Opinnäytetyö AMK

Tieto- ja viestintätekniikka (insinööri)

2021

Juhani Pirilä

AWS IOT -JÄRJESTELMÄN PUHEOHJAUS

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ AMK | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka (insinööri)

2021 | 29 sivua

Juhani Pirilä

AWS IOT -JÄRJESTELMÄN PUHEOHJAUS

IoT-laitteiden lisääntyessä kiinnostus puheohjaukseen on lisääntynyt ja luonut tarpeen liittää puheohjaus valmiisiin IoT-järjestelmiin. Tämän opinnäytetyön aiheena on puhekäyttöliittymän luominen Amazon Web Services -palvelua käyttävään esineiden internet -järjestelmään. Tarkoituksena on selvittää, miten järjestelmään voidaan liittää Alexa-virtuaaliavustaja ja ohjata laitteita äänikomennoilla. Työ esittelee tarvittavia ohjelmistokehityspalveluja ja -teknologioita, käy läpi vaiheet Alexan liittämisestä järjestelmään, sekä raportoi järjestelmän käytettävyyttä mittaavia lukuja.

Työ toteutettiin luomalla AWS IoT -järjestelmä ja liittämällä siihen Alexa-virtuaaliavustaja käyttämällä Amazonin Alexa dokumentointeja, ja dokumentoimalla järjestelmän toteutus. Käytettävyyttä tutkittiin tarkastelemalla Alexan raportoimia käytettävyyttä mittaavia lukuja.

Työn lopputuloksena saatiin prosessin dokumentointi ja käsitys tarvittavien osien rooleista järjestelmässä sekä puhekäyttöliittymän liittämisen edellytykset ja vaatimukset. Johtopäätöksenä huomattiin Amazon Web Services ympäristön hyötyjen korreloitumisen Alexa ympäristön kanssa ja järjestelmän toimivuuden toteutuminen.

ASIASANAT:

Esineiden internet, puhekäyttöliittymä, pilvipalvelut

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and communications technology (engineer)

2021 | 29 pages

Juhani Pirilä

SPEECH CONTROL OF AWS IOT SYSTEM

As IoT devices have increased, interest in speech control has increased and created the need to integrate speech control into ready-made IoT systems. The topic of this thesis work is the creation of voice user interface for an Internet of Things system that is using Amazon Web Services. The purpose was to find out how to connect an Alexa virtual assistant to the system and control devices using voice commands. The thesis introduces the necessary software development services and technologies, goes through steps of connecting Alexa to the system and reports the numbers measuring the usability of the system.

The thesis was implemented by creating AWS IoT system and attaching an Alexa virtual assistant to it using Amazon Alexa documents, and documenting the implementation. Usability was examined by looking at usability figures reported by Alexa.

The result of the thesis was the documentation of the process and the understanding of the roles of the necessary parts in the system, as well as the conditions and requirements for connecting the voice user interface. In conclusion, it was clear that the benefits of the Amazon Web Services environment were correlated with the Alexa environment and the system functionality was successful.

KEYWORDS:

Internet of things, voice user interface, cloud services

SISÄLTÖ

KUVAT

1 JOHDANTO

Elektronisten laitteiden valmistajat kytkevät laitteensa yhä useammin internetverkkoon, jossa laitteita voidaan seurata tai niitä voidaan ohjata verkon yli käyttöliittymillä. Vuonna 2010 esineiden internetiin (IoT) liitettyjä laitteita oli 0,8 miljardia ja kaikkiaan laitteita oli 8,8 miljardia. Vuonna 2019 maailmassa oli yhteensä noin 20 miljardia laitetta ja IoT-laitteiden osuus kaikista laitteista oli noin 50% eli 10 miljardia. Määrän uskotaan nousevan vuoteen 2025 mennessä noin 30,9 miljardiin ja 75 prosenttiin. Iot:n yleistyessä graafiset käyttöliittymät ovat saaneet seurakseen puhekäyttöliittymät eli puhekomentoja ymmärtävät ohjelmat. Puhekäyttöliittymällä on käytettävyyden näkökulmasta merkittäviä etuja verrattuna graafiseen käyttöliittymään kuten käsien ja silmien vapaus käytettäessä ja käytettävyys etänä ilman mukana kannettavia laitteita. Puhekäyttöliittymällä on myös hyötynä se, että laitteiden määrän kasvaessa suureksi se ei vaikuta käytettävyyteen, sillä ihmisen on luontevampaa kertoa mitä haluaa tehdä kuin etsiä laite esimerkiksi puhelin -sovelluksesta. Usein jokaisella valmistajalla on myös oma sovellus, jolloin usean laitteen ohjaaminen kännykällä on kömpelöä, sillä käyttäjä joutuu vaihtelemaan eri sovellusten välillä.[1]

Työn tavoitteena on toteuttaa ja tutkia puhekäyttöliittymän liittämistä Amazon Web Services (AWS) -palvelua käyttävään IoT-järjestelmään ja tutkia puhekäyttöliittymän toimivuutta ja käytettävyyttä. AWS on yksi mailman käytetyimmistä pilvilaskentapalveluista ja jota myös Amazonin omistama Alexa-virtuaaliavustaja hyödyntää.[2] Opinnäytetyön tarkoituksena on vastata kysymyksiin, miten puhekäyttöliittymä saadaan liitettyä AWS IoT -järjestelmään, miten se toimii, sekä kuinka varmasti puheohjaus toimii ja kuinka nopeasti. IoT-järjestelmänä työssä käytetään älykotijärjestelmää ja puhekäyttöliittymänä Amazonin kehittämän Alexa-virtuaaliavustajan älykoti rajapintaa. IoT-järjestelmän ja puhekäyttöliittymän yhteensopivuuden varmistamiseksi työssä on käytetty saman yhtiön palveluja ja tuotteita. Työssä käsitellään vain olennaisimpia Alexan tarjoamia toimintoja, eikä käsitellä esimerkiksi Alexa-sovelluksen toimintoja.

2 AWS IOT -JÄRJESTELMÄ

AWS IoT on Amazonin tarjoama palvelu, joka mahdollistaa internetiin liitettyjen laitteiden ja pilvisovellusten turvallisen kommunikoimisen keskenään. Keskenään kommunikoivat laitteet ja sovellukset muodostavat IoT-järjestelmän. Järjestelmän osapuolet kommunikoivat keskenään käyttäen MQ Telemetry Transport (MQTT) protokollaa, jossa viesti lähetetään aina aiheella viestien välittäjälle ja välittäjä välittää viestit osapuolille, jotka ovat tilanneet kyseisen aiheen viestit. [3]

