Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2020

Eppu Peltonen

ANTURIDATAN TALLENNUKSEN TOTEUTTAMINEN IOTA-JÄRJESTELMÄÄN

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2020 | 26 sivua

Eppu Peltonen

ANTURIDATAN TALLENNUKSEN TOTEUTTAMINEN IOTA-JÄRJESTELMÄÄN

Tämän opinnäytetyön tavoitteena oli perehtyä IOTA-järjestelmään sekä rakentaa ympäristö, jossa kerätään anturidataa reaalimaailmasta ja lähetetään sitä salattuna IOTA:n Tangle-verkkoon MAM-teknologian avulla.

Työn alussa selvitettiin teoriaa IOTA:sta ja Tanglesta, minkä jälkeen rakennettiin tavoitteiden mukainen ympäristö. Ympäristössä käytettiin Raspberry Pi 4 -tietokonetta, digitaalista DHT11-lämpötila- ja kosteusanturia, sekä Node.js-ohjelmaa. Ohjelmoinnissa käytettiin IOTA:n mam.js-ohjelmointikirjastoa. Lisäksi konfiguroitiin Tanglen IRI-noodi pilvipalvelimelle.

Tuloksena on prototyyppi anturidatan tallennuksesta IOTA-järjestelmään, sekä pohdintaa vaihtoehtoisista teknologioista, joita työssä voisi käyttää.

ASIASANAT:

IOTA, Tangle, anturidata, IoT, lohkoketjut

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Informations and Communications Technology

2020 | 26 pages

Eppu Peltonen

IMPLEMENTATION OF SENSOR DATA BROADCASTING TO IOTA ECOSYSTEM

The goal for this thesis was to explore IOTA system and Tangle network from a theoretical perspective first and then implement a prototype where real world sensor data would be gathered and sent to Tangle using Masked Authenticated Messaging (MAM) technology.

The thesis began by studying IOTA and Tangle and the different technologies they use. Based on the acquired information, a prototype was built using Raspberry Pi 4, digital DHT11 temperature and humidity sensor, Node.js application and IRI Tangle software. The end result was a working prototype which sends a data package to Tangle that contains the temperature and humidity data. The thesis also discusses alternative technologies and hardware that could be used.

KEYWORDS:

IOTA, Tangle, sensor data, IoT, blockchain

SISÄLTÖ

KUVAT

SANASTO

DAG	Suunnattu syklitön verkko eli DAG (*directed acyclic graph*) on solmuista ja linkeistä koostuva verkko, jossa suunnattuja linkkejä (yhteydet solmujen välillä) pitkin ei ole mahdollista saapua lähtöpisteeseen. Solmun B esivanhempia ovat ne solmut jotka päättyvät pisteeseen B. Solmun A jälkeläisiä ovat ne solmut jotka alkavat pisteestä A. [1]
Hajautettu tilikirja	Hajautettu tilikirja (engl. *distributed ledger*) käyttää itsenäisiä tietokoneita transaktioiden tallentamiseen, jakamiseen ja synkronoimiseen sähköisten tilikirjojen välillä, toisin kuin perinteiset tilikirjat, joiden data on keskitetty. [2]
MAM	Masked Authenticated Messaging (*MAM*) on kommunikointiprotokolla joka mahdollistaa salattujen datavirtojen lähettämisen ja hakemisen Tanglessa. [3]
Noodi	Noodi *(*engl*. node)* on Tangle verkossa oleva palvelin, jossa ylläpidetään jotakin Tanglen ohjelmistoa. Noodin tehtävänä on vahvistaa transaktioita, tallettaa niitä paikalliseen tilikirjaan, ja kommunikoida asiakasohjelmien kanssa. [4]
Solmu	Solmu on DAG -verkon solmukohta, joka tarkoittaa Tangle verkkoon lisättyä uutta transaktiota. [5]

1 JOHDANTO

Internetiin yhdistettyjen laitteiden määrän ennustetaan ylittävän 75 miljardin rajan vuonna 2025. Laitteiden kirjo on laaja ja niitä on moniin eri tarkoituksiin: maanteihin asennettavia pieniä antureita, puettavaa elektroniikkaa, älypuhelimia, ja paljon muuta. On selvää että päivä päivältä maailma on entistä enemmän verkossa. Jo nyt kaikkien laitteiden yhteenlaskettu tiedon ja datan määrä on valtava. Vuoden 2016 loppuun mennessä internetliikenteen määrä oli 1,2 ZB vuodessa. Seuraavien viiden vuoden aikana, maailmanlaajuisen internetliikenteen odotetaan viisinkertaistuvan, eli vuonna 2021 kuukausittainen internetliikenne saavuttaa 31 GB:n rajan käyttäjää kohden. [6]

Jatkuvassa kasvussa on kuitenkin ongelma. Samassa ajassa kun internetliikenteen ennustetaan viisinkertaistuvan, laajakaistojen nopeuksien odotetaan vain tuplaantuvan, mikä johtaa globaalilla tasolla kaistojen tukkeutumiseen. Ei ole yksinkertaisesti mahdollista, että kaikki laitteet yhdistettäisiin keskitettyihin datasiiloihin, eikä datasiiloissa olevilla analytiikkamoottoreilla ole mahdollisuuksia muuntaa raakadatasta hyödyllistä tietoa laitteille reaaliajassa. Yhtenä ratkaisuna tarjotaan sumu- tai usvalaskentaa (engl. *fog computing*, *mist computing*), mutta käytännön ongelmaksi koituu resurssien käyttöönotto laitteiden välisen talouden eri toimijoiden kesken, joita voi olla jopa satoja, ellei tuhansia. IOTA:n ratkaisu ongelmaan on kryptovaluutta ja datasiirtoväylä, joka ei perustu lohkoketjuun, vaan Tangle-verkkoon. Maksuttomien rahasiirtojen ansiosta laitteet jakavat dataa reaaliajassa hajautetussa verkossa, jolloin datasiiloihin ei pääse kertymään suuria datatukoksia. [6]

Opinnäytetyön tavoitteena on tutustua IOTA:n Tangle-verkkoon. Lisäksi tavoitteena on rakentaa ympäristö, jossa kerätään anturidataa reaalimaailmasta ja lähetetään sitä Tangleen. Aluksi selvitetään mitä tarkoittaa IOTA, Tangle ja niiden käyttämät eri teknologiat, minkä jälkeen siirrytään toteutusvaiheeseen. Lopuksi tarkastellaan tavoitteita ja pohditaan jatkokehitysideoita. Työn ulkopuolelle rajataan IOTA:n kryptovaluutta, koska työ keskittyy anturidatalähetyksiin.

