TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TONI HEIKKINEN

REAALIAIKAKÄYTTÖJÄRJESTELMÄN VALITSEMINEN

Diplomityö

Tarkastaja: professori Timo D. Hämäläinen

Tarkastaja ja aihe hyväksytty 30. toukokuuta 2018

TIIVISTELMÄ

TONI HEIKKINEN: Reaaliaikakäyttöjärjestelmän valitseminen
Tampereen teknillinen yliopisto
Diplomityö, 48 sivua
Joulukuu 2018
Sähkötekniikan DI-tutkinto-ohjelma
Pääaine: Sulautetut järjestelmät
Tarkastaja: professori Timo D. Hämäläinen

Avainsanat: reaaliaikakäyttöjärjestelmä, RTOS, valinta, vertailu

Asioiden internetin yleistyessä ja sulautettujen järjestelmien käydessä monimutkaisemmiksi niiden kehittäminen suoraan mikrokontrollerille vie yhä enemmän resursseja ja virheiden mahdollisuus kasvaa. Mikrokontrollereiden markkinat ovat myös erittäin hajanaiset ja jos sama ohjelma harvoin toimii muilla kuin juuri sillä mikrokontrollerilla kuin mille se on kirjoitettu.

Tähän ongelmaan vastaus löytyy usein mikrokontrollereille suunnatuista reaaliaikakäyttöjärjestelmistä. Käyttöjärjestelmiä on tehty laaja valikoima eri mikrokontrollereille ja usein tuki myös on päällekkäistä. Käyttöjärjestelmistä on tehty erittäin vähän vertailuja ominaisuuksien osalta. Tässä työssä on tarkoitus tutustua muutamaan suhteellisen yleiseen käyttöjärjestelmään ja tehdä hieman vertailua niiden ominaisuuksista ja esimerkkien avulla selventää kuinka käyttöjärjestelmän valintaa voi rajoittaa.

Työssä ei pyritä etsimään mitään yksittäistä käyttöjärjestelmää ja asettamaan näitä mitenkään parhausjärjestykseen ja luotetaan reaaliaikaominaisuuksien osalta kehittäjien dokumentaatiota järjestelmistä.

Työhön on valittu seitsemän reaaliaikakäyttöjärjestelmää, jotka ovat Mbed OS 5, FreeRTOS, ChibiOS, Zephyr, Mongoose OS, Contiki ja Phoenix RTOS. Käyttöjärjestelmistä on kerätty muun muassa taulukoita niiden kehittäjän lupaamasta mikrokontrolleritуesta, verkkoyhteyksistä, asiakaspalvelun ja tuen saatavuudesta ja yleisiä tietoja käyttöjärjestelmistä.

ABSTRACT

TONI HEIKKINEN: On Selection of Real Time Operating Systems
Tampere University of Technology
Master's Thesis, 48 pages
December 2018
Electrical Engineering
Major: Embedded Systems
Examiner: Professor Timo D. Hämäläinen

Keywords: real time, RTOS, selection criterion

As more embedded devices are connected to the Internet and the complexity of embedded software keeps growing it is taking more resources to develop programs directly for chosen hardware. Implementing standardized components from zero also creates more openings for errors and security flaws.

In this work, we look at some of the real-time operating systems created for microcontrollers. There exists plenty of real-time operating systems for microcontrollers but not really studies comparing these operating systems. The aim of this work is to familiarize the reader to seven quite popular operating systems and list some of the features. The goal is to show the features of these operating systems, not to create a competitive study to find the best operating system.

The operating systems chosen for this work are Mbed OS 5, FreeRTOS, ChibiOS, Zephyr, Mongoose OS, Contiki, and Phoenix RTOS. In this work, all the data for the operating systems are collected from their home sites and some of the papers published about these systems.

ALKUSANAT

Idea tähän diplomityöhön tuli siitä, että heti ensimmäisinä päivinä töissä joutui käyttämään reaaliaikakäyttöjärjestelmiä ja tekemään enemmän tai vähemmän yksinkertaisia ohjelmia niiden testaamiseksi. Koska koulun ainut aiheeseen liittyvä kurssi oli täysin teoreettinen niin itse käyttöjärjestelmät olivat täysin uusi tuttavuus.

Työn aikana reaaliaikakäyttöjärjestelmät kävivätkin yhä tutummiksi ja myös työhön käytetty aika on auttanut töissä kun on joutunut siirtymään käyttöjärjestelmästä toiseen.

Haluan kiittää tarkastajaa prof. Timo Hämäläistä varsinkin siitä, että tarvittaessa vastaukset ovat olleet aina nopeita ja palaute on aina auttanut viemään työtä eteenpäin suurin harppauksin.

Tampereella, 14.11.2018

Toni Heikkinen

SISÄLLYSLUETTELO

KUVALUETTELO

TAULUKKOLUETTELO

LYHENTEET JA MERKINNÄT

API	engl. *Application Programming Interface*, ohjelmointirajapinta
AWS	engl. *Amazon Web Services*, Amazonin pilvipalvelu
BSD	engl. *Berkeley Software Distribution*, käyttöjärjestelmänimitys
CPU	engl. *Central Processing Unit*, suoritin tai prosessori
CoAP	engl. *The Constrained Application Protocol*, kevyt tiedonsiirtoprotokolla IoT laitteille
DHCP	engl. *Dynamic Host Configuration Protocol*, protokolla TCP/IP asetuksien automatisointiin
GPIO	engl. *General Purpose Input/Output*, sisään- ja ulostulo pinni
GPLv3	Gnu General Public License v3
HAL	engl. *Hardware Abstraction Layer*, laitteistoabstraktiokerros
I2C	NXP Semiconductors:n kaksisuuntainen ohjaus- ja tiedonsiirtoväylä
IC	engl. *Integrated Circuit*, mikropiiri
IDE	engl. *Integrated Development Environment*, ohjelmointi- tai kehitysympäristö
IoT	engl. *Internet of Things*, verkkoon yhdistetty sulautettu laite
MCU	engl. *Microcontroller Unit*, mikrokontrolleri
MIT	engl. *Massachusetts Institute of Technology*, Massachusettsin teknillinen korkeakoulu
MQTT	Kevyt kommunikointiprotokolla
NFC	engl. *Near Field Communication*, lyhyen matkan langaton yhteys
OS	engl. *Operating System*, käyttöjärjestelmä
OTA	engl. *Over-The-Air programming*, etäohjelmointi
PoC	engl. *Proof of concept*, soveltuvuusselvitys
RAM	engl. *Random Access Memory*, hajasaantimuisti, tässä yhteydessä keskusmuisti
RMS	engl. *Rate-monotonic scheduling*
ROM	engl. *Read Only Memory*, lukumuisti, tässä tallennusmuisti
RTOS	engl. *Real Time Operating System*, reaaliaikakäyttöjärjestelmä
SLIP	engl. *IP over Serial Line*, Zephyr OS:ssä käytetty verkkoprotokolla
SPI	engl. *Small Computer System Interface*
TCP	engl. *Transmission Control Protocol*, internet tietoliikenneprotokolla
TTY	Tampereen teknillinen yliopisto
UDP	engl. *User Datagram Protocol*, tietoliikenneprotokolla
URL	engl. *Uniform Resource Locator*, verkkosivun osoite
USART	engl. *Universal Synchronous/Asynchronous Receiver-Transmitter*, tiedonsiirtoväylä