2.1 AWS IoT -laitteiden ohjaus

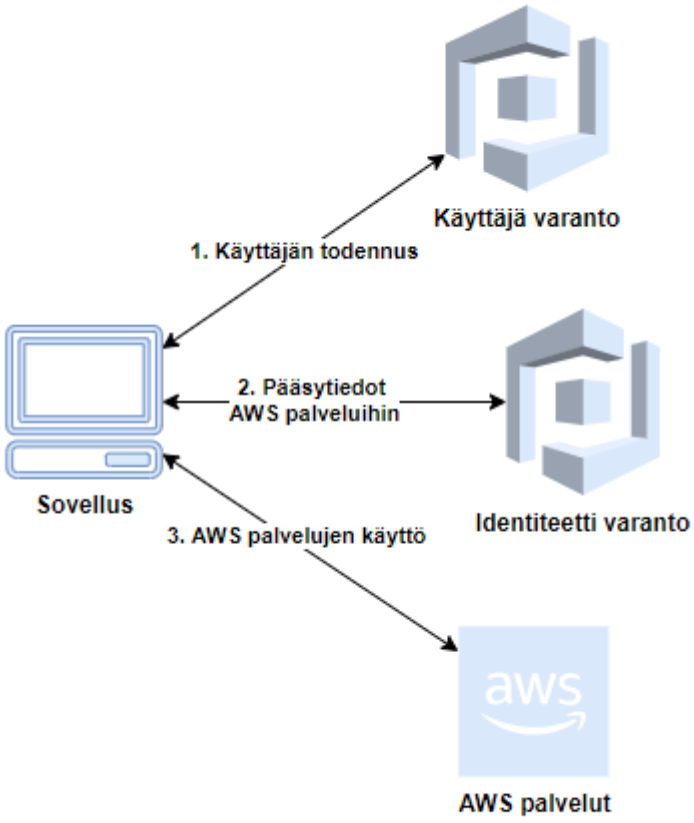
IoT -järjestelmän laitteita pystytään ohjaamaan verkon yli käyttöliittymällä, eli laitteille pystytään lähettämään viestejä ohjelmista, joiden mukaan laite toimii. Viestit lähetetään yleensä Java Script Object Notation (JSON) -muodossa, mutta myös tekstimuoto on sallittu. Viestien lähetys AWS IoT -palvelussa ohjelmasta laitteelle tapahtuu viestien välittäjän nimeltä AWS IoT Core kautta. AWS IoT Core sijaitsee AWS palvelimella, joka hoitaa myös järjestelmän osapuolten todentamisen ja valtuutuksen. Ohjelmat voivat kommunikoida laitteiden kanssa palvelimen rajapintojen kautta, jotka käyttävät joko MQTT tai HTTPS protokollaa. AWS tarjoaa ohjelmistokehityspaketteja eri kielillä, joissa on valmiiksi rakennettu toiminnallisuus yhdistää ja lähettää viestejä palvelimelle. [3]

2.2 Asiakasohjelmien todentaminen ja valtuutus

AWS IoT -järjestelmän laitteiden todentamiseen ja valtuuttamiseen käytetään x.509-sertifikaatteja, jotka luodaan laitteiden luomisen yhteydessä AWS IoT -konsolissa. Ohjelmat taas hyödyntävät Cognito-palvelua ja Identity and Access Management (IAM) -palvelua. Cognito-palvelussa luodaan käyttäjävaranto, joka hallinnoi käyttäjien käyttäjätunnuksia ja salasanoja, sekä identiteettivaranto, joka valtuuttaa todennetuille käyttäjille pääsyn AWS-palveluihin. IAM-palvelussa päätetään mihin AWS-palveluihin oikeuksia annetaan. Oikeuksia voidaan antaa luomalla käytäntöjä palveluihin ja samalla valitaan mitä oikeuksia käytännöllä on palveluun. Esimerkiksi voidaan luoda käytäntö, jolla on lukemisoikeudet DynamoDB tietokanta -palvelun Laitteet nimiseen tauluun. Lisäksi luodaan rooli, johon voidaan liittää käytäntöjä, jotka antavat roolille niiden

sisältämät oikeudet ja identiteettivarannossa rooli konfiguroidaan todennettuille käyttäjille. [4,5]

Kuvan 1 tapaan ohjelman on tarkoitus käyttäjätunnusta ja salasanaa käyttämällä pyytää käyttäjävarannolta valtuutuskoodin identiteettivarantoon, ja valtuutuskoodia käyttämällä ohjelma saa identiteettivarannolta pääsytiedot AWS -palveluihin. AWS ohjelmisto -palveluissa, kuten AWS Lambda, oikeudet voidaan myös asettaa suoraan IAM -palvelun kautta. [4,5]

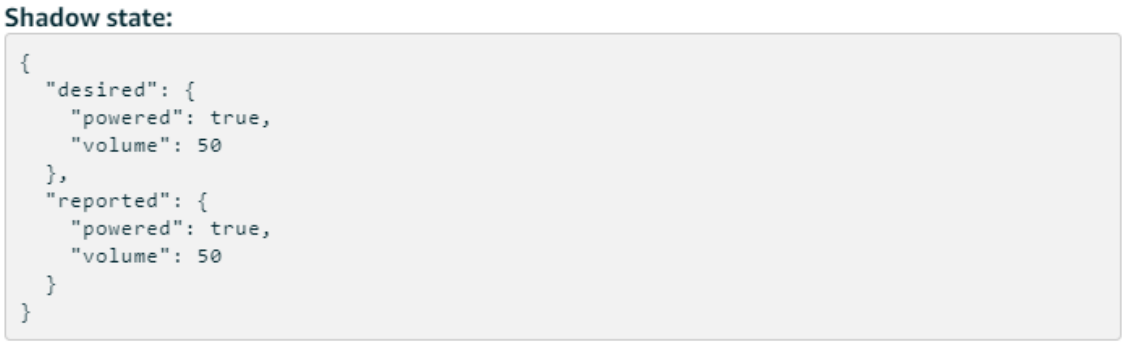


Kuva 1. Ohjelman käyttäjän todentaminen ja valtuutus. [4]

2.3 Laiteiden varjot

Osana AWS IoT -palvelua on laiteiden varjo -palvelu, joka ylläpitää ja tallentaa AWS IoT -palveluun liitettyjen laitteiden varjoja. Laitteiden varjoilla tarkoitetaan laitteen tilaa

kuvaavaa JSON-tiedostoa, jossa on sekä haluttu että raportoitu tila kuvan 2 mukaisesti. Laitteet ja sovellukset voivat vuoro vaikuttaa varjojen kanssa kolmella tavalla. Ne voivat päivittää, pyytää tai poistaa varjoja. Varjon päivittäminen tapahtuu lähettämällä JSON-objekti, jossa on varjon mukainen rakenne sekä kentät ja niiden arvot, joita halutaan päivittää. Varjon kentät päivittyvät lähetetyn mukaisiksi ja kentät, joita lähetetty objekti ei sisällä, pysyvät samoina. [3]



Kuva 2. Esimerkki AWS IoT -laitteen varjo tiedostosta.

