNTOPP, K., JA PFITZMANN, A. The Open Source approach opportunities and limitations with respect to security and privacy. *Computers Security 21*, 5 (2002), 461–471.

[37] HARJU, H., JA KOSKELA, M. *Kustannustehokas ohjelmiston luotettavuuden suunnittelu ja arviointi. Osa 2*. VTT tiedotteita. Valtion teknillinen tutkimuskeskus, Espoo, 2003.

[38] HATZIVASILIS, G., PAPAEFSTATHIOU, I., JA MANIFAVAS, C. Software Security, Privacy, and Dependability: Metrics and Measurement. *IEEE Software 33*, 4 (2016), 46–54.

[39] HELANDER, N., JA ULKUNIEMI, P. Customer perceived value in the software business. *The Journal of High Technology Management Research 23*, 1 (2012), 26–35.

[40] HOME ASSISTANT. About Home Assistant Community. URL https://community.home-assistant.io/about, viitattu 24.3.2024.

[41] HOME ASSISTANT. Awaken your home. URL https://www.home-assistant.io/, viitattu 17.1.2024.

[42] HOME ASSISTANT. Home Assistant Analytics. URL https://analytics.home-assistant.io/, viitattu 22.1.2024.

[43] HOME ASSISTANT. Home Assistant SkyConnect. URL https://www.home-assistant.io/skyconnect, viitattu 17.1.2024.

[44] HOME ASSISTANT. Home Assistant Yellow. URL https://www.home-assistant.io/yellow, viitattu 17.1.2024.

[45] HOME ASSISTANT. Security audits of Home Assistant. URL https://www.home-assistant.io/blog/2023/10/19/security-audits-of-home-assistant/, viitattu 17.1.2024.

[46] HOWARD, M., PINCUS, J., JA WING, J. Measuring Relative Attack Surfaces. Kirjassa *Computer security in the 21st century*, D. T. Lee, S. P. Shieh, ja J. D. Tygar, Eds. Springer, 2005, ss. 109–137.

[47] IBM. What is DevOps? URL https://www.ibm.com/topics/devops, viitattu 12.11.2023.

[48] INDIE HACKERS: PIERRE-GILLES LEYMARIE. Gladys Assistant. URL https://www.indiehackers.com/product/gladys, viitattu 31.1.2024.

[49] INFIELD, G. A Computer in the Basement? *Popular Mechanics 129*, 4 (huhtikuu 1968), 77–79, 209, 229.

[50] INTEL CORPORATION. *USB 3.0* Radio Frequency Interference Impact on 2.4 GHz Wireless Devices*. Whitepaper 327216-001, Huhtikuu 2012.

[51] ISO25000.COM. The ISO/IEC 25000 series of standards. URL https://iso25000.com/index.php/en/iso-25000-standards, viitattu 1.11.2023.

[52] ISO/IEC 25022:2016. *Systems and software engineering Systems and software quality requirements and evaluation (SQuaRE) Measurement of quality in use*, 2016.

[53] JARCZYK, O., GRUSZKA, B., JAROSZEWICZ, S., BUKOWSKI, L., JA WIERZBICKI, A. GitHub Projects. Quality Analysis of Open-Source Software. Julkaisusarjassa *Social Informatics: 6th International Conference, SocInfo 2014, Barcelona, Spain, November 11-13, 2014. Proceedings* (Cham, 2014), Springer International Publishing, 80–94.

[54] JEEDOM. Bluetooth Compatibility. URL https://compatibility.jeedom.com/index.php?p=home&lang=en_US&page=1&protocol=Bluetooth, viitattu 15.4.2024.

[55] JEEDOM. À propos du site Communauté Jeedom. URL https://community.jeedom.com/about, viitattu 24.3.2024.

[56] JEEDOM COMMUNITY. Ruuvi qui ne redescend aucune info. URL https://community.jeedom.com/t/ruuvi-qui-ne-redescend-aucune-info/51157, viitattu 15.4.2024.

[57] JEEDOM DOCUMENTATION. Changelog Jeedom V4.4. URL https://doc.jeedom.com/en_US/core/4.4/changelog, viitattu 7.2.2024.

[58] JEEDOM DOCUMENTATION. Installation on Raspberry Pi. URL https://doc.jeedom.com/en_US/installation/rpi, viitattu 7.2.2024.

[59] KANDENGWA, E., JA KHOZA, L. T. Measuring Agile software project success beyond the triple constraint. *South African Journal of Information Management 23*, 1 (2021).

[60] KASANEN, E., LUKKA, K., JA SIITONEN, A. The constructive approach in management accounting research. *Journal of management accounting research 5* (1993), 243.

[61] KUPIAINEN, E., MÄNTYLÄ, M. V., JA ITKONEN, J. Using metrics in Agile and Lean Software Development A systematic literature review of industrial studies. *Information and Software Technology 62* (2015), 143–163.