2 IOTA

IOTA on kryptovaluutta ja datasiirtoväylä, jonka kohteena on IoT (*Internet of Things*). IOTA pohjautuu hajautettuun tilikirja teknologiaan, Tangleen. Tangle mahdollistaa uuden tavan konsensuksen saavuttamiseen hajautetussa vertaisverkossa (engl. *peer-to-peer*), ja tarjoaa ratkaisun lohkoketjun heikkouksille. IOTA:lla laitteet ja ihmiset voivat tehdä transaktioita maksuttomasti hajautetussa ympäristössä, joka mahdollistaa mikro- ja nanomaksut ilman välittäjää. [7]

2.1 IOTA Foundation

IOTA Foundation on voittoa tavoittelematon säätiö, joka perustettiin Saksassa vuonna 2017.

Säätiön tavoitteena on

- tutkia ja turvata IOTA:n protokollakerrosta sekä luoda uutta tietoa hyödyttämään IOTA ekosysteemiä
- kehittää tuotantovalmiita ohjelmistoja yhteisölle, yhtiökumppaneille ja ekosysteemille
- opettaa ja edistää teknologioita ja esimerkkitapauksia uusille sukupolville. [8]

2.2 Tangle

Tangle on IOTA:n keskeinen tietorakenne. Tangle perustuu DAG:iin (engl. *directed acyclic graph*), jossa solmut ovat transaktioita ja solmujen välillä oleva yhteys tarkoittaa, että solmut ovat hyväksyttyjä transaktioita. Tanglen etu lohkoketjuun on sen skaalautuvuus. Lohkoketjussa transaktioiden luomistiheys on rajattu, koska haarautuvat ketjut hylätään, toisin kuin Tanglessa, jossa haarautuvat ketjut voidaan lopulta yhdistää. Tämän ansioista Tanglella on suurempi suorituskyky kuin lohkoketjulla. [5]

Kun uusi transaktio luodaan, siitä tehdään uusi solmu Tangleen. Solmu liittyy aina kahteen edelliseen solmuun. Solmujen valinta suoritetaan Markov Chain Monte Carlo Random Walk -algoritmilla. Transaktion vahvistus edellyttää, että kaksi valittua transaktiota ovat vahvistettu. Tosiasiassa vahvistus tarkoittaa sitä, että transaktioihin liitettyjen tilien saldo on positiivinen. [5]

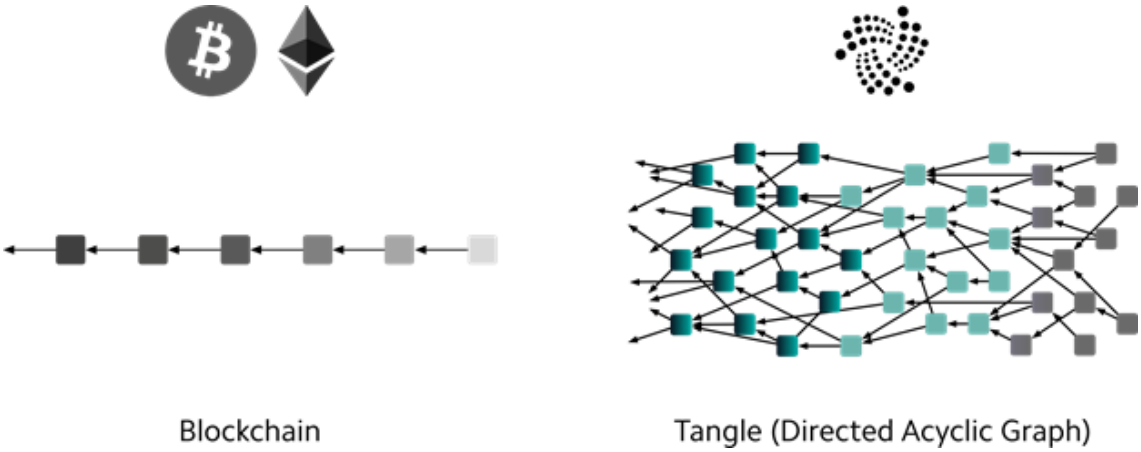
Kun riittävän moni transaktiota uudempi transaktio on vahvistanut sen, voidaan todeta, että transaktio on osa konsensusta, jonka jälkeen sitä on mahdotonta muunnella. [5]

2.2.1 Tanglen ja lohkoketjun ero

Tangle ja lohkoketju ovat kryptovaluuttojen taustalla olevia tietorakenteita, joihin valuuttojen toiminta perustuu. Vaikka niiden periaate on sama, niiden välillä on eroavaisuuksia.

Tietorakenne

Lohkoketju on peräkkäisistä lohkoista koostuva ketju, jossa jokainen lohko viittaa edelliseen lohkoon kronologisessa järjestyksessä. Tanglessa jokainen yksittäinen transaktio viittaa kahteen edelliseen transaktioon ja muodostaa suunnatun syklittömän verkon. Tanglen rakenne mahdollistaa samanaikaisen, asynkronisen kasvun, kun taas lohkoketju laajenee lineaarisesti. Kuvassa 1 havainnollistetaan rakenteiden eroja. [9]



Kuva 1. Lohkoketju ja Tangle. [9]