Laitteiden varjoja voidaan käyttää palvelun rajapintojen kautta MQTT tai HTTPS protokollaa käyttäen. Palvelussa on varattu tietyt MQTT aiheet varjojen eri toiminnoille, kuten computer nimisen laitteen varjon päivittämiseen on varattu aihe $aws/things/computer/shadow/update. HTTPS protokollalla varjoja voidaan pyytää käyttämällä GET metodia, päivittää käyttämällä POST metodia tai poistaa käyttämällä DELETE metodia. [3]

Ohjelmien on tarkoitus käyttää varjoja laitteiden ohjaamiseen päivittämällä haluttua tilaa, jolloin varjo -palvelu lähettää laitteeseen halutun tilan ja laite muuttaa tilaansa sen mukaan. Lopuksi laite päivittää raportoidun tilan varjo -palveluun. Palvelussa on varattu MQTT aihe $aws/things/(laitteen nimi)/shadow/update/delta johon palvelu lähettää halutun ja raportoidun tilan erot. Tällöin laite voi tilata aiheen ja vastaan ottaa eron, jolloin laitteen ei itse tarvitse verrata haluttua ja nykyistä tilaa. [3]

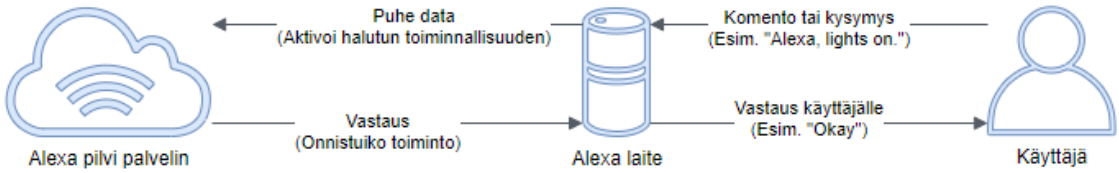
3 PUHEKÄYTTÖLIITTYMÄ

Puhekäyttöliittymä on ohjelmisto, joka tunnistaa ihmisen puhetta ja etsii siitä valmiiksi määritettyjä ilmauksia ja toimii niiden mukaan, esimerkiksi ohjaa laitetta. Ohjelmaa voidaan ajaa esimerkiksi puhelimessa ja älykaiuttimessa tai muulla tietokoneella, johon on liitetty mikrofoni ja kaiutin. Puhekäyttöliittymä voidaan toteuttaa itse hyödyntämällä puheentunnistus teknologiaa tai käyttää markkinoilla olevia virtuaaliavustajia. Virtuaaliavustajia tarjoavilla yhtiöillä on palveluja, joiden avulla voidaan luoda yksilöllistetty käyttöliittymä omalle järjestelmälle. Palvelujen käyttö nopeuttaa ja helpottaa huomattavasti käyttöliittymän tekemistä, joten niiden hyödyntäminen on suositeltavaa. Markkinoilla on tarjolla muutamien eri valmistajien virtuaaliavustajia ja kaksi eniten käytetyintä näistä on Amazon Alexa ja Google Assistant. Koska liitettävä järjestelmä on jo Amazon ympäristössä, on loogisinta valita Amazon Alexa.

3.1 Amazon Alexa -virtuaaliavustaja

Amazon Alexa on mahdollista ladata älypuhelimille sovelluskaupasta tai asentaa kotitietokoneelle. Alexa löytyy myös muun muassa useista älykaiuttimista tai älytelevisioista. Laitteeseen integroitu Alexa kuuntelee jatkuvasti puhetta ja aktivoituu, jos kuulee aktivointi sanan, joka on oletuksena Alexa. [6]

Alexa koostuu pääasiassa kahdesta osasta, paikallisesta laitteeseen integroitavasta ohjelmasta ja mahdollisesta laitteistosta sekä pilvipalvelusta. Ohjelman ja laitteiston tehtävänä on prosessoida ääni ja tunnistaa aktivointisana, sekä aktivointi sanan huomattuaan lähettää puhe data pilvipalvelulle. Pilvipalvelun tehtävänä on prosessoida puhedata ja aktivoida dataa vastaava toiminnallisuus. Toiminnon jälkeen palvelu lähettää vastauksen ohjelmalle, joka toistaa kaiuttimesta vastauksen käyttäjälle. Kuva 3 havainnollistaa miten laite on yhteydessä palvelimella sijaitsevaa pilvipalveluun ja että käytössä olevat toiminnallisuudet on konfiguroitu sinne. Palvelimen hyödyntäminen mahdollistaa uusien toimintojen helpon päivityksen ja tarjoaa suuremman laskentatehon toiminnoille. [6]



Kuva 3. Alexa-virtuaaliavustajan datan kulku. [6]

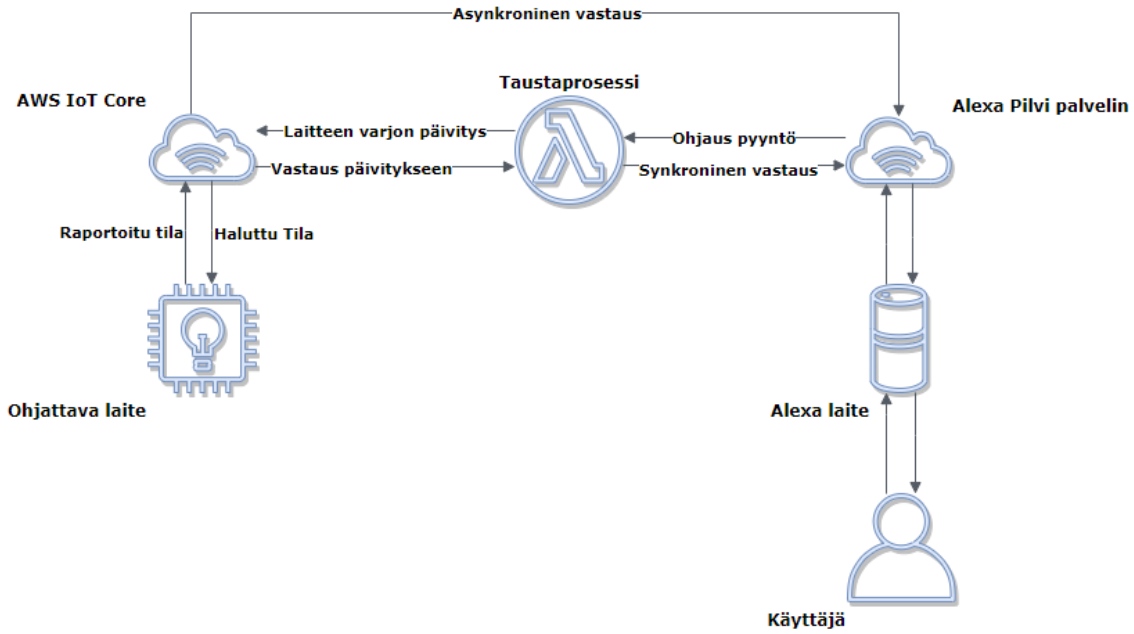
3.2 Alexa skills -palvelu

Alexan toiminnallisuuksia kutsutaan nimellä skills eli taidot. Amazon tarjoaa Alexa skills kit -palvelun ohjelmistokehittäjille, jossa pystyy luomaan uusia taitoja. Uutta taitoa luodessa taidolle pitää ensin valita vuorovaikutusmalli taidon tyypin perusteella. Taidot on luokiteltu eri tyyppeihin, kuten musiikki ja älykoti taidot. Valitsemalla eri tyypin palvelu luo valmiin käyttöliittymän mallin, joka määrittää miten käyttäjä voi käyttää taitoa. Malli rajaa mitkä käyttäjän komennot ja kysymykset ovat sallittuja ja miten ne toimivat. Palvelussa on myös mahdollista valita tyhjä malli, jossa kehittäjä luo oman vuorovaikutusmallin. Lisäksi taidolle valitaan taustaprosessin päätepiste eli valitaan palvelu, jota kutsutaan taitoa käyttäessä ja missä itse toiminto tapahtuu.