1. JOHDANTO

Koska nykyään yhä enemmän sulautettuja järjestelmiä yhdistetään verkkoon ja sitä kautta erilaisiin pilvipalveluihin [43] on tullut yhä suurempi tarve kehittää käyttöjärjestelmiä jotka vastaavat sekä reaaliaikatarpeeseen että verkkoyhteyden tuomaan haasteeseen. Laitteen kehityskustannukset ja -aika voivat karata käsistä, jos joka kerta kun alustaa vaihdetaan joudutaan koko verkkoliikenteestä vastaava pino (engl. network stack) tekemään uusiksi. Tähän ongelmaan pyritään vastaamaan käyttämällä reaaliaikakäyttöjärjestelmiä jotka vastaavat tarpeeksi nopeasti ja ennustettavasti reaaliaikatarpeisiin ja kuitenkin tarjoavat usein valmiin toteutuksen verkkoliikenteen hallintaan.

Reaaliaikakäyttöjärjestelmät (engl. real-time operating system, RTOS) ovat käyttöjärjestelmiä joiden tehtävähallinta (engl. task manager) on täysin ennustettavissa ja näin ollen ohjelmoija voi tarkasti määrittää kuinka paljon aikaa tehtävän tai tehtävien suorittamiseen menee pahimmassa tapauksessa (engl. worst case scenario). Perinteisessä käyttöjärjestelmässä, kuten Microsoft Windows, ohjelmoija ei voi tarkkaan määrittää kuinka paljon aikaa hänen ohjelmansa suorittamiseen huonoimmassa tapauksessa käytetään.

RTOS:n suurimpia etuja on ohjelman uudelleenkäyttö useammalla alustalla, näin koko ohjelmapohjaa (engl. code base) ei tarvitse kirjoittaa uudestaan jokaiselle käytetylle mikrokontrollerille (engl. micro controller unit, MCU) [12].

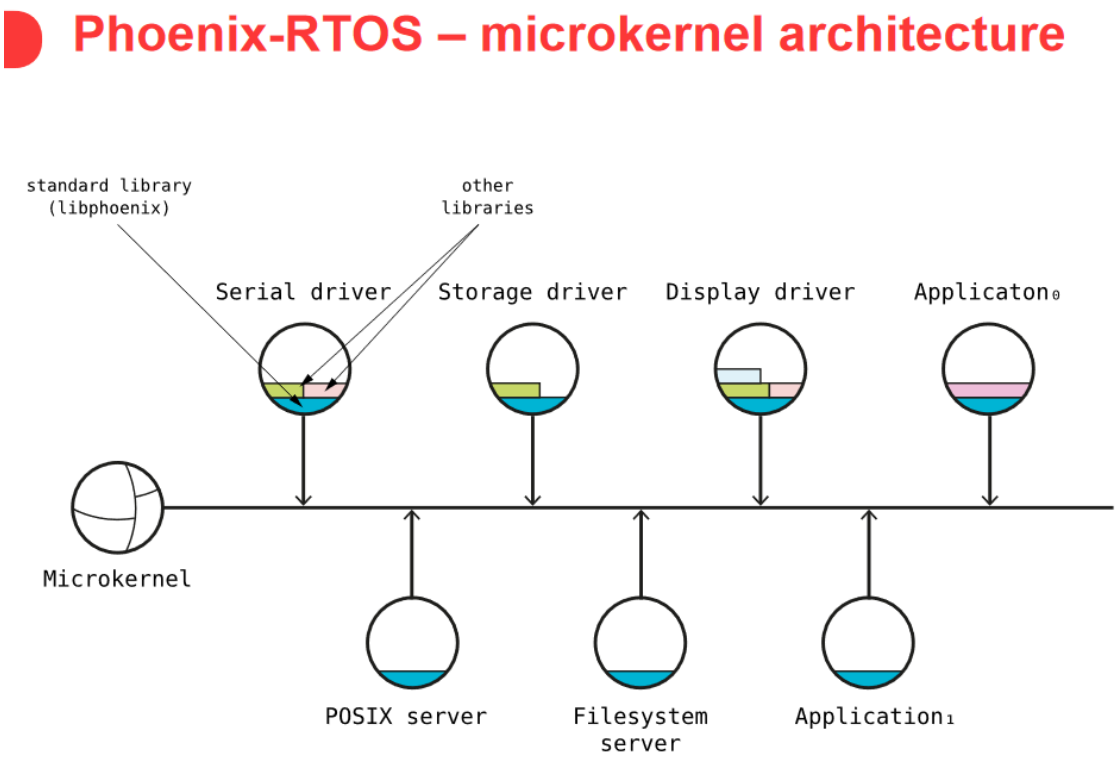
Käyttöjärjestelmät poikkeavat toisistaan alustan, ytimen ja laajennettavuuden mukaan. Reaaliaikakäyttöjärjestelmien muistivaatimukset ja muut ominaisuudet eroavat suuresti. Pienimmät vievät vain muutamia tavuja keskusmuistia (RAM) ja ne on suunnattu halvemmille pienille kontrollereille kuten Microchipin (ennen Atmel) AVR. Nämä minimalistiset käyttöjärjestelmät yleensä tarjoavat pelkästään säikeistyksen (engl. threads), säikeiden priorisoinnin, lukot (engl. mutex, lock) ja tarvittavat muistipinot (engl. memory stacks). Tästä esimerkkinä kaksoislisenssillä jaossa oleva Femto OS [60].