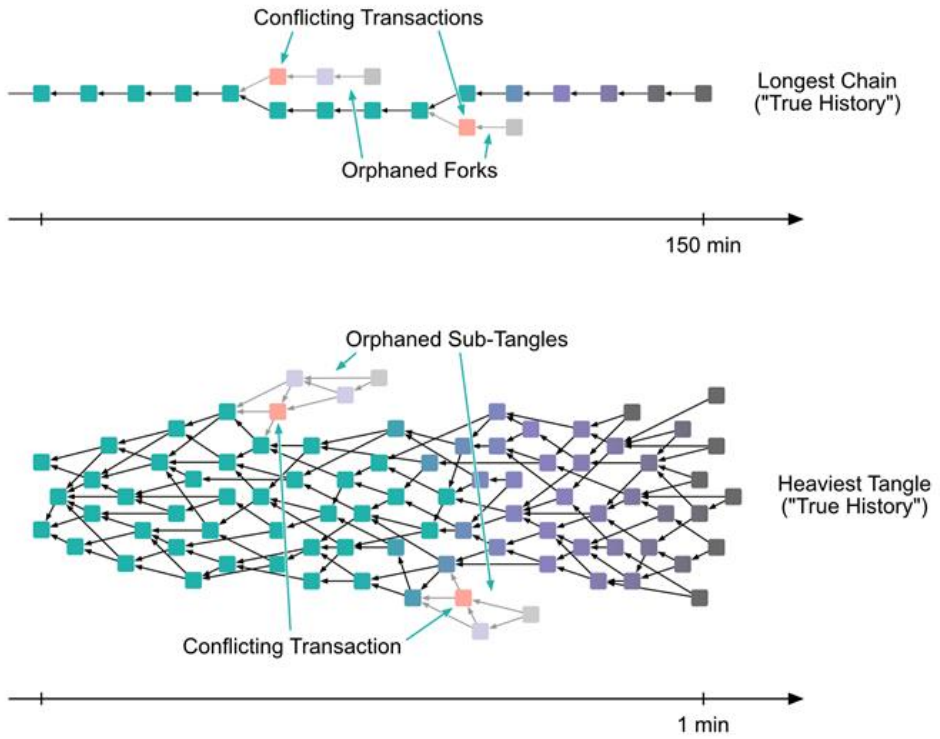
Konsensus

Lohkoketjun konsensusmekanismi perustuu pisimpään ketjuun (engl. *longest chain*). Louhijoiden tavoitteena on lisätä seuraava lohko ketjuun, ja kannustimena tälle he saavat ”lohkopalkinnon” eli uutta valuuttaa. Seuraavan lohkon lisääjä arvotaan työtodistuksella (engl. *proof of work*), jossa laskentatehoa käyttämällä ratkotaan kryptografisia tehtäviä. Koska useampi tietokone eli louhija voi ratkaista tehtävän samaan aikaan, ja

täten luoda uuden lohkon samanaikaisesti, tarvitaan konsensus, jossa osapuolet sopivat mikä ketju on validi. [9]

Haarakohdaksi kutsutaan tilannetta, jossa louhija huomaa kaksi validia lohkoa, jotka viittaavat samaan lohkoon. Louhijan on tällöin valittava kummalle puolelle uusi lohko yritetään lisätä. Konsensus saavutetaan valitsemalla pidempi ketju. Tämä tarkoittaa sitä, että transaktiota ei voi pitää varmennettuna ennen kuin riittävän monta lohkoa on rakennettu sen päälle. Yleissääntönä pidetään, että kuuden lohkon jälkeen transaktiota voidaan pitää varmennettuna. [9]

IOTA:ssa konsensusmekanismina on suurin Tangle. Transaktion lopullinen varmistus tapahtuu nopeammin kuin lohkoketjussa, koska Tanglessa transaktioiden liikkeelle-lasku on suhteessa verkon validointiin. Mitä enemmän Tanglessa on aktiivisuutta, sitä enemmän transaktioita vahvistetaan, joten transaktioiden vahvistus on nopeampaa. Mikäli Tanglessa tapahtuva huijausyritys paljastuu, kelvottomat haarat unohdetaan Tanglesta. Kuvassa 2 havainnollistetaan lohkoketjun ja Tanglen konsensusmekanismien eroja. [9]



Kuva 2. Lohkoketjun ja Tanglen konsensusmekanismien ero. [9]

Turvallisuus

IOTA:ssa transaktion tekemiseen tarvitaan työtodistus. Lohkoketjussa työtodistusta käytetään hajautettuna arvontamekanismina toisin kuin Tanglessa; jossa se toimii ainoastaan roskapostin ja hyökkäysten estämisenä. Kun transaktioiden määrä kasvaa, työtodistusten määrä lisääntyy kumulatiivisesti, joka lisää Tanglen turvallisuutta hyökkäyksiä vastaan. Käyttäjämäärien kasvaessa Tanglen turvallisuus kasvaa. Pienen laskentatyön lisääminen transaktioon on halpaa yksilön näkökulmasta, mutta hyökkäyksen tekijälle kaikkien transaktioiden tekemä yhteenlaskettu työ on vaikea päihittää. [9]

Siirtokulut

Tanglessa ei ole siirtokuluja, koska Tanglessa ei ole louhintaa, ja transaktioiden vahvistus on olennaisessa roolissa. Lähetty summa vastaa aina vastaanotettua summaa. Tämä mahdollistaa niin sanotut mikromaksut jotka ovat koneiden välisen talouden (engl. *M2M economy*) kasvava tarve. IOTA:ssa kaikki transaktioita tekevät osapuolet ovat myös aktiivisesti osana konsensusta. Lohkoketjuihin perustuvissa kryptovaluutoissa on kahtiajako louhijoiden ja järjestelmän käyttäjien välillä: louhijat haluavat hitaammat transaktioiden vahvistusajat ja korkeammat siirtokulut, kun taas käyttäjät haluavat juuri päinvastoin. IOTA:ssa kaikkien osapuolten edut kohtaavat. [9]

2.3 Transaktio

Transaktion tekeminen koostuu neljästä vaiheesta. Ensimmäisenä allekirjoitetaan syöte ja luodaan viesti. IOTA:ssa on kahdenlaisia transaktioita. Transaktioita, joissa siirretään IOTA valuuttaa, ja transaktioita joissa siirretään vain viesti tai dataa (engl. *zero value transaction*). Valuuttasiirrot vaativat digitaalisen allekirjoituksen valuutan omistajan todentamiseen. Pelkissä viesti- tai datasiirroissa digitaalista allekirjoitusta ei vaadita. [10]

Kun viesti on valmis, valitaan kärki ja vahvistetaan transaktio. Kärki (engl. *tip*) on transaktio, jota ei ole vielä vahvistettu. Kärjen valinta on prosessi, jossa kaksi kärkeä valitaan satunnaisesti käyttämällä kärjen valinta-algoritmia (engl. *Markov Chain Monte Carlo Random Walk -tip selection algorithm*). Kun kaksi kärkeä on valittu, ne vahvistetaan tarkastamalla että molempien historia on johdonmukainen ja kaksinkertaista kulutusta tai muita huijaamisen keinoja ei ole käytetty. [10]

Kärjen valinnan ja vahvistuksen jälkeen transaktiolta edellytetään työtodistus, jossa laskentatehoa käyttämällä ratkotaan yksinkertaisia kryptografisia tehtäviä. Viimeisenä transaktio lähetetään manuaalisesti asetetuille naapurinoodeille. [10]

Kun transaktio on lähetetty, satunnaisesti valittu transaktio valitsee sen vahvistettavaksi. Kun riittävä määrä uusia transaktioita eli kärkiä viittaa transaktioon joko suoraan tai epäsuoraan, transaktiota voidaan pitää vahvistettuna. [10]

2.4 Kärjen valinta-algoritmi

Uutta, vahvistamatonta transaktiota Tanglessa kutsutaan kärjeksi. Kun uusi transaktio saapuu Tangleen, sen tulee vahvistaa kaksi edeltävää transaktiota [11]. Kärjen valinta suoritetaan algoritmilla, joka käyttää Markovin ketjua, Monte Carlo -algoritmia sekä satunnaiskulkua.