3.2.1 Älykoti rajapinta

Alexaa liitettäessä älykoti IoT -järjestelmään kannattaa malliksi valita valmis älykoti malli. Älykoti-mallissa taito käyttää silloin älykoti rajapintaa, jolle taito kuvailee ensin järjestelmässä olevien laitteiden ominaisuudet ja rajapinta muodostaa vuorovaikutusmallin näiden perusteella. Taito kuvaa laitteiden ominaisuudet lähettämällä rajapinnalle laitteen ominaisuuksia kuvaavan JSON-mallin, jossa on määritelty mitä toimintoja laite tukee, sekä laitteen kommunikointi päätepisteen. JSON-malli lähetetään rajapinnalle joko vastauksena ”löytöpyyntöön”, joka lähetetään käyttäjän pyynnöstä löytää laitteita, tai ennakoivasti esimerkiksi uuden laitteen liittyessä lähiverkkoon. Kun vuorovaikutusmalli on muodostettu, voi käyttäjä hallita laitteita vuorovaikutusmallin mukaisilla komennoilla, jolloin komennon tieto liikkuu verkossa kuvan 4 mukaisesti. Alexa lähettää pyynnön taustaprosessille, joka toteuttaa toiminnon ja vastaa Alexalle synkronisesti tai IoT -

järjestelmästä asynkronisesti. Kaikkiin toimintoihin ei kuitenkaan voi vastata asynkronisesti. [7]



Kuva 4. Yleiskuva älykoti rajapinnan toiminnasta. [7]

Tieto liikkuu Alexa palvelimen ja taustaprosessin välillä JSON-muodossa, jossa on määritelty, mitä toimintoa käyttäjä ohjaa, mahdolliset arvot toiminnolle ja laitteen päätepiste. Käyttäjä voi esimerkiksi ohjata valon kirkkautta, jolloin älykoti rajapinta kutsuu ohjatun laitteen taustaprosessia pyynnöllä, jossa on kuvan 5 mukainen JSON-sisältö. [8]

```
{
    "directive": {
        "header": {
            "namespace": "Alexa.ColorController",
            "name": "SetColor",
            "payloadVersion": "3",
            "messageId": <Pyynnön tunniste koodi>,
            "correlationToken": <Pyynnön viittaus koodi>
        },
        "endpoint": {
            "scope": {
                "type": "BearerToken",
                "token": <Valtuutuskoodi>
            },
            "endpointId": <Laitteen nimi>,
            "cookie": {}
        },
        "payload": {
            "color": {
                "hue": 240,
                "saturation": 1,
                "brightness": 1
            }
        }
    }
}
```

Kuva 5. Taustaprosessille lähetettävän värin ohjaus pyynnön esimerkki sisältö.

Älykoti rajapinta myös mahdollistaa laitteiden ryhmittämisen ja laitteiden käyttämisen rutiineissa. Käyttäjä voi asettaa laitteita ryhmiin ja antaa komentoja koko ryhmälle. Esimerkiksi käyttäjä voi liittää valoja kitchen -ryhmään ja käyttää komentoa ”Alexa, kitchen lights on”. Rutiinit ovat automaattisia toimintoja, jotka aktivoituvat määrätyn tapahtuman seurauksena. Esimerkiksi käyttäjä voi luoda rutiinin, jossa valot syttyvät kello 8.

3.2.2 Käyttäjätunnuksen linkitys

Käytettäessä älykoti mallia taidon luomisessa taidolla täytyy olla käyttäjätunnuksen linkitys mahdollisuus. Taidolle täytyy konfiguroida rajapinnat, joista Alexa saa käyttäjän identiteetin laitteiden kommunikointi järjestelmässä OAuth2 -valtuutus protokollaa käyttämällä. Alexa käyttää valtuutuskoodia myöntämistapana, jolloin Alexa saa käyttöoikeuden käyttäjätunnusta ja salasanaa vastaan toisesta rajapinnasta ja käyttää sitä

noutaakseen valtuutuskoodin toisesta. Alexa hoitaa valtuutuskoodin haun ja päivityksen automaattisesti ja lähettää sen taustaprosessille yhdessä pyyntöjen kanssa. Käyttäjän näkökulmasta tämä tarkoittaa sitä, että käyttäjän ottaessa taidon käyttöön Alexa-sovelluksessa, täytyy käyttäjän kirjautua palveluun eli linkittää Amazon tunnukset palvelun tunnuksiin. Ideana käyttäjätunnuksen linkityksessä on, että taidon taustaprosessi tietää, ketä taitoa käyttää ja voidaan selvittää mitkä laitteet kuuluvat käyttäjälle. Laitteiden kommunikoinnin ollessa AWS IoT -palvelussa voidaan kirjautumispalveluna käyttää Amazon Cognito -palvelua. [9]

4 PUHEKÄYTTÖLIITTYMÄN INTEGROIMINEN AWS IOT -JÄRJESTELMÄÄN

Alexa voidaan liittää AWS IoT -järjestelmään ohjaamalla laitteita käyttäen AWS -ohjelmistokehityspakettia Alexa taidon Lambda taustaprosessista. AWS Lambda on ohjemistopilvipalvelu, jossa voidaan luoda funktioita eli koodinpätkiä. Niitä voidaan ajaa tarvittaessa ja niille voidaan konfiguroida laukaisin. Laukaisimena voi toimia esimerkiksi muutos tietokannassa tai kuten Alexan tapauksessa datan lähetys rajapintaan.

AWS IoT -järjestelmään integroitavan Alexa puhekäyttöliittymän integroiminen koostuu kahdesta osasta. Ensin Alexalle täytyy luoda taito ja sen jälkeen ohjelmoida taustaprosessi. Taidon luomisessa täytyy myös konfiguroida käyttäjätunnus linkityksen rajapinnat. Tässä työssä taustaprosessi toteutetaan käyttäen AWS Lambda -palvelua Node.js -suoritusympäristössä, ja se vastaa pyyntöihin synkronisesti. Alexa taidon käyttäjätunnuksen linkitykseen käytetään AWS Cognito -palvelua ja se hyödyntää älykoti vuorovaikutusmallia.

4.1 Alexa älykoti taidon luominen

Taidon luominen tapahtuu selaimella Alexan ohjelmistokehittäjäkonsolissa. Ensin kirjaudutaan konsoliin ja aloitetaan taidon luominen. Taidolle syötetään nimi, valitaan oletus kieli, valitaan vuorovaikutusmalliksi älykoti malli ja lopuksi valitaan taustaprosessiksi itse toteutettu palvelu. Kun taito on luotu, taidolle täytyy asettaa taustaprosessin päätepiste ja valita päätepisteeseen lähetettävän tietosisällön version. Tietosisällön versio vaikuttaa miten taustaprosessin pitää käsitellä ja vastata sille lähetettyyn dataan. Tässä työssä on käytetty tietosisältö versiota 3. Taustaprosessin päätepiste voidaan asettaa vasta Lambda funktion luomisen jälkeen, jolloin tiedetään sen päätepiste.