Tässä työssä ei käydä läpi reaaliaikakäyttöjärjestelmän toimintaa vaan keskitytään vertailemaan eri käyttöjärjestelmiä. Oletus on, että lukijalla on jo käsitys ja mahdollisesti kokemusta reaaliaikakäyttöjärjestelmien käyttökohteista ja käyttämisestä.

1.1 RTOS ytimet

Reaaliaikakäyttöjärjestelmiä on tarjolla erilaisilla ytimillä (engl. kernel). Esimerkiksi Phoenix-RTOS käyttää mikroydintä (engl. microkernel) [52]. Kuvassa 1 on esitetty Phoenix-RTOS mikroytimen arkkitehtuuri. Ytimen valinnalla on väliä, koska ytimen tyyppi vaikuttaa myös siihen kuinka kauan pahimmassa tapauksessa ohjelman suorittaminen kestää ja kuinka eri palvelut toteutetaan käyttöjärjestelmässä.

Mikroytimessä ydin tarjoaa vain järjestelmän ytimen kannalta tärkeimmät palvelut. Kaikki mikä on voitu siirtää ydintilasta (engl. kernel space) käyttäjätilaan (engl. user space) ilman ytimen toiminnan rikkoontumista on siirretty käyttäjätilaan. Tämä pitää sisällään esimerkiksi verkkoliikenteestä ja tiedostojärjestelmistä vastaavat palvelut [40].



Kuva 1. *Phoenix RTOS mikroytimen arkkitehtuuri, lähde: [52]*

Monoliittisessa ytimessä palvelut, kuten tiedostojärjestelmän ylläpito, suoritetaan ydintilassa. Tämä nopeuttaa eri osien välistä liikennettä, mutta monoliittisessa järjestelmässä yhden palvelun kaatuminen voi kaataa koko järjestelmän. Mikroytimessä esimerkiksi tiedostojärjestelmästä vastaavan palvelun kaatuminen on verrannollinen normaalin ohjelman kaatumiseen ja muut käyttöjärjestelmän toiminnot, jotka eivät ole riippuvaisia tiedostojärjestelmästä, voivat jatkaa toimintaansa.

Kuvassa 2 on vertailun vuoksi esitetty ChibiOS:n käyttöjärjestelmäarkkitehtuuri. ChibiOS on selkeästi kerrostettu eri osiin, jotka keskustelevat toisilleen rajapintoja pitkin. Tämä eroaa huomattavasti puhtaasta mikroydinarkkitehtuurista, jossa eri osat keskustelevat toisilleen ja ytimelle käyttäen samaa postitusmekaniikkaa [40].

1.2 Tutkimuksen menetelmä

Työ on suoritettu vertailututkimuksena ja tarkoitus on kerätä eri käyttöjärjestelmätoimittajilta ja tutkimuksista tarvittavat tiedot vertailun tekemiseen. Käyttöjärjestelmien kotisivuilla

suoraan esitellään yleisesti ne ominaisuudet ja tiedot jotka käyttöjärjestelmän toimittajan mukaan ovat tärkeimmät ja näyttävät käyttöjärjestelmän parhaimmat puolet.

1.3 Tavoitteet

Työn tavoitteena on esitellä hieman eri reaaliaikakäyttöjärjestelmien ominaisuuksia ja kriteerejä mitkä voivat vaikuttaa käyttöjärjestelmän valinnassa. Tarkoituksena on myös esittää esimerkkien avulla hieman sitä prosessia millä käyttöjärjestelmän valitsemista voi rajoittaa.

Tarkoituksena ei siis ole etsiä mitään yksittäistä parasta käyttöjärjestelmää koska käyttöjärjestelmät eroavat suuresti ominaisuuksiltaan ja tueltaan eri mikrokontrollereille.