Markovin ketju on stokastinen eli sattumanvaraisesti etenevä malli, joka kuvaa peräkkäisiä tapahtumia, jossa tapahtuman todennäköisyys riippuu ainoastaan edellisestä tapahtumasta. Todennäköisyysteoriassa Markovin prosessiksi kutsutaan prosessia, joka täyttää Markovin ominaisuuden. Prosessi täyttää Markovin ominaisuuden, jos prosessin tulevaisuuden voi ennustaa prosessin nykytilasta yhtä hyvin kuin tuntisi prosessin historian. [12]

Monte Carlo -algoritmi on satunnaisalgoritmi, jonka tulos voi olla väärä pienellä todennäköisyydellä. Monte Carlo algoritmit ovat nopeita, ja niitä usein toistetaan peräkkäin simulaatiossa, jolloin virheen todennäköisyyttä voidaan hallita. Satunnaiskulku on stokastinen prosessi, joka kuvaa onnistuneita askelia jossakin matemaattisessa tilassa, kuten vaikka DAG:ssa. [12]

Kun nämä kolme algoritmia yhdistetään, saadaan onnistunut satunnaiskulku, joka kuvaa DAG:ssa kuljettua polkua kohti kärkeä. [12]

2.5 Masked Authenticated Messaging

Masked Authenticated Messaging on kommunikointiprotokolla, joka mahdollistaa salattujen datavirtojen lähettämisen ja hakemisen Tanglessa. IOTA:n datan eheyden

mahdollistavan konsensusmekanismin kanssa, se täyttää aukon toimialoilla, joissa datan eheys ja yksityisyys korostuvat. [3]

IOTA:ssa käyttäjä voi lähettää viestejä milloin tahansa. Työtodistus vaaditaan, jotta viestin voi julkaista Tangleen. Tilaaja vastaanottaa viestin, jos kyseinen noodi kuuntelee reaaliajassa osoitetta josta viesti lähtee. Nämä viestit voivat olla eri kokoisia. Tosin, pikaisesti arvioimalla voidaan todeta, että pienemmät viestien koot saavuttavat potentiaalisesti paremman datan eheyden. Esimerkiksi käyttäjä voi lähettää kryptatun 4K videon MAM:in avulla, mutta se täyttää verkon kaistanleveyden ja johtaa korkeaan latenssiin käyttäjän puolella. [3]

Sekä valuuttasiirrot että datalähetykset ovat osallisena kumulatiivisen turvallisuuden luomiseen, koska molemmat kuuluvat hajautettuun tilikirjaan. Molemmat hyötyvät verkon eheydestä, kun muut transaktiot viittaavat niihin epäsuorasti. [3]

MAM käyttää Merkle-puuhun pohjautuvaa allekirjoitusmallia. Puumallin juurta käytetään kanavan identifioivana tunnisteena. Jokainen viesti sisältää seuraavan puun juuren, koska puita säilytetään vain väliaikaisesti. Viestit salataan OTP-salausmenetelmällä *(*engl*. one-time pad),* joka sisältää kanavan ID:n ja salausavaimen indeksin. Tästä saatava tiivistefunktio allekirjoitetaan yksityisellä avaimella. Kryptattu sisältö, allekirjoitus ja puun jälkeläiset julkaistaan Tangleen, jossa kuka tahansa voi oikealla avaimella avata viestin. [3]

MAM-protokollaa voidaan käyttää julkisesti, yksityisesti, tai rajatusti. Julkisessa lähetyksessä puun juurta käytetään viestin osoitteena. Tällöin kuka tahansa viestin löytäjä voi avata viestin käyttämällä osoitetta. Julkista lähetystä voidaan käyttää esimerkiksi julkisiin ilmoituksiin joltakin laitteelta. Etuna tässä on muihin vastaaviin alustoihin verrattuna datan eheys. [3]

Yksityisissä lähetyksissä Merkle-puun juuren tiivistettä käytetään viestin osoitteena. Näin ollen vain puun juurella voi avata viestin. Rajatuissa lähetyksissä yksityiseen lähetykseen lisätään salasana. Tällöin rajatun lähetyksen osoitteena käytetään salasanan tiivistettä sekä Merkle-puun juurta. [3]

3 IRI-NOODI

Työn vaatimuksena oli ottaa käyttöön pilvipalvelin, johon asennetaan IOTA:n IRI-sovellus. Kun sovellus on käynnissä, siihen otetaan yhteys erillisellä Node.js-ohjelmalla, joka lähettää anturidatapaketin IRI:lle tarkastettavaksi. Pilvipalvelimeksi valittiin Amazon AWS EC2, johon asennettiin Ubuntu 18.04 Server -käyttöjärjestelmä.

Ubuntulle asennettiin IRI sekä apuohjelmia monitorointiin ja naapurinoodien hallintaan. IRI sekä apuohjelmat asennettiin Linuxin systemd-prosessinhallintajärjestelmään, joka käynnistää ohjelmat automaattisesti palvelimen uudelleenkäynnistyksen yhteydessä.

3.1 Amazon Web Services

Amazon Web Services (AWS) on laaja valikoima internetissä toimivia pilvipalveluita, jotka muodostavat pilvilaskenta-alustan. AWS sisältää 175 eri palvelua laskentaan, tietokantoihin ja tallennustilaan. Pilvilaskennalla tarkoitetaan IT-resurssien toimittamista internetin välityksellä, ja yleensä käyttöön perustuvalla laskutuksella. [13]

Pilvilaskenta mahdollistaa monia etuja fyysiseen palvelinkeskukseen nähden. Esimerkiksi pienten startup yrityksien ei tarvitse tehdä suurta alkuinvestointia fyysisiin laitteisiin, vaan yritys voi valita mieleisensä palvelut heti käytettäväksi kohtuullisella hinnalla. Pilvipalveluiden käyttöönotto ja poistaminen, sekä resurssien skaalautuvuus on käytännöllisempää kuin fyysisessä palvelinkeskuksessa. [13]

3.2 Ubuntu

Ubuntu on ilmainen avoimen lähdekoodin Linux jakelupaketti, joka pohjautuu Debian jakelupakettiin. Linux jakelupaketilla tarkoitetaan Linux ydintä käyttävää avoimen lähdekoodin käyttöjärjestelmää. Ubuntua julkaistaan työpöytä- ja palvelinversioina, sekä IoT -laitteille suunnattuna Ubuntu Core -versiona. Ubuntua kehittää Canonical sekä joukko muita ohjelmistokehittäjiä. Uusi käyttöjärjestelmäversio julkaistaan puolen vuoden välein, ja LTS (*long-term support*) -versio kahden vuoden välein. LTS-versiolle taataan usean vuoden tuki ja päivitykset. Viimeisin LTS-versio 20.04 julkaistiin 23. huhtikuuta 2020, ja käyttöjärjestelmälle taattu tuki kestää vuoteen 2025. [14]