Viimeisenä Alexa taidolle täytyy konfiguroida käyttäjätunnuksen linkitys -rajapinnat. Ensin täytyy luoda käyttäjävaranto Cognito -palvelussa ja luoda sovellusasiakas Alexalle käyttäjävarantoon, jolloin saadaan tunniste ja salasana, jotka syötetään Alexa taidolle ja joiden avulla Alexa voi käyttää rajapintoja. Seuraavaksi taidolle syötetään rajapintojen osoitteet, joiden verkkotunnus löytyy käyttäjävarannon verkkotunnus -kohdasta sekä kirjautumisrajapinnan polku on /oauth2/authorize ja valtuutuskoodi rajapinnan polku on /oauth2/token. Seuraavaksi valitaan käyttäjävarannosta sovellusasiakas

asetuksista oAuth2 valtuutuksen tyypiksi valtuutuskoodi, joka on ainoa tyyppi, mitä Alexa tukee, ja tuetuksi laajuuksiksi aws.cognito.user.admin ja syötetään laajuudet myös taidolle. Laajuus määrittää, mitä oikeuksia valtuutuskoodilla on. Laajuus aws.cognito.user.admin mahdollistaa taustaprosessin Cognito käyttäjävaranto rajapinnan käytön, jolloin taustaprosessi voi sitä käyttäen tunnistaa valtuutuskoodia vastaavan käyttäjän. Lopuksi sovellusasiakas asetuksiin syötetään Alexan tarjoamat uudelleen ohjaus osoitteet, joihin kirjautumisen jälkeen siirrytään Alexa-sovelluksessa. [10,11]

4.2 Taustaprosessi

Alexa taidon taustaprosessi toimii Alexan ja AWS IoT -järjestelmän yhdistävänä osana. Prosessi käsittelee sille lähetettyä JSON-dataa ja ohjaa AWS IoT -laitteita sen mukaan. Lambda -palvelulla voidaan luoda ohjelma funktio, joka suoritetaan tarvittaessa ja skaalautuvasti. Funktio voidaan konfiguroida kutsuttavaksi muista AWS -palveluista kuten myös Alexasta ja se tukee useita ohjelmointi kieliä ja suoritus ympäristöjä kuten Node.js ja Python. Funktio voidaan ohjelmoida ohjaamaan laitteita päivittämällä laitteiden varjoja käyttäen AWS -ohjelmistokehityspakettia. AWS -ohjelmistokehityspaketti sisältyy automaattisesti lambda funktioihin. [12]

Lambda funktio luodaan AWS konsolissa ja luomisen yhteydessä valitaan nimi, ajoympäristö, sekä IAM oikeus käytännöt. Funktiolle luodaan rooli oletus lambda oikeuksilla ja sille voidaan antaa lisää oikeuksia tarvittaessa IAM -palvelussa. Lambda funktion palvelin sijainti täytyy valita Alexan kielen mukaan, sillä Alexa älykoti malli tukee jokaiselle kielelle vain yhden palvelin sijainnin. Esimerkiksi Yhdysvaltojen englannin kielelle on varattu palvelimen sijainti us-east-1. [13]

Jotta funktio saa oikeudet AWS IoT -palveluun lähettääkseen viestejä laitteille, sille täytyy luoda käytäntö, jossa on kyseiset oikeudet. Käytäntö luodaan IAM -palvelussa valitsemalla kyseiset oikeudet ja luomisen jälkeen se liitetään funktion suoritus rooliin. Kun funktio on luotu, kopioidaan funktion päätepiste ja liitetään se Alexa taitoon, sekä kopioidaan taidon tunniste. Funktiolle luodaan uusi laukaisin, valitaan Alexa älykoti ja liitetään taidon tunniste-kenttään. Nyt funktio on konfiguroitu ja taito kutsuu funktiota. Funktio voidaan ohjelmoida selaimella tai se voidaan tehdä muualla ja siirtää palveluun. Funktiossa luodaan ensin käsittelijä funktio kuvan 6 tapaan, joka toimii päätoimintona ohjelmalle ja sen sisälle muu ohjelma luodaan. Koska funktio on synkroninen, se ottaa

kaksi parametriä, request JSON-objektin, joka sisältää sille lähetetyn datan, sekä context objektin, jolla lähetetään vastaus käyttämällä sen succeed metodia.

4.2.1 Taustaprosessin pyyntöjen käsittely

Alexan lähettämien pyyntöjen data koostuu directive-kentästä ja taustaprosessin vastaukset koostuvat event-kentästä sekä halutessaan context-kentästä, joka sisältää tietoa laitteen tilasta, jota Alexa käyttää laitteen tilan näyttämiseen Alexa-sovelluksessa. Directive ja event kentät voivat sisältää kolme osiota, header, endpoint ja payload. Header-kenttä sisältää tunnistetietoja pyynnöstä, endpoint-kenttä sisältää laitteen ohjaamiseen tarvittavia tietoja ja payload sisältää pyyntöjen arvoja, kuten säätöarvoja ohjaus pyynnöissä. [8]

Lambda funktion täytyy ensimmäiseksi tarkistaa pyynnön sisältö. Pyynnön tyyppi löytyy sen JSON-datan directive.header.namespace osiosta. Jokaisella pyynnön tyypillä on yksi tai useampi toiminto, joka löytyy datan directive.header.name -kentästä. Esimerkiksi Alexa.PowerController tyypillä on TurnOn ja TurnOff toiminnot. Funktion täytyy pystyä käsittelemään löytöpyyntöjä ja jokaista laitteen ominaisuutta ja niiden toimintoja ohjaavaa pyyntöä. Kuvassa 6 on funktion osa, joka tarkistaa pyynnön tyypin ja kutsuu sitä vastaavaa metodia. [8]

```
exports.handler = function (request, context)
{
    var requestNamespace = request.directive.header.namespace;

    if (requestNamespace === 'Alexa.PowerController')
    {
        handlePowerControl(request, context);
    }
    else if (requestNamespace === 'Alexa.ColorController')
    {
        handleColorControl(request, context);
    }
    else if (requestNamespace === 'Alexa.BrightnessController')
    {
        handleBrightnessControl(request, context);
    }
    else if (requestNamespace === 'Alexa.Speaker')
    {
        handleVolumeController(request, context);
    }
    else if (requestNamespace === 'Alexa.SceneController')
    {
        handleSceneController(request, context);
    }
    else if (requestNamespace === 'Alexa.Discovery')
    {
        handleDiscovery(request, context);
    }
    else
    {
        context.fail();
    }
};
```

Kuva 6. Lambda funktion kutsun tyypin tarkistus node.js ympäristössä.

4.2.2 Löytöpyyntö

Löytöpyynnössä pyynnön tyyppi saa arvon Alexa.Discovery, toiminto saa arvon Discover ja sen muu sisältö on kuvan 7 mukainen. Datan payload.scope.token-kentässä on käyttäjän valtuutuskoodi, jota taustaprosessi käyttää tunnistamaan mitkä laitteet kuuluvat käyttäjälle. [8]

```
{
    "directive": {
        "header": {
            "namespace": "Alexa.Discovery",
            "name": "Discover",
            "payloadVersion": "3",
            "messageId": <Pyynnön tunniste koodi>
        },
        "payload": {
            "scope": {
                "type": "BearerToken",
                "token": <Valtuutuskoodi>
            }
        }
    }
}
```

Kuva 7. Löytöpyynnön datan sisällön muoto.