[62] KYNE, D. OpinionX: How To Calculate Conjoint Analysis Results [8 Steps]. URL https://www.opinionx.co/research-method-guides/conjoint-analysis-calculation, viitattu 2.4.2024.

[63] LEYMARIE, P.-G. Refactoring Gladys Developer Platform (Part 1). URL https://pierregillesleymarie.com/blog/gladys/2017/04/22/refactoring-gladys-developer-website.html, viitattu 31.1.2024.

[64] LI, S., XU, L. D., JA ZHAO, S. The internet of things: a survey. *Information Systems Frontiers 17*, 2 (2014), 243–259.

[65] LUKKA, K. Konstruktiivinen tutkimusote: luonne, prosessi ja arviointi. Kirjassa *Soveltava Yhteiskuntatiede Ja Filosofia*, K. Rolin, M.-L. Kakkuri-Knuuttila, E. Henttonen, ja K. Eräranta, Eds. Gaudeamus, 2006, ss. 111–133.

[66] MANADHATA, P. K., JA WING, J. M. An Attack Surface Metric. *IEEE Transactions on Software Engineering 37*, 3 (2011), 371–386.

[67] MCLEOD, S. Simply Psychology: Likert Scale Questionnaire: Examples Analysis. URL https://www.simplypsychology.org/likert-scale.html, viitattu 8.4.2024.

[68] MERK, H.-J. OpenHAB Community: openHAB analytics. URL https://community.openhab.org/t/openhab-analytics/121676/9, viitattu 17.1.2024.

[69] MIPS. MQTT Discovery documentation. URL https://mips2648.github.io/jeedom-plugins-docs/MQTTDiscovery/en_US/, viitattu 16.4.2024.

[70] MORRISON, P., MOYE, D., PANDITA, R., JA WILLIAMS, L. Mapping the field of software life cycle security metrics. *Information and Software Technology 102* (2018), 146–159.

[71] NABU CASA, INC. About us. URL https://www.nabucasa.com/about/, viitattu 17.1.2024.

[72] NORTH DATA. Jeedom SAS, Rillieux-la-Pape, France. URL https://www.northdata.com/Jeedom+SAS,+Rillieux-la-Pape/Siren+810505784, viitattu 31.1.2024.

[73] OPENHAB. About openHAB Community. URL https://community.openhab.org/about, viitattu 24.3.2024.

[74] OPENHAB. OpenHAB Add-on Reference. URL https://www.openhab.org/addons/, viitattu 7.2.2024.

[75] OPENHAB. openHABian - Hassle-free openHAB Setup. URL https://www.openhab.org/docs/installation/openhabian.html#passwords, viitattu 15.4.2024.

[76] OPENHAB. Welcome to myopenHAB. URL https://www.myopenhab.org/, viitattu 17.1.2024.

[77] OPENHAB COMMUNITY. Phoscon/deCONZ on openHABian. URL https://community.openhab.org/t/phoscon-deconz-on-openhabian/85030, viitattu 15.4.2024.

[78] OPENHAB FOUNDATION. Become a Member. URL https://www.openhabfoundation.org/join/, viitattu 17.1.2024.

[79] PERERA, C., BARHAMGI, M., BANDARA, A. K., AJMAL, M., PRICE, B., JA NUSEIBEH, B. Designing privacy-aware internet of things applications. *Information Sciences 512* (2020), 238–257.

[80] PHOSCON. ConBee II Installations: RaspBerry Pi OS. URL https://phoscon.de/en/conbee2/install#raspbian, viitattu 15.4.2024.

[81] PIIRAINEN, K. A., JA GONZALEZ, R. A. Seeking Constructive Synergy: Design Science and the Constructive Research Approach. Julkaisusarjassa *8th International Conference, DESRIST 2013* (Helsinki, kesäkuu 2013), Springer, 66–75.

[82] POLLACK, J., HELM, J., JA ADLER, D. What is the Iron Triangle, and how has it changed? *International Journal of Managing Projects in Business 11*, 2 (2018), 527–547.

[83] RAO, V. R. *Applied Conjoint Analysis*. Springer Berlin, Heidelberg, 2014.

[84] RAUTIAINEN, A., SIPPOLA, K., JA MÄTTÖ, T. Perspectives on Relevance: the Relevance Test in the Constructive Research Approach. *Management Accounting Research 34* (2017), 19–29.

[85] RINDELL, K., RUOHONEN, J., HOLVITIE, J., HYRYNSALMI, S., JA LEPPÄNEN, V. Security in agile software development: A practitioner survey. *Information and Software Technology 131*, 1 (2021), 106488.

[86] ROCKETREACH. Nabu Casa Information. URL https://rocketreach.co/nabu-casa-profile_b7e40836c0713cf2, viitattu 17.1.2024.

[87] ROY, J., CONTINI, C., BRODEUR, F., DIOUF, N., JA SURYN, W. Method for the Evaluation of Open Source Software Quality from an IT Untrained User Perspective. Julkaisusarjassa *Proceedings of the 2014 International C* Conference on Computer Science & Software Engineering* (New York, NY, USA, 2014), C3S2E '14, Association for Computing Machinery.