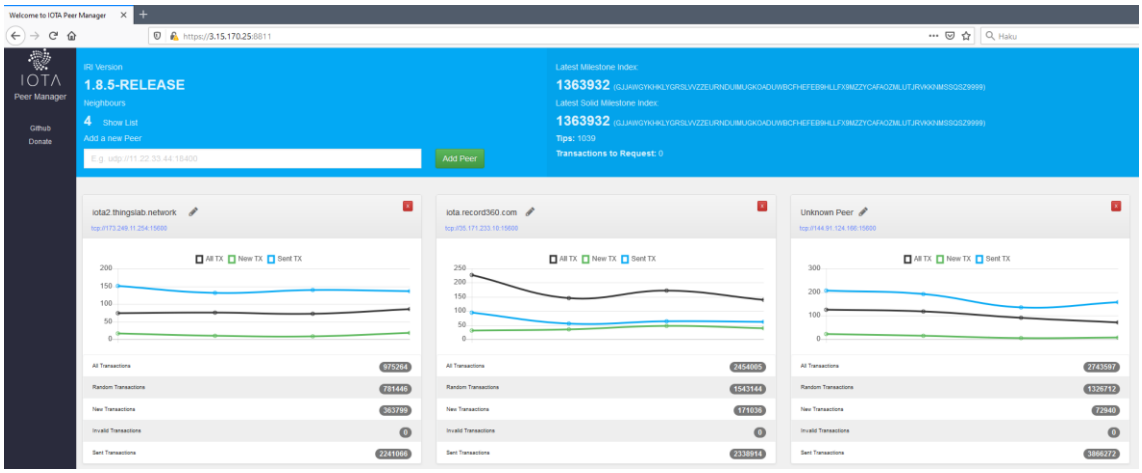
3.3 IRI

IOTA Reference Implementation (IRI) on avoimen lähdekoodin Java-ohjelma, eli IRI-noodi (engl. *IOTA Full Node*). IRI:n tehtävänä on vahvistaa transaktioita, tallettaa niitä tilikirjaan, ja vastaanottaa ja käsitellä asiakasohjelmien pyyntöjä [4]. Tässä työssä IRI:lle lähetetään Raspberry Pi -tietokoneelta salattu datapaketti MAM-kommunikointi-teknologialla, jonka IRI vahvistaa naapurinoodien avulla.

3.4 Palvelut

IRI:n yhteyteen asennettiin myös Grafana sekä IOTA Peer Manager. Grafana on avoimen lähdekoodin analytiikkaohjelma, joka tarjoaa reaaliaikaista datan visualisointia. Grafana asennettiin palvelimelle, ja siellä ajetaan valmista IOTA-näkymää, joka visualisoi palvelimen suorituskykyä, ja näyttää IRI:n reaaliaikaisen tilanteen. [15]

IOTA Peer Manager on Node.js-ohjelma IRI:n naapurinoodien seurantaan ja hallintaan. Kuvassa 3 on IPM-hallintapaneeli, josta seurataan transaktioiden liikkumista naapurinoodien välillä. [16]



Kuva 3. IOTA Peer Manager

4 ANTURIDATAN LÄHETYS

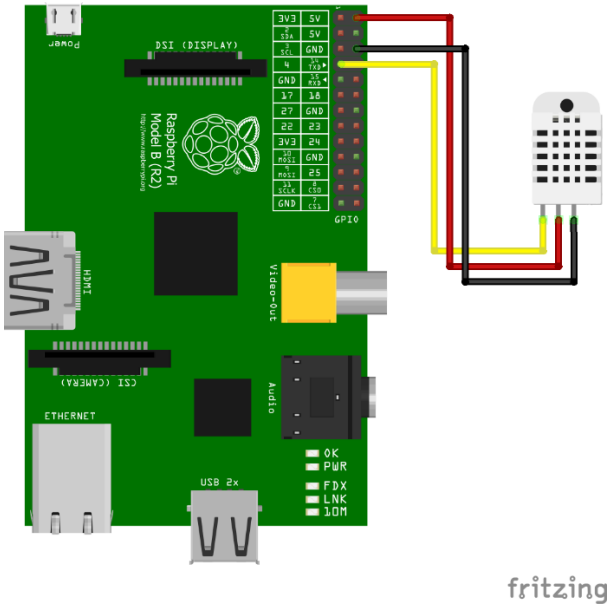
Työn alussa määritettiin vaatimukset käytettäville laitteille ja teknologioille. Anturiksi valittiin DHT11-lämpötila- ja kosteusanturi. Vaatimuksena oli mitata dataa reaalimaailmasta, joten ympäristöstä mitattu lämpötila- ja kosteusdata soveltui tähän tarkoitukseen hyvin. Anturi kytkettiin Raspberry Pi 4 -tietokoneeseen, jolla anturia kontrolloidaan, ja josta data lähetetään Tangleen. Raspberry Pi ei tue analogisia sisääntuloja, joten työhön valittiin digitaalinen anturi. Vaihtoehtona olisi käyttää analogista anturia A/D -muuntimen kanssa. Lisäksi tietokoneen voisi korvata esimerkiksi STMicroelectronicsin STM32-tuoteperheen kehitysalustalla. Työssä päädyttiin käyttämään Raspberry Pi:tä datan lähetystä vaativan Node.js-ohjelman takia.

Työn tekovaiheen aikana IOTA Streams -ohjelmistokehyksestä julkaistu alfaversio korvaa nykyisen MAM-teknologian ja tekee datalähetyksien tekemisestä helpompaa. Kehys on toteutettu Rust-ohjelmointikielellä, ja sillä voisi korvata kokonaan työssä käytetyn Node.js-ohjelman. Myös Streams sisältää ohjelmointirajapinnan, jolla voidaan kommunikoida IRI:n kanssa.

4.1 DHT11-lämpötila- ja kosteusanturi

DHT11 on digitaalinen lämpötila- ja kosteusanturi. Anturi käyttää kapasitiivista kosteusanturia kosteuden mittaamiseen ja termistoria lämmön mittaamiseen. Anturissa on myös sisäänrakennettu A/D muunnin joka muuntaa analogisen sisääntulosignaalin digitaaliseksi ulostulosignaaliksi. Kuvassa 4 kytkentäkaavio anturin liittämisestä Raspberry Pi 4 -tietokoneeseen. [17]

Anturin ohjaamiseen käytettiin node-dht-sensor -node.js-kirjastoa. Kirjastoa varten Raspberry Pi:lle asennettiin BCM2835 C-ohjelmointikirjasto, joka tarjoaa pääsyn GPIO-pinneihin ja Raspberry Pi:n BCM2835-mikroprosessoriin. [18]



Kuva 4. Kytkentäkaavio. Keltainen dataliitin kytkettynä Raspberry Pi:n GPIO 4 pinniin, punainen virtaliitin kytkettynä 5V pinniin, ja musta maaliitin kytkettynä Raspberryn maadoituspinniin (GND).