Löytöpyyntöön vastataan lähettämällä vastaava JSON-objekti kuin pyynnössä, jossa directive-kentän tilalle muutetaan event-kenttä, toiminnoksi muutetaan Discover.Response ja payload sisältää vain endpoints-taulukon, jossa on kuvattu laitteet ja niiden ominaisuudet. Kuvassa 8 on esitetty yhden valolaitteen kirkkauden säätö ja värin vaihto-ominaisuudet. EndpointId-kenttää käytetään laitteen tunnistamiseen, joten ohjauspyyntö lähetetään myös löytöpyynnön vastauksessa asetetuilla arvoilla. Koska laitteen varjon päivittämiseen tarvitaan vain laitteen nimi ja laitteiden nimet ovat yksilöllisiä, laitteen nimi voidaan asettaa endpointId-kentän arvoksi. DisplayCategories-kenttä määrittää laitteen tyypin ja Alexa käyttää sitä luokitteluun Alexa-sovelluksessa. Cookie-kenttään on mahdollista asettaa omia avain-arvo-pareja, jotka Alexa lähettää kyseisen laitteen ohjauspyyntöjen datan mukana. Taustaprosessi voi hyödyntää tätä dataa laitteen ohjaamisessa. Esimerkiksi cookie-kenttä voi sisältää laitteen sallimia maksimi- ja minimiarvoja. Capabilities-taulukossa on laitteen tukemat ominaisuudet ja ominaisuuksien konfiguroinnit. Laitteiden ominaisuuden properties-kentässä on konfiguroinnit laitteen ominaisuuden tilan päivittämiseen Alexa-sovelluksessa. ProactivelyReported-kentän arvon ollessa true Alexa olettaa, että sille lähetetään tieto laitteen tilasta sen muuttuessa ja retrievable kentän arvon ollessa true Alexa voi lähettää pyyntöjä, johon taustaprosessin vastaa lähettämällä laitteen tilan. Supported-kenttä kertoo mitä laitteen ominaisuuksien päivittämistä tuetaan. Tässä työssä ei ole käytetty kumpaakaan ominaisuutta, jolloin myös supported-kentän voi jättää kokonaan pois.

```
"payload": {
  "endpoints":[
    {
      "endpointId": "<unique ID of the endpoint>",
      "manufacturerName": "<the manufacturer name of the endpoint>",
      "modelName": "<the model name of the endpoint>",
      "description": "<a description that is shown in the Alexa app>",
      "friendlyName": "<device name, displayed in the Alexa app>",
      "displayCategories": ["LIGHT"],
      "cookie": {},
      "capabilities": [
        {
          "type": "AlexaInterface",
          "interface": "Alexa.BrightnessController",
          "version": "3",
          "properties": {
            "supported": [
              {
                "name": "brightness"
              }
            ],
            "proactivelyReported": true,
            "retrievable": true
          }
        },
        {
          "type": "AlexaInterface",
          "interface": "Alexa.ColorController",
          "version": "3",
          "properties": {
            "supported": [
              {
                "name": "color"
              }
            ],
            "proactivelyReported": true,
            "retrievable": true
          }
        }
      ]
    }
```

Kuva 8. Esimerkki löytöpyynnön vastauksen payload kentän sisällöstä.

Löytöpyyntöjen käsittely-funktio voidaan muodostaa kuvassa 9 näkyvällä tavalla, jossa endpoint-kentän data eli laitteet ja niiden ominaisuudet säilytetään DynamoDB-tietokannassa yhdessä laitteen omistavan käyttäjätunnuksen kanssa. Se mahdollistaa sen, että funktiossa voidaan ensin hakea valtuutuskoodia vastaava käyttäjätunnus ja sen jälkeen hakea laitteet, joiden käyttäjänä on kyseinen käyttäjätunnus. Lisäksi uusia laitteita on mahdollista lisätä muista sovelluksista.

Laitteet sisältävä taulu luodaan AWS-konsolista DynamoDB-palvelussa ja taulun avaimeksi voidaan valita endpointId, sillä se on yksilöllinen. DynamoDB tauluissa data voidaan säilyttää JSON-muodossa, joten niiden hakeminen on helppoa. Haku tietokannasta palauttaa laitteet valmiiksi taulukkona, jolloin niistä tarvitsee poistaa vain käyttäjätunnus ja ne voidaan liittää vastauksen endpoint-kenttään. Jotta haku tietokannasta voidaan suorittaa, funktio tarvitsee oikeudet muodostaa kyselyjä laitteita säilyttävään tauluun. Oikeudet annetaan IAM-palvelussa muodostamalla käytäntö DynamoDB-palveluun lukuoikeuksilla tauluun ja liittämällä se funktion rooliin. Haut tietokantaan tehdään käyttämällä AWS-ohjelmistokehityspaketin DynamoDB.documentClient-objektia ja sen scan-metodia. [13]

```
function handleDiscovery(request, context)
{
    var requestName = request.directive.header.name;
    if (requestName === "Discover")
    {
        var token = request.directive.payload.scope.token;
        cognito.getUser({AccessToken: token}, function(err, data){
            if(err)
            {
                console.log("ERROR: " + err);
                context.fail(err);
            }
            else
            {
                var userName = data.Username;
                var params = {
                    TableName: "Devices",
                    FilterExpression: "#user = :userName",
                    ExpressionAttributeNames: {
                        "#user": "user"
                    },
                    ExpressionAttributeValues: {
                        ":userName": userName
                    }
                };
                dynamodb.scan(params, function(err, data)
                {
                    if (err)
                    {
                        console.log("ERROR: " + err);
                        context.fail(err);
                    }
                    else
                    {
                        data.Items.forEach(function(item, index){
                            delete item["user"];
                        });

                        var payload = { "endpoints": data.Items };
                        var header = request.directive.header;
                        header.name = "Discover.Response";
                        context.succeed({ event: { header: header, payload: payload } });
                    }
                });
            }
        });
    }
    else
    {
        context.fail();
    }
}
```

Kuva 9. Löytöpyyntöjen käsittely metodi.

4.2.3 Ohjauspyyntö

Taustaprosessin lähetettyä vastaus löytöpyyntöön, kertoo se Alexa-pilvipalvelulle, että taustaprosessi osaa käsitellä ohjauspyyntöjä vastauksessa esitettyjen laitteiden ominaisuuksiin. Pyyntöjen perus rakenne on kaikilla samanlainen riippumatta ohjatusta

ominaisuudesta. Kuvassa 10 on laitteen käynnistyspyyntö, josta näkee ohjauspyyntöjen rakenteen. Ohjauspyynnön datassa directive-kenttä kertoo, mitä ominaisuutta halutaan ohjata ja directive.header-kenttä kertoo mitä laitetta halutaan ohjata. Lisäksi directive.payload-kenttä voi sisältää ohjaus arvoja, kuten kirkkauden suuruus. Directive.endpoint-kenttä sisältää endpointId ja cookie kentät, jotka saavat arvot mitkä on aikaisemmin määritelty löytöpyynnön vastauksessa. Ohjaus pyynnössä endpoint-kenttä sisältää valtuutuskoodin toisinkuin löytöpyynnössä.

```
{
    "directive": {
        "header": {
            "namespace": "Alexa.PowerController",
            "name": "TurnOn",
            "payloadVersion": "3",
            "messageId": <Pyynnön tunniste koodi>,
            "correlationToken": <Pyynnön viittaus koodi>
        },
        "endpoint": {
            "scope": {
                "type": "BearerToken",
                "token": <Valtuutuskoodi>
            },
            "endpointId": <Laitteen nimi>,
            "cookie": {}
        },
        "payload": {}
    }
}
```