[88] RÉPUBLIQUE FRANÇAISE: BULLETIN OFFICIEL DES ANNONCES CIVILES ET COMMERCIALES. Annonce n^o 1226 du BODACC A n^o 20230186

publié le 27/09/2023. URL https://www.bodacc.fr/pages/annonces-commerciales-detail/?q.id=id:A202301861226, viitattu 31.1.2024.

[89] SANASTOKESKUS RY. TEPA-termipankki. URL https://termipankki.fi/tepa/fi/haku/%C3%A4lykoti, viitattu 23.9.2023.

[90] SARRAB, M., JA REHMAN, O. M. H. Empirical study of open source software selection for adoption, based on software quality characteristics. *Advances in Engineering Software 69* (2014), 1–11.

[91] SAWTOOTH SOFTWARE. Adaptive choice-based conjoint. URL https://sawtoothsoftware.com/conjoint-analysis/acbc, viitattu 4.3.2024.

[92] SCHOMAKERS, E.-M., BIERMANN, H., JA ZIEFLE, M. Users Preferences for Smart Home Automation Investigating Aspects of Privacy and Trust. *Telematics and Informatics 64* (2021), 101689.

[93] SCHUSTER, F., JA HABIBIPOUR, A. Users Privacy and Security Concerns that Affect IoT Adoption in the Home Domain. *International journal of human-computer interaction* (2022), 1–12.

[94] SETZ, B., GRAEF, S., IVANOVA, D., TIESSEN, A., JA AIELLO, M. A Comparison of Open-Source Home Automation Systems. *IEEE Access 9* (2021), 167332–167352.

[95] SFS-ISO/IEC 25010:2019. *Järjestelmäkehitys ja ohjelmistotuotanto. Järjestelmien ja ohjelmistojen laatuvaatimukset ja niiden laadun arviointi (SQuaRE). Tietojärjestelmien ja ohjelmistojen laatumallit*, 2019.

[96] SHANKER, A. An Enterprise Perspective on Customer Value Propositions for Open Source Software. *Technology Innovation Management Review 2*, 12 (12 2012), 28–36.

[97] SOFIA, R. C., JA SOLDATOS, J. *Shaping the Future of IoT with Edge Intelligence : How Edge Computing Enables the Next Generation of IoT Applications*. River Publishers, New York, 2024.

[98] SPICER, D. Computer History Museum Blog: The ECHO IV Home Computer: 50 Years Later. URL https://computerhistory.org/blog/the-echo-iv-home-computer-50-years-later/, viitattu 23.9.2023.

[99] STATISTA. Number of Smart Homes per segment forecast in Finland from 2020 to 2028. URL https://www.statista.com/forecasts/887697/number-of-smart-homes-per-segment-in-finland, viitattu 24.4.2024.

[100] THE MITRE CORPORATION - CVE. FAQ: What does DATE RECORD CREATED signify in a CVE Record? URL https://www.cve.org/ResourcesSupport/FAQs#pc_cve_recordsdate_record_created_in_cve_record, viitattu 9.3.2024.

[101] THE MITRE CORPORATION - CVE. Process. URL https://www.cve.org/About/Process, viitattu 3.4.2024.

[102] THEENGS. Theengs Gateway: BLE to MQTT bridge - Install. URL https://gateway.theengs.io/install/install.html#install-theengs-gateway-as-a-pip-package, viitattu 16.4.2024.

[103] THREAD GROUP, INC. Thread Network Fundamentals. URL https://www.threadgroup.org/Portals/0/documents/support/Thread%20Network%20Fundamentals_v3.pdf, viitattu 14.2.2024.

[104] TYRVÄINEN, P., SAARIKALLIO, M., AHO, T., LEHTONEN, T., JA PAUKERI, R. Metrics Framework for Cycle-Time Reduction in Software Value Creation. Julkaisusarjassa *International Conference on Software Engineering Advances* (Barcelona, marraskuu 2015), 220–227.

[105] VALMISTAJAT.FI. Automaatio ja automaatiojärjestelmät. URL https://valmistajat.fi/menetelmat/elektroniikka/automaatio-ja-automaatiojarjestelmat, viitattu 23.9.2023.

[106] YILMAZ, N., JA KOLUKISA TARHAN, A. Quality evaluation models or frameworks for open source software: A systematic literature review. *Journal of Software: Evolution and Process 34*, 6 (2022), e2458.

[107] ZIGBEE2MQTT. IKEA E1525/E1745. URL https://www.zigbee2mqtt.io/devices/E1525_E1745.html, viitattu 8.4.2024.

[108] ZIGBEE2MQTT. Installation: Linux. URL https://www.zigbee2mqtt.io/guide/installation/01_linux.html, viitattu 16.4.2024.

[109] ZIGBEE2MQTT. Zigbee2MQTT: Devices. URL https://www.zigbee2mqtt.io/supported-devices/, viitattu 7.2.2024.