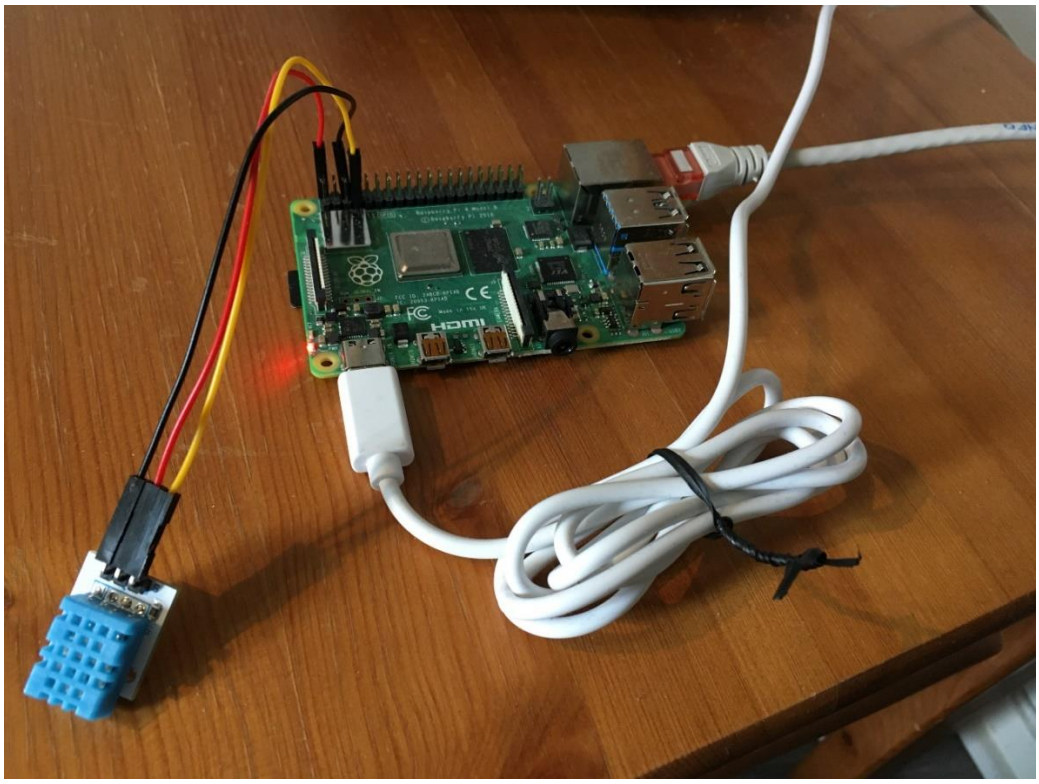
4.2 Raspberry Pi

Raspberry Pi on Raspberry Pi Foundationin valmistama sarja yhden piirilevyn tietokoneita. Ensimmäinen malli, Raspberry Pi 1 Model B julkaistiin vuonna 2012 ja sen jälkeen säätiö on julkaissut useita eri malleja. Tässä työssä käytetään vuonna 2019 julkaistua Raspberry Pi 4 Model B mallia. [19]

Raspberry Pi:ssä on Linux pohjainen avoimen lähdekoodin Raspbian käyttöjärjestelmä, jonka lisäksi siinä on myös laaja valikoima GPIO (*general purpose input/output*) pinnejä joihin voi liittää erilaisia komponentteja. Raspberry Pi:tä käytetään usein ohjelmoinnin opetteluun, elektroniikkaprojekteihin ja erilaisiin IoT -sovelluksiin [19].

Raspberry Pi 4:n käyttöönotto aloitettiin asentamalla microSD-kortille Raspbian käyttöjärjestelmä. Seuraavaksi otettiin käyttöön SSH-yhteys, jolla tietokonetta voi käyttää terminaalin välityksellä toiselta tietokoneelta internet-yhteyden välityksellä. Etuna SSH-yhteydessä on se, että erillistä monitoria, hiirtä tai näppäimistöä ei tarvita. Tässä työssä tietokonetta käytettiin Windows PC:ltä puTTY-ohjelmalla. Yhteyden muodostamisen jälkeen tietokoneelle asennettiin tarvittavat työkalut, ohjelmat ja kirjastot. Lopuksi kiinnitettiin DHT11-anturi ja ohjelmoitiin lyhyt testausohjelma, jolla varmistettiin anturin toiminta.

Node.js-ohjelma kirjoitettiin Windows PC:llä ja tallennettiin github-palvelimelle, josta se haettiin Raspberry Pi:lle ajettavaksi. Kuvassa 5 Raspberry Pi 4 -tietokone liitettynä verkkoon ja anturi kytkettynä GPIO-pinneihin.



Kuva 5. DHT11 anturi kytkettynä Raspberry Pi 4 -tietokoneeseen.

4.3 Node.js-ohjelma

Node.js on avoimen lähdekoodin tapahtumapohjainen Javascript -ajoympäristö, jolla luodaan asynkronisia web-sovelluksia. Node.js ei käytä säikeitä rinnakkaisuuden luomiseen vaan takaisinkutsufunktioita (engl. *callback function*). Takaisinkutsufunktiolla ohjelma ei pysähdy odottamaan hidasta tehtävää, vaan pyytää vastauksen myöhemmin ja suorittaa muita tehtäviä sillä välin. Tämä on tärkeää WWW-palvelimilla jotka vastaanottavat satoja tai tuhansia pyyntöjä, jotka saattavat sisältää ohjelman pysäyttäviä tehtäviä. Node.js käsittelee pyynnöt asynkronisesti takaisinkutsufunktiolla, joka parantaa ohjelman toimivuutta huomattavasti. [20]

Tässä työssä ohjelmoitiin Node.js-ohjelma kontrolloimaan anturia, sekä lähettämään ja hakemaan anturidata Tanglesta. Aluksi ohjelmassa alustetaan asennetut kirjastot ja moduulit (Kuva 6).

IOTA:n ohjelmakirjastot sekä anturin lukemiseen käytetty node-dht-sensor -kirjasto asennettiin npm (*node package manager*) -sovelluksella. Node-dht-sensor -kirjastoa varten asennettiin myös bcm2835 C-ohjelmointikirjasto, joka mahdollistaa pääsyn Raspberry Pi:n GPIO-pinneihin.

```
const { composeAPI } = require('@iota/core');
const { asciiToTrytes, trytesToAscii } = require('@iota/converter')
const { createChannel, createMessage, parseMessage, mamAttach, mam-
Fetch } = require('@iota/mam.js');
const crypto = require('crypto');
const fs = require('fs');

const sensor = require('node-dht-sensor');

const SENSOR_TYPE = 11; //11=DHT11, for bcm2835 library
const GPIO = 4; //RasPi GPIO data pin
```

Kuva 6. Kirjastojen ja moduulien alustus.