Kuva 10. Ohjaus pyynnön esimerkki sisältö

Ohjauspyyntö käsitellään tarkastamalla pyynnön toiminto ja päivittämällä halutun laitteen varjon haluttu tila sen mukaiseksi. Kuvassa 11 on käsitelty virran kytkentä -ominaisuuden ohjaus-pyynnön käynnistys- ja sammutustoiminnot, jossa varjon päivitysmetodin parametrit määräytyvät toiminnon mukaan. Metodi tarvitsee laitteen nimen, joka saadaan pyynnön request.directive.endpoint.endpointId-kentästä ja varjon halutun tilan muutettavat kentät ja niiden arvot. Valtuutuskoodia ei välttämättä tarvita enää tunnistamaan ohjaavaa käyttäjää, sillä Alexa ei anna käyttäjän ohjata laitteita, joita löytöpyynnön vastaus ei sisältänyt.

```
function handlePowerControl(request, context)
{
    var requestName = request.directive.header.name;
    var params;

    if (requestName === "TurnOn")
    {
        params =
        {
            thingName: request.directive.endpoint.endpointId,
            payload: '{"state": {"desired": {"powered": true}}}'
        };
    }
    else if (requestName === "TurnOff")
    {
        params =
        {
            thingName: request.directive.endpoint.endpointId,
            payload: '{"state": {"desired": {"powered": false}}}'
        };
    }
    else
    {
        context.fail();
    }

    iotdata.updateThingShadow(params, function(err, data)
    {
        if (err)
        {
            console.log("ERROR: " + err);
            context.fail();
        }
        else
        {
            context.succeed(constructResponse(request));
        }
    });
}
```

Kuva 11. Laitteen virran kytkentä tilan ohjauspyyntöjen käsittely metodi

Kun varjo on päivitetty onnistuneesti metodi lähettää vastauksen, joka muodostetaan constructResponse metodissa. Vastauksen endpoint ja payload kentät ovat täysin samat kuin pyynnössä ja header-kenttä muutetaan kuvan 12 mukaiseksi. Event.header.correlationToken-kenttä sisältää viittaus koodin, jolla Alexa tunnistaa mikä vastaus on mihinkin pyyntöön. Eli vastauksen viittaus koodi täytyy olla sama kuin pyynnössä.

```
{
    "event"{
        "header": {
            "namespace": "Alexa",
            "name": "Response",
            "payloadVersion": "3",
            "messageId": <Vastauksen tunniste koodi>,
            "correlationToken": <Pyynnön viittaus koodi>
        },
        "endpoint": {
            "scope": {
                "type": "BearerToken",
                "token": <Valtuutuskoodi>
            },
            "endpointId": <Laitteen nimi>,
            "cookie": {}
        },
        "payload": {}
    }
}
```

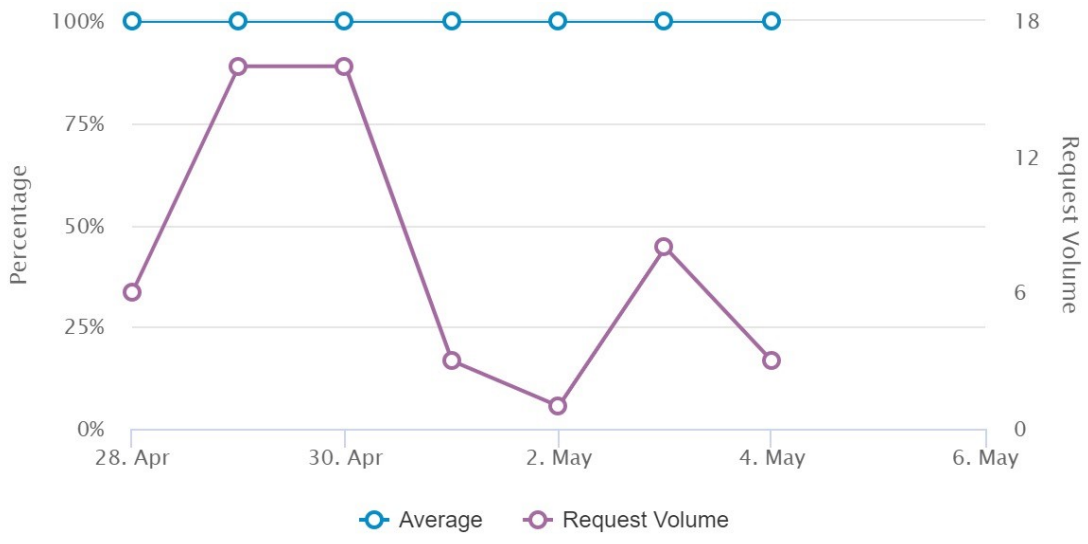
Kuva 12. Ohjaus pyynnön vastauksen rakenne

5 TESTAUS JA TULOKSET

Testauksen tarkoituksena on selvittää AWS:n ja Alexan yhteensopivuutta ja puheohjauksen käytettävyyttä. Käytettävyyden kannalta tärkein mittari on ohjauspyyntöjen latenssi ja AWS:n ja Alexan yhteen sopivuutta mitataan pyyntöjen onnistumisprosenttilla. Alexa monitoroi molempia mittareita, ja ne ovat nähtävissä Alexa-konsolissa. Testauksessa ei ole huomioitu Alexan puheentunnistusta, sillä se on järjestelmästä riippumatonta ja siihen ei työssä voida vaikuttaa. Puheentunnistuksen testaukseen liittyy myös paljon muuttujia, kuten puhujan kielitaito ja ympäristö. Järjestelmä, jota on testattu, sisältää kaksi laitetta, joiden ominaisuuksia ovat laitteen virran kytkentä, kirkkaus, väri ja äänen voimakkuus. Järstelmän tautaprosessina on käytetty AWS Lambda -funktiota ja siinä Node.js ajoympäristön versiota 12.x. Taustaprosessi sijaitsi us-east-1 -palvelimella ja AWS IoT järjestelmä eu-central-1 -palvelimella. Järjestelmää on testattu yleisessä käytössä viikon ja pyyntöjä tänä aikana lähetettiin 53 kappaletta. Testauksen tuloksena, puhekäyttöliittymä todettiin toimivaksi ja riittävän responsiiviseksi.