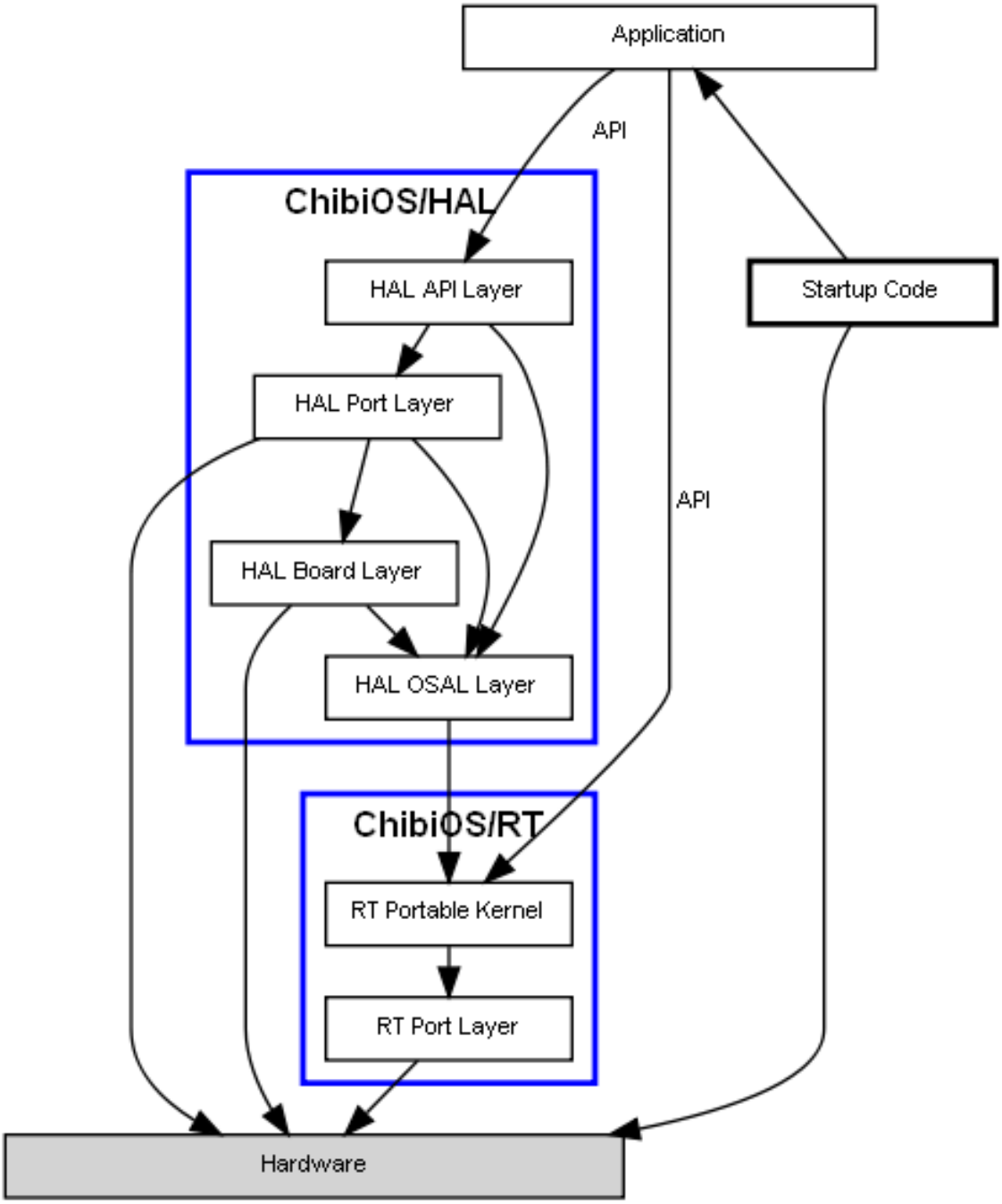
1.4 Työn rakenne

Luvussa 2 esitetään aiheeseen liittyviä töitä. Vaikka reaaliaikakäyttöjärjestelmiä tutkitaan paljon niin kattavia vertailuja niiden ominaisuuksista on tehty vähän tai suppea vertailu on osana muuta työtä.

Luvussa 3 esitellään vertailukriteerejä mitä tässä työssä asetetaan reaaliaikakäyttöjärjestelmien vertailulle. Kriteerit eivät ole järjestetty mitenkään tärkeysjärjestykseen vaan vaikuttavat käyttöjärjestelmän valintaan riippuen sovelluksesta. Esimerkiksi sulautettu järjestelmä mitä ei yhdistetä verkkoon ei välttämättä vaadi lainkaan tietoturvan suunnittelua.

4. luvussa esitellään tähän työhön valittujen käyttöjärjestelmien perustietoja. Käyttöjärjestelmät on pääasiassa valittu käyttöjärjestelmän lisenssin perusteella.

Luvussa 5 valittujen käyttöjärjestelmien ominaisuudet taulukoidaan ja käydään läpi, jonka jälkeen luvussa 6 käydään läpi esimerkkien avulla mitkä asiat vaikuttavat käyttöjärjestelmän valitsemiseen.



Kuva 2. *ChibiOS järjestelmän arkkitehtuuri, lähde: [19]*

2. LIITTYVÄT TYÖT

Reaaliaikakäyttöjärjestelmistä ja niiden käyttämisestä on kattavasti tutkimuksia, lehtiartikkeleita, kirjoja ja lopputöitä, mutta vähän kattavaa vertailua. Vertailua hankaloittaa jo sekin, että käyttöjärjestelmät kehittyvät nopeasti ja vertailu voi osittain olla vanhentunut jo julkaisuhetkellä. Tässä työssä on tarkoitus kartoittaa pääosiltaan mikä käyttöjärjestelmä olisi mahdollisesti sopiva valitulle mikrokontrollerille ja laitetoteutukselle yleisellä tasolla.

Tampereen Teknillisen Yliopiston diplomitöistä löytyi kaksi, joissa on aihetta sivuttu nopeasti muun työn yhteydessä. Juha Onkilan työssä esitellään 3 käyttöjärjestelmää pintapuolisesti ja tämän jälkeen keskitytään pelkästään työssä käytettävän käyttöjärjestelmän ominaisuuksiin [50]. Ville Juven käy tarkemmin läpi reaaliaikakäyttöjärjestelmälle asetettuja vaatimuksia ja tutkii valmiin käyttöjärjestelmän käyttöä langattomassa sensoriverkossa [39]. Tässäkin työssä itse käyttöjärjestelmien vertailu on lyhyehkö ja järjestelmien ominaisuuksiin ei mennä kovinkaan syvälle.

2.1 Aiemmin tehtyjä vertailuja

Tran Nguyen Bao Anh et al. on julkaissut vuonna 2009 vertailua reaaliaikakäyttöjärjestelmistä [4]. Vertailussa on muun muassa mainittu, että vain muutamille järjestelmille on käyttöjärjestelmän huomaava (engl. operating-system-awareness) tuki ohjelmointiympäristöissä. Kuitenkin nykyään esimerkiksi IAR tukee jo yli kymmentä käyttöjärjestelmää [35].

Mir Ashfague Ali et al. tutkii julkaisussaan kaupallisten käyttöjärjestelmien vuorontimia (engl. scheduler) [1]. Työssä esitellään suppeasti viidentoista käyttöjärjestelmän ominaisuuksista käytetty vuoronnus, säikeiden prioriteettitasojen määrä (engl. task priority levels), säikeiden synkronointimekanismit (engl. task synchronization mechanisms) ja "priority inversion protection mechanism". Käyttöjärjestelmät on valittu kahden pääkriteerin avulla. Käyttöjärjestelmän täytyy täyttää RTOS reaaliaikavaatimukset ja sen täytyy olla POSIX 1003.1b yhteensopiva.

Tej Bahadur Chandra et al. tutkielma vuodelta 2016 vertailee IoT-käyttöjärjestelmiä [17] ja sisältää tässäkin työssä esiintyvät Contikin, FreeRTOS:n, mbed:n ja ChibiOS:n osana vertailua. Työssä ei kuitenkaan kerrota mikä pääversio (engl. major version) käyttöjärjestelmästä on kyseisellä hetkellä ollut käytössä. Tämä tieto kertoisi suoraan, kuinka monta suurempaa versiopäivitystä (engl. major version updates) käyttöjärjestelmät ovat saaneet tänä aikana.