Seuraavaksi tehdään run-funktio, joka vastaanottaa anturidatapaketin. Funktion alussa käsitellään MAM-kanavan ylläpito. Kanava ladataan erillisestä JSON-tiedostosta, tai luodaan uusi, jos sellaista ei ole vielä tehty. Kanavan luomiseen käytetään generate-Seed -funktiota, joka esitellään myöhemmin (Kuva 7).

```
async function run(payload) {

    //Setup the channel.
    const mode = 'restricted';
    const sideKey = 'MYKEY';
    let channelState;

    //Load the channel state from json file
    try {
        const currentState = fs.readFileSync('./channelState.json');
        if (currentState) {
            channelState = JSON.parse(currentState.toString());
        }
    } catch (e) { }

    //Create new channel if load failed
    if (!channelState) {
        channelState = createChannel(generateSeed(81), 2, mode, sideKey)
    }
```

Kuva 7. Run -funktion luonti jonka alussa kanavan ylläpito.

MAM -viesti luodaan käyttämällä funktion parametrina olevaa payload vakiota, joka sisältää anturidatan. createChannel ja createMessage -funktioiden luomat tiedot esitetään näytöllä, jonka jälkeen puretaan viesti trinäärijärjestelmän triteistä takaisin alkuperäiseksi viestiksi. Lopuksi tallennetaan kanavan tila seuraavaa suoritusta varten (Kuva 8).

```
    //Create a MAM message
    const mamMessage = createMessage(channelState, asciiToTrytes(JSON.stringify(pay
load)));

    //Display the details for the MAM message.
    console.log('Seed:', channelState.seed);
    console.log('Address:', mamMessage.address);
    console.log('Root:', mamMessage.root);
    console.log('NextRoot:', channelState.nextRoot);

    //Decode the message using the root and sideKey.
    //The decode is for demonstration purposes, there is no reason to decode at thi
s point.
    const decodedMessage = parseMessage(mamMessage.payload, mamMessage.root, sideKe
y);

    //Display the decoded data.
    console.log('Decoded NextRoot', decodedMessage.nextRoot);
    console.log('Decoded Message', decodedMessage.message);

    //Store channel state for the next execution
    try {
        fs.writeFileSync('./channelState.json', JSON.stringify(channelState, undefi
ned, "\t"));
    } catch (e) {
        console.error(e)
    }
```

Kuva 8. Viestin luominen

Ennen kuin paketti lähetetään Tangleen, kerrotaan ohjelmalle provider-kentässä, mihin IRI-palvelimeen otetaan yhteys. Kuvassa 9 näkyvä osoite on aiemmin luodun pilvipalvelimen julkinen IP-osoite ja porttina on IRI:n API-portti 14265. Paketti lähetetään Tangleen mamAttach -funktiolla, jonka mukana annetaan argumenttina osoite, viesti, maksimisyvyys, jolla kärjen valinta aloitetaan (koodissa arvo 3), sekä MWM-arvo (*minimum weight magnitude*), joka kertoo tritteinä, kuinka paljon laskennallista työtä tehdään. Lisäksi generoidaan linkki MAM subscriber -työkaluun, josta lähetystä voi tarkastella.

Lopuksi lähetys haetaan Tanglesta mamFetch-funktiolla osoitteen, viestin juuren, valitun lähetystavan (julkinen, yksityinen tai rajattu) ja salasanan avulla. Viestin sisältö muunnetaan triteistä ASCII-muotoon trytesToAscii-funktiolla ja tulostetaan näytölle.

```
    //Attach to Tangle

    // IRI Full Node to connect
    const api = composeAPI({ provider: "http://3.15.170.25:14265" });

    //Attach the message
    console.log('Attaching to tangle, please wait...')
    await mamAttach(api, mamMessage, 3, 14);
    console.log(`Link for the MAM subscriber: https://utils.iota.org/mam/${mamMessa
ge.root}/${mode}/${sideKey}/mainnet`);

    //Fetch from Tangle
    console.log('Fetching from tangle, please wait...');
    const fetched = await mamFetch(api, mamMessage.root, mode, sideKey)
    if (fetched) {
        console.log('Fetched', JSON.parse(trytesToAscii(fetched.message)));
    } else {
        console.log('Nothing was fetched from the MAM channel');
    }
}
```

Kuva 9. Tangleen lähetys ja haku

Luodaan generateSeed-funktio, joka generoi 81-merkkisen tunnisteen kanavalle. Funktiota käytetään kuvassa 6. Viimeiseksi luetaan anturista lämpötila ja kosteusprosentti. Lopuksi käynnistetään run-funktio payload-vakion kanssa (Kuva 10).

```
function generateSeed(length) {
    const charset = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ9';
    let seed = '';
    while (seed.length < length) {
        const byte = crypto.randomBytes(1)
        if (byte[0] < 243) {
            seed += charset.charAt(byte[0] % 27);
        }
    }
    return seed;
}

//Read temperature and humidity and store it to payload
//Start the process
let payload = {}

sensor.read(11, 4, async function(err, temperature, humidity) {
  if (!err) {
    payload.data = 'temp: ' + temperature.toFixed(1) + 'C, ' + 'humidity: ' + humid
ity.toFixed(1) + '%';
    payload.timestamp = (new Date()).toLocaleString();

   await run(payload)
        .then(() => console.log("done"))
        .catch((err) => console.error(err));

  }
});
```

Kuva 10. generateSeed -funktio, anturin hallinta, ja run -funktion käynnistys.