5.1 Pyyntöjen onnistuminen

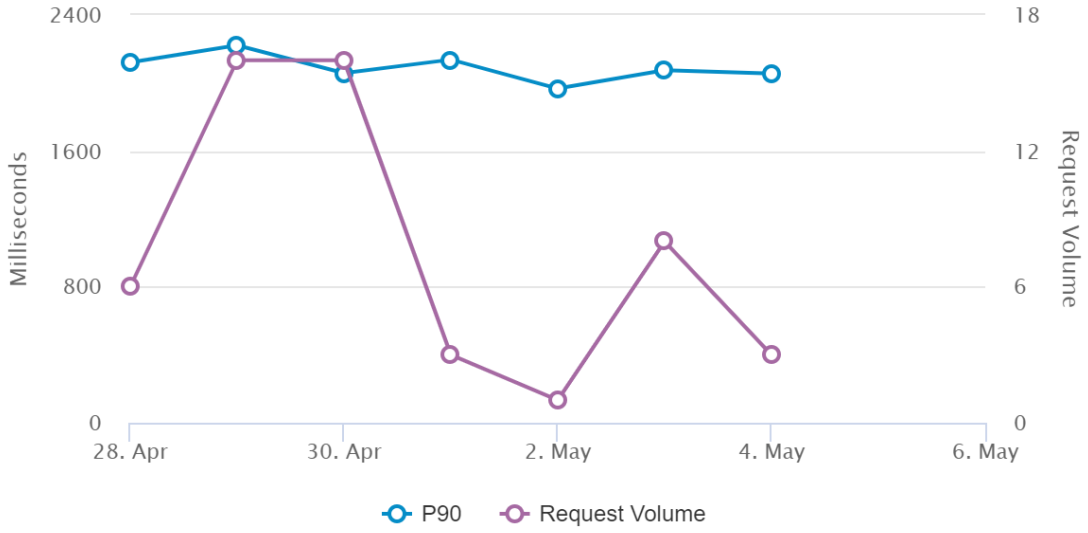
Kuvan 13 kuvaajasta nähdään, että ajanjaksona pyynnöt ovat onnistuneet täydellisesti. Mukana pyynnöissä on kuitenkin vain Alexan huomaamat pyynnöt, sillä puheentunnistus ei välttämättä huomaa tai tulkitsee väärin käyttäjän käskyjä.



Kuva 13. AWS IoT -järjestelmän puheohjauksen onnistumisprosentti.

5.2 Pyyntöjen latenssi

Kuvan 14 kuvaajasta saadaan pyyntöjen 90-persentiili latenssiksi noin kaksi sekuntia. Kaksi sekuntia on suhteellisen pitkä odotus aika, etenkin verrattuna graafiseen käyttöliittymään, mutta puhekäyttöliittymän hyötyjen vaikutusten takia, on latenssi siedettävä. Latenssiin vaikuttaa erityisesti järjestelmän AWS-palvelujen palvelin sijanti, joka voidaan optimoida valitsemalla sijanniksi sama kuin Alexan pilvipalvelun sijainti.



Kuva 14. AWS IoT -järjestelmän puheohjauksen latenssi.

6 YHTEENVETO

Opinnäytetyön tarkoituksena oli tutkia puhekäyttöliittymän liittämistä AWS IoT -järjestelmään ja sen toimivuutta.

Puhekäyttöliittymä luo käyttäjälle vapaan ja rajaamattoman tavan ohjata laitteita, sillä komentoja voi sanoa monella tavalla. Puhekäyttöliittymän tulee ottaa huomioon kaikki mahdolliset tavat ja siinä Alexa onnistuu hyvin mahdollistamalla valmiiden vuorovaikutusmallien käytön, jotta ohjelmistokehittäjän ei tarvitse ottaa huomioon jokaista tapaa erikseen. Alexan älykoti malli tarjoaa kattavan vuorovaikutusmallin IoT -laitteiden ohjaamiseen ja helpottaa käyttäjän laitteiden hallinnointia.

AWS:n ja Alexan modulaarisuus mahdollistaa puhekäyttöliittymän helpon integroimisen AWS IoT -järjestelmään. Kaikki tarvittavat teknologiat löytyvät AWS:n palveluista joka varmistaa järjestelmän osien yhteensopivuuden, skaalautuvuuden, helpon käytön ja tietoturvallisuuden. Usein puhekäyttöliittymän lisäksi halutaan käyttäjälle luoda myös graafinen käyttöliittymä tai muita sovelluksia, jotka on helppo yhdistää järjestelmään AWS:n avulla ja jotka toimivat hyvin yhteen myös puhekäyttöliittymän kanssa.

Työtä voisi laajentaa ottamalla mukaan Alexan laitteiden hallinta-ominaisuuksia, joita työssä ei käsitelty, kuten Alexa-sovelluksen graafisen käyttöliittymän, josta käyttäjä voi ohjata laitteita ja tarkistaa niiden tilan. Lisäksi voidaan tutkia järjestelmän laajentamista AWS IoT Greengrass -ympäristöön ja mitä hyötyjä tällä saavutetaan.

LÄHTEET

[1]IoT Analytics 2020. Cellular IoT & LPWA Connectivity Market Tracker 2010-25. Viitattu 8.2.2021
https://iot-analytics.com/wp/wp-content/uploads/2020/11/IoT-connections-total-number-of-device-connections-min.png

[2]Net Solutions 2020. What is the Difference Between AWS, Azure, and Google Cloud? Viitattu 8.2.2021
https://www.netsolutions.com/insights/aws-vs-azure-vs-google-cloud-comparison/

[3]Amazon Web Services Inc. 2020. AWS IoT: Developer Guide. Viitattu 28.1.2020
https://docs.aws.amazon.com/IoT/latest/developerguide/IoT-dg.pdf

[4]Amazon Web Services Inc. 2020. AWS Cognito: Developer Guide Viitattu 15.2.2020
https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-dg.pdf

[5]Amazon Web Services Inc. 2020. AWS Identity and Access Management: User Guide Viitattu 5.3.2020
https://docs.aws.amazon.com/IAM/latest/UserGuide/iam-ug.pdf

[6]Amazon.com Inc. 1996-2020. Alexa and Alexa Device FAQs. Viitattu 8.3.2020
https://www.amazon.com/gp/help/customer/display.html?nodeId=201602230

[7]Amazon.com Inc. 2010-2020. Understand the Smart Home Skill API. Viitattu 10.3.2020
https://developer.amazon.com/en-US/docs/alexa/smarthome/understand-the-smart-home-skill-api.html

[8]Amazon.com Inc. 2010-2020. Smart Home Skill API Message Reference. Viitattu 18.3.2020
https://developer.amazon.com/en-US/docs/alexa/smarthome/smart-home-skill-api-message-reference.html

[9]Amazon.com Inc. 2010-2020. Understand Account Linking for Alexa Skills. Viitattu 23.3.2020
https://developer.amazon.com/en-US/docs/alexa/account-linking/understand-account-linking.html

[10]Amazon Web Services Inc. 2020 Configuring a User Pool App Client Viitattu 5.4.2020
https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-app-idp-settings.html

[11]Amazon.com Inc. 2010-2020. Configure an Authorization Code Grant Viitattu 10.4.2020
https://developer.amazon.com/en-US/docs/alexa/account-linking/configure-authorization-code-grant.html

[12]Amazon Web Services Inc. 2020. AWS Lambda: Developer Guide. Viitattu 15.4.2020
https://docs.aws.amazon.com/lambda/latest/dg/lambda-dg.pdf

[13]Amazon.com Inc. 2010-2020 Steps to Build Smart Home Skill Viitattu 17.4.2020
https://developer.amazon.com/en-US/docs/alexa/smarthome/steps-to-build-a-smart-home-skill.html

[14]Amazon Web Services Inc. 2020. Class: AWS.DynamoDB.DocumentClient Viitattu 21.4.2020
https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/DynamoDB/DocumentClient.html