Näistä vertailuista voidaan ottaa pohjaa siihen, kuinka tietoa voidaan esittää luvussa 5. Koska tutkimukset ovat jo sisällöltään muutoin suurimmilta osin vanhentuneet niin näiden

sisältämää tietoa ei käyttöjärjestelmien kohdalla voi käyttää. Esimerkiksi FreeRTOS:sta on elokuussa 2018 julkaisu versio v10.1.0 kun Tej Bahadur Chandra et al. vertailussa pääversio on ollut joko v8.2.3 tai v9.0.0 [31].

3. VERTAILUKRITEERIT

RTOS:n valintaan vaikuttavat useat eri kriteerit. Kaikissa järjestelmissä kuitenkin tärkeimpänä on vuoronnuksen ennustettavuus (engl. preemptive scheduling). Säikeiden suorituksen takaraja (engl. deadline) täytyy olla määritettävissä, oli se sitten "rate-monotonic scheduling"(RMS) tai prioriteetti pohjainen vuoronnus (engl. priority based scheduling).

Nykyään myös entistä tärkeämmäksi osaksi on tullut asioiden internet (engl. Internet of Things, IoT) ja kaikki tässä vertailussa otetut käyttöjärjestelmät mainitsevatkin ominaisuudekseen IoT-kehittämisen helppouden.

Tässä luvussa esitellään ominaisuudet jotka vaikuttavat käyttöjärjestelmän valinnassa. Näitä ovat esimerkiksi lisenssit, tietoturva, kustannukset, päivitettävyys, tuki, ohjelmointikieli, mille mikrokontrollereille järjestelmä on saatavilla ja kuinka paljon järjestelmä tarvitsee muistia ja tallennustilaa. Työssä ei ole tarkoitus etsiä yksittäistä parasta järjestelmää vaan tutkia kuinka ne vertautuvat toisiinsa eri tilanteissa ja mitkä asiat vaikuttavat järjestelmän valitsemiseen.

3.1 Lisenssit

Lisenssit millä järjestelmä ja sen osat on julkaistu, täytyy ottaa huomioon järjestelmää valittaessa. Avoimen lähdekoodin järjestelmät on julkaistu joko Apache- [5] tai MIT-lisenssillä [42] joka sallii ohjelmiston käyttämisen kaupallisiin tarkoituksiin, kunhan ohjelmiston alkuperä on mainittuna ja lisenssi on myös lähdekoodin mukana.

Eclipse Public License ja Eclipse Distribution License ovat myös avoimen lähdekoodin lisenssejä jotka antavat luvan käyttää, muokata, uudelleen julkaista ja myydä ohjelmaa [27].

Muita käytössä olevia lisenssejä on GNU General Public License v.2 ja v.3 (GPLv2, GPLv3) jotka sallivat ohjelmiston vapaan levittämisen, muokkaamisen ja uudelleen julkaisun myös kaupallisiin tarkoituksiin ehdolla, että lisenssi säilytetään ja lähdekoodi julkaistaan vapaaseen levitykseen [29].

Tietenkin vaihtoehtona on myös suljettu lisenssi missä kehittäjän täytyy ostaa lupa ohjelmiston käyttämiseen. Näissä tapauksissa itse käyttöjärjestelmän lähdekoodi ei aina ole saatavilla vaan ohjelmoijalle annetaan vain dokumentaatio järjestelmän käyttämiseen ja käännetty ohjelma ja käyttöjärjestelmäbinääri yhdistetään ennen laitteelle lataamista.

Tässä työssä keskitytään vapaan lähdekoodin käyttöjärjestelmiin. Jotkin näistä järjestelmistä kuitenkin voivat tarjota ilmaiseksi vain osan käyttöjärjestelmän ominaisuuksista ja

kaupallinen versio avaa kehittäjälle enemmän ominaisuuksia ja paremman asiakastuen. Esimerkkinä tämänlaisesta toteutuksesta on Mongoose OS, joka tarjoaa käyttöjärjestelmän minimaalisella toiminnallisuudella ilmaiseksi ja kaupallisella lisenssillä pääsyn täyteen järjestelmään [45].

3.2 Tietoturva

Tietoturvan merkitys kasvaa yhä suuremmaksi osaksi suunnittelua laitteiden verkkoyhteyksien yleistyessä. Tietoturva on kuitenkin tällä hetkellä vieläkin takasijalla sulautetuissa järjestelmissä [25, 34]. Esimerkkejä löytää muun muassa isobritannialaisen Pen Test Partnersin blogista [51] joka sisältää suuren määrän IoT-laitteisiin liittyvää testaamista.

Kryptografiaa käytettäessä sulautetuissa järjestelmissä on otettava huomioon laitteiden rajalliset resurssit. Esimerkiksi laite voi toimia paristoilla jolloin raskaat laskutoimitukset vaikuttavat suoraan laitteen käyttöaikaan [38].

Osa käyttöjärjestelmistä tarjoaa sisäänrakennettuna tuen salatulle liikenteelle ja kaikkiin käyttöjärjestelmiin on mahdollista lisätä salaus erillisen kirjaston avulla, esimerkiksi mbed TLS [9].

Nykyään on tosin mikrokontrollereita, joissa on erillinen laitetoteutus kryptografialle. Tässä tapauksessa salaus tehdään erillisellä tähän tarkoitetulla toteutuksella. Tämä nopeuttaa salaamista, salauksen purkamista ja autentikointia. Tämä myös pienentää itse ohjelmaa koska salaukselle ei tarvitse enää tehdä omaa kirjastoa tai ohjelmaa. [32, 44]

3.3 Firmware päivitykset

Asioiden internetin myötä tarve pystyä päivittämään tuote ilman uudelleen ohjelmointia on kasvanut koska laitteet voivat olla asennettuna useisiin eri kohteisiin jopa vuosiksi ilman, että kukaan käy paikan päällä katsomassa laitetta. Etäohjelmointiin (engl. Over-The-Air programming, OTA) on valmiita kaupallisia ratkaisuja kuten Arm:n tarjoama "Pelion Device Management" [10] tai Amazonin tarjoama AWS IoT [2].

Valmiiden palveluiden käyttö on kehittäjän kannalta helppo valinta koska sekä Amazon että Arm tarjoavat esimerkkiohjelmat heidän pilvipalveluidensa käyttöön. Nämä palvelut myös hoitavat liikenteen salattuna ja hoitavat tarvittavat tarkistukset siihen, että uusi laitteelle lähetetty firmware on varmasti oikeasta lähteestä eikä esimerkiksi kolmannen osapuolen tekemä haittaohjelma.

Etäohjelmoinnin pitää myös pystyä tarkistamaan, että uusi firmware myös lähtee käyntiin ja tarvittaessa ladattava takaisin vanha toimivaksi todettu firmware, jos uuden asentamisessa on tapahtunut virhe tai itse ohjelmassa on ohjelmistovirhe (engl. bug) joka kaataa ohjelman suorituksen. Tämä on erittäin tärkeä ominaisuus varsinkin laitteissa jotka ovat käytännössä lähes mahdotonta saada uudelleen ohjelmoitavaksi, esimerkkinä satelliitit.

3.4 Asiakaspalvelu ja tuki

Asiakaspalvelun ja tuen määrä vaihtelee suuresti järjestelmästä riippuen. Täysin vapaan lähdekoodin järjestelmissä ei välttämättä ole maksullista tukea lainkaan vaan kaikki tieto täytyy etsiä joko kyseisen järjestelmän sivuilta, lähdekoodista tai keskustelupalstoilta kuten stackoverflow.com.

Maksullisissa järjestelmissä lisenssiin kuuluu yleensä tekninen tuki kuten Mongoose OS:n tapauksessa, jolloin ohjelmistokehittäjä voi kysyä suoraan apua ongelmaansa käyttöjärjestelmän kehittäjältä. Kuitenkin näissäkin on yleensä ensin nopeampaa tarkistaa keskustelualueet siltä varalta, että joku muukin on miettinyt samankaltaiseen ongelmaan ratkaisua.

Luvussa 5 listataan tarkemmin mitä tukipalveluita käyttöjärjestelmille on tarjolla.

3.5 Kustannukset

Järjestelmän valintaan vaikuttaa välittömät ja välilliset kustannukset. Vaatiiko järjestelmän käyttö kalliita lisenssejä vai perustuuko järjestelmä avoimeen lähdekoodiin jolloin kuka tahansa voi käyttää ja osallistua järjestelmän kehittämiseen?

Välittömät kustannukset liittyvät järjestelmässä sen lisensointi- ja käyttökustannuksiin. Välilliset taas sulautetun järjestelmän suunnitteluun, komponenttien kustannuksiin ja siihen kuinka paljon resursseja tarvitaan itse ohjelmistokehitykseen.

3.6 Ohjelmointikieli

Yleinen ennakkoluulo on, että ohjelmoidakseen sulautetuille järjestelmille täytyy osata C-kieltä. Tämä on kuitenkin nykyään väistymään päin, koska ainakin C++ on yhä yleisemmässä käytössä [20, 33]. Joillakin järjestelmillä voi prototyypin tehdä muillakin kielillä esimerkiksi Mongoose OS:llä JavaScriptillä [45].

3.7 Mikrokontrollerit

Koska valinnanvaraa on todella paljon niin kontrollerin valinta on yksittäisesti suurin rajoittava tekijä käyttöjärjestelmän valinnassa. Esimerkiksi Mbed OS ei ole tarjolla kuin pelkästään Arm:n prosessoria käyttäville mikrokontrollereille. Tämä tietenkin toimii toisinkin päin, jos tiedetään mitä ominaisuuksia järjestelmältä vaaditaan niin mikrokontrolleri voidaan valita vastaamaan järjestelmän vaatimuksia.

Tässä työssä käyttöjärjestelmät on pääasiassa valittu siten, että niillä on tuki tehokkaammille mikrokontrollereille kuten STMicroelectronics STM32-sarjalaisille tai Espressif EPC32-kontrollereille, mutta ei silti ole rajoitettu pelkästään näihin.

3.8 Tuki oheislaitteille

Oheislaitetuki tässä työssä tarkoittaa valmista tukea mikrokontrollerin eri ominaisuuksille kuten USART (engl. Universal Synchronous/Asynchronous Receiver-Transmitter), SPI (engl. Small Computer System Interface), I2C ja tuki digitaalisille ja analogisille sisään- ja ulostulopinneille. Digitaalisilla pinneillä voi olla vain kaksi tilaa ja analogiset taas muuttavat jännitteen digitaaliseksi arvoksi useamman bitin tarkkuudella tai muuttavat digitaalisen arvon ulostulojännitteeksi.

Oheislaitteita ovat esimerkiksi erilliset verkkoliikennepiirit, GPS-piirit (engl. Global Satellite Positioning), näyttölaitteet, valodiodit ja toiset mikrokontrollerit.

Kaikki tämän työn käyttöjärjestelmät ovat mikrokontrollereilla toimivia ja sisältävät tuen yllämainituille väylille ja GPIO-pinneille. Tämän työn rajoissa kiinnostaa kuinka laaja tuki käyttöjärjestelmillä on verkkolaitteille.

3.8.1 Verkkoyhteydet

IoT-laitteilla tärkeänä ominaisuutena on laaja tuki eri verkkolaitteille ja -protokollille. Useimmat käyttöjärjestelmät tukevat yleisimmät yhteysmuodot eli langattoman verkkoyhteyden WiFi:n yli ja langallisen Ethernet-yhteyden, mutta lisäksi on IoT-laitteille suunnattuja yhteysvaihtoehtoja kuten 6LoWPAN (engl. IPv6 over Low-Power Wireless Personal Area Networks) [56, 49] ja Narrowband IoT (NB-IoT) [18].

3.9 Järjestelmän koko- ja muistirajoitteet

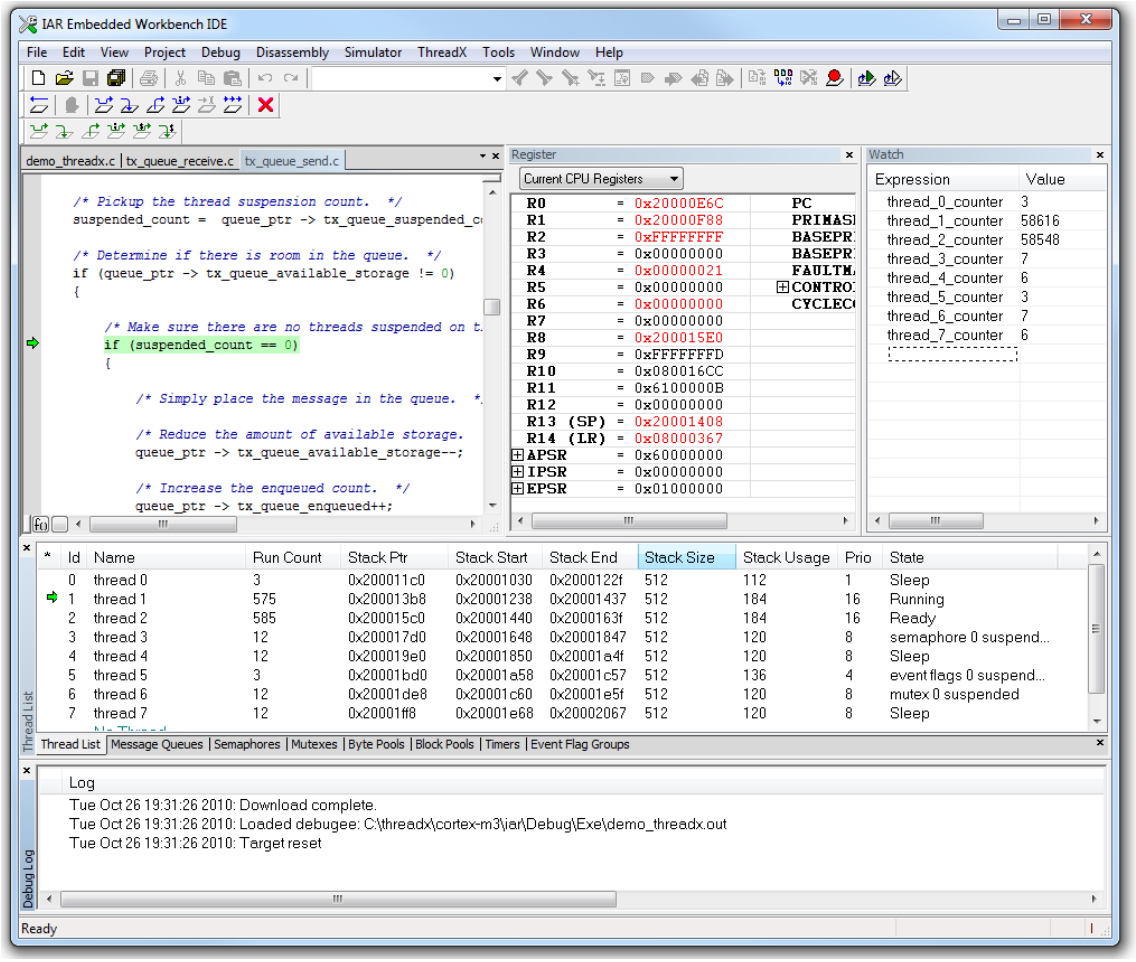
Järjestelmän vaatima muisti (engl. memory footprint) ja tallennustila on myös rajoittava tekijä. 8-bittisillä kontrollereilla on usein muistia vain muutamia kilotavuja ja vapaata tallennustilaa muutamia kymmeniä kilotavuja. Näille kontrollereille ei yleensä joko ole tai ei käytetä käyttöjärjestelmää vaan ohjelma kirjoitetaan suoraan laitteelle räätälöidyksi.

Reaaliaikakäyttöjärjestelmien kokovaatimukset vaihtelevat suuresti aina muutamista kilotavuista satoihin kilotavuihin riippuen myös siitä mitä kirjastoja ja ominaisuuksia tarvitaan.

3.10 Kehitysympäristö ja -työkalut

Sulautettujen järjestelmien kehitysympäristöjä (engl. Integrated development environment, IDE) on saatavilla lähes jokaiselta mikrokontrollerivalmistajalta oma, yleensä ilmainen, näistä esimerkkinä NXP:n julkaisema MCUxpresso [47] ja Texas Instrumentsin Code Composer Studio [59]. Molemmat pohjautuvat Eclipse ilmaiseen IDE:n.

Saatavilla on myös maksullisia kehitysympäristöjä ja näistä esimerkkinä IAR Embedded Workbench [36] jonka mukana tulee myös IAR:n oma kääntäjä. Nämä maksulliset järjestelmät tarjoavat pääasiallisena tuotteena vianetsintätyökaluja (engl. debugging tools)



Kuva 3. *Säikeet ThreadX-ytimelle IAR Embedded Workbench vianetsinnässä, lähde: [37]*

joissa on enemmän ominaisuuksia kuin monissa ilmaisissa versioissa, esimerkkinä IAR:n tarjoama käyttöjärjestelmätietoinen vianetsintä (engl. OS-aware debugging) joka pystyy näyttämään käyttöjärjestelmän sisäisiä tietoja kuten säikeiden määrän ja suoritustilanteen, muistipinot, käytössä olevat semaforit ja mutexit ja niiden tilat. Säikeiden näyttämisestä esimerkkinä kuva 3.

Muita ilmaisia kehitysympäristöjä on esimerkiksi Eclipse ja Visual Studio Code. Näiden kanssa vianetsintätilassa käytetään yleensä kolmansien osapuolien tarjoamia ohjelmia kuten GDB:tä (The GNU Project Debugger) yhdistettynä OpenOCD:hen (Open On-Chip Debugger) tai pyOCD:hen (Python On-Chip Debugger). Kuvassa 4 näytetään ohjelmapino tässä tapauksessa. Käytetty IDE käynnistää GDB-istunnon joka ottaa taas käytetyn OCD-palvelun kautta yhteyden mikrokontrollerille.

Kehitysympäristön käyttäminen ei ole kuitenkaan pakollista vaan ohjelman kirjoittamiseen voi käyttää mitä tahansa tekstieditoria, mutta kääntäjät ovat kuitenkin riippuvaisia käytetystä mikrokontrollerista.

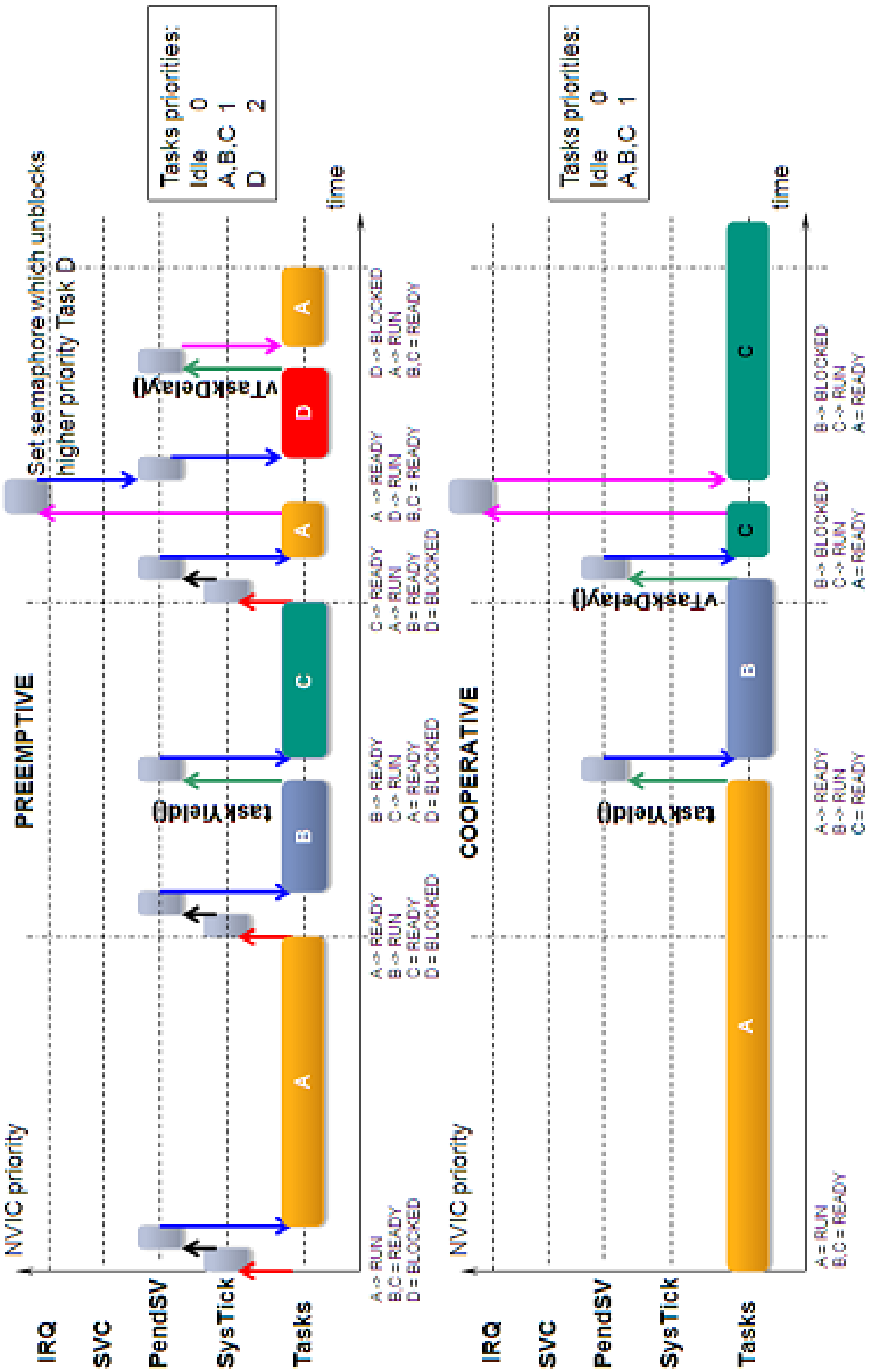
Kuva 4. *IDE yhdistettynä GDB:n ja OCD:n kautta mikrokontrolleriin*

3.11 Reaaliaikaominaisuudet

Reaaliaikakäyttöjärjestelmän reaaliaikaominaisuuksiin kuuluu tuki prosesseille ja säikeille sekä niiden vuoronnukselle. Käyttöjärjestelmä voi olla yksinkertainen tapahtumapohjainen (engl. event based) tai monisäikeinen jolloin tarvitaan vuoronnusta. Vuoronnus voi olla ennustava (engl. preemptive), yhteistyö (engl. cooperative), round-robin tai näiden yhdistelmä. Round-robin vuoronnuksessa prosesseille jaetaan suoritusaika sykleittäin ja ajoituksessa yhden syklin maksimiaika on helppo määrittää. Round-robin yleensä yhdistetään prioriteettipohjaiseen ennustavaan vuoronnukseen jossa samalla prioriteetilla suoritettavat säikeet ja vuoronnetaan round-robinia käyttäen.

Kuvassa 5 on esitetty, kuinka ennustava- ja yhteistyövuoronnus eroavat toisistaan. Ennustavassa vuoronnuksessa säikeillä on prioriteetti ja korkeamman prioriteetin säikeet menevät ensin suoritukseen ja voivat keskeyttää matalampien prioriteettien säikeiden suorituksen. Yhteistyöpohjaisessa vuoronnuksessa kaikilla säikeillä on sama prioriteetti ja säikeet vapauttavat suorituksen erillisellä komennolla tai suoritettuaan ohjelman loppuun.

Keskeytyksien hallinta on myös tärkeä osa monisäikeistä järjestelmää ja käyttöjärjestelmän keskeytysjärjestelmän nopeus on tässä huomattavassa roolissa. Keskeytystilanteessa usein halutaan, että keskeytys menee suoritukseen seuraavana tai välittömästi. Keskeytyksen saapumisen ja suorituksen aloittamisen välinen aika on useimmissa järjestelmissä se mikä voi rajoittaa järjestelmän käyttöä sovelluksessa, esimerkiksi