Ohjelma suoritetaan *node index.js*-komennolla, jonka jälkeen näytölle tulostetaan kanavan ja viestin tiedot. Kun Tangleen lähetys on aloitettu, ohjeistetaan käyttäjää odottamaan, sillä lähetys saattaa kestää muutaman minuutin. Kun lähetys on valmis, tulostetaan linkki MAM subscriber -työkaluun, josta lähetystä voi tarkastella. Lopuksi haetaan data Tanglesta ja tulostetaan näytölle (Kuva 11).

```
pi@raspberrypi: ~/mam
pi@raspberrypi:~/mam $ node index.js
Seed: 9XJZRIHZQDNLKD9OMYVEHCFFTJG9PZZ9AXRCTPMRFGJ9DSZAVNRACWTXKXTVZNGETTDOLYLKWIXMONXN9
Address: ZYKKUDBITC9TGYX9XYVGGSXJWYFL9PMOXXOIZY9TCVRXJAALRRJFCFYJBLAVUJCMBTCYCHDT9VAHTUQYI
Root: MMXGMKGOKZCEL9CIDYJGKYJGCAEXHLIQQHLLSJNNFAIPPPYFNSEEFTIYXFOXTJXVTKFIGQCBTVGLNRSWP
NextRoot: RWRISNCQYHAIGYDHPLCTCXNVLSPMMNVXOSOQZCKJQBLPV99RFCZJOXMWCGZLGV9LMKUCQWFFIOVHUBDZB
Decoded NextRoot RWRISNCQYHAIGYDHPLCTCXNVLSPMMNVXOSOQZCKJQBLPV99RFCZJOXMWCGZLGV9LMKUCQWFFIOVHUBDZB
Decoded Message ODGASCPCHDPCGADBGAHDTCADDDDBEAWAVASAUAMBQAEAWCIDADXCSCXCHDMDDBEAXABBSAUAJAGAQAGAHDXCADTCGDHDPCADDDGADBGAYATAWATAWAUAWAUAQAEACBDBUAVADBUAUAEAZBWBGAQD
Attaching to tangle, please wait...
Link for the MAM subscriber: https://utils.iota.org/mam/MMXGMKGOKZCEL9CIDYJGKYJGCAEXHLIQQHLLSJNNFAIPPPYFNSEEFTIYXFOXTJXVTKFIGQCBTVGLNRSWP/restricted/MYKEY/mainnet
Fetching from tangle, please wait...
Fetched {
  data: 'temp: 21.0C, humidity: 38.0%',
  timestamp: '4/2/2020, 9:01:00 PM'
}
done
pi@raspberrypi:~/mam $
```

Kuva 11. Node.js ohjelman suoritus

5 LOPUKSI

Opinnäytetyön tavoitteena oli tutustua IOTA:n Tangle-verkkoon ja rakentaa ympäristö, jossa anturidataa lähetetään Tangleen. Työn alussa tutkittiin, mikä on IOTA ja miten Tangle toimii.

Lopputuloksena toteutettiin tavoitteen mukainen ympäristö. Ympäristössä käytettiin Raspberry Pi 4 -tietokonetta, Node.js-ohjelmaa ja digitaalista DHT11-lämpötila- ja kosteusanturia. Node.js-ohjelmassa käytettiin IOTA:n mam.js-ohjelmointikirjastoa. Lisäksi konfiguroitiin erilliselle palvelimelle Tanglen IRI-noodi. Prototyypissä kerättiin lämpötila- ja kosteusdataa DHT11-anturilla, muunnettiin datapaketti lähetettävään muotoon ja lähetettiin paketti IRI:lle tarkastettavaksi, jonka jälkeen datapaketti on Tanglessa.

Prototyyppiä on mahdollista jatkaa esimerkiksi esittämällä data graafisesti www-sivulla tai mobiilisovelluksella. Lisäksi Node.js-ohjelmaa on mahdollista muokata siten, että dataa lähetetään jatkuvasti määritetyn intervallin välein.

Työssä käytetty lähdemateriaali on pääosin IOTA:n omilta internetsivuilta aiheen uutuudesta johtuen. IOTA:n taustalla olevat matemaattiset perusteet sekä aiheeseen liittyvät, IOTAa vanhemmat teknologiat voitiin pääosin käsitellä IOTA:n ulkopuolisella materiaalilla. Aihe oli mielenkiintoinen, mutta myös varsin laaja. Työn teoriaosa kattaa tärkeimmät aihealueet ja rajaa kokonaan ulkopuolelle IOTA kryptovaluutan.

LÄHTEET

1. Barber, D. Bayesian Reasoning and Machine Learning. Cambridge University Press, 2012. 690 s. Saatavissa: http://www.cs.ucl.ac.uk/staff/d.barber/brml/

2. Worldbank, Blockchain & Distributed Ledger Technology (DLT), 2018. Saatavissa: https://www.worldbank.org/en/topic/financialsector/brief/blockchain-dlt. Viitattu 6.4.2019

3. Handy, P. Introducing Masked Authenticated Messaging, 2017. Saatavissa: https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e. Viitattu 16.4.2019

4. IOTA Foundation, Node software, 2020a. Saatavissa: https://docs.iota.org/docs/node-software/0.1/introduction/overview. Viitattu 13.3.2020

5. IOTA Foundation, Meet the Tangle, 2018a. Saatavissa: https://www.iota.org/research/meet-the-tangle. Viitattu 5.4.2019

6. IOTA Foundation, Our Vision, 2020b. Saatavissa: https://www.iota.org/the-foundation/our-vision. Viitattu 20.4.2020

7. IOTA Foundation, Frequently Asked Questions: What is IOTA?, 2018b. Saatavissa: https://www.iota.org/get-started/faqs. Viitattu 12.4.2019

8. IOTA Foundation, The IOTA Foundation, 2020c. Saatavissa: https://www.iota.org/the-foundation/the-iota-foundation. Viitattu 21.4.2020

9. IOTA Foundation, Frequently Asked Questions: How is IOTA different from Blockchain?, 2018c. Saatavissa: https://www.iota.org/get-started/faqs. Viitattu 6.4.2019

10. IOTA Foundation, Frequently Asked Questions: What is needed to issue a transaction?, 2018d. Saatavissa: https://www.iota.org/get-started/faqs. Viitattu 15.4.2019

11. Popov, S. The Tangle, 2018. Saatavissa: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf

12. IOTA Foundation, Frequently Asked Questions: You mentioned a Markov Chain Monte Carlo Random Walk algorithm for tip selection, what is that? , 2020d. Saatavissa: https://www.iota.org/get-started/faqs. Viitattu 12.5.2019

13. What is AWS. Saatavissa: https://aws.amazon.com/what-is-aws/. Viitattu 16.4.2020

14. Wikipedia, Ubuntu, 2020. Saatavissa: https://en.wikipedia.org/wiki/Ubuntu. Viitattu 27.4.2020

15. Grafana. Saatavissa: https://grafana.com/grafana. Viitattu 10.4.2020

16. akashgoswami, IOTA Peer Manager. Saatavissa: https://github.com/akashgoswami/ipm. Viitattu 15.4.2020

17. DHT11, DHT22 and AM2302 Sensors. Saatavissa: https://www.adafruit.com/product/386. Viitattu 9.4.2020

18. McCauley, M. C library for Broadcom BCM 2835 as used in Raspberry Pi, 2020. Saatavissa: https://www.airspayce.com/mikem/bcm2835/index.html. Viitattu 9.4.2020

19. Opensource, What is a Raspberry Pi?, 2019. Saatavissa: https://opensource.com/resources/raspberry-pi. Viitattu 9.4.2020

20. OpenJS Foundation, About | Node.js, 2020. Saatavissa: https://nodejs.org/en/about/. Viitattu 14.4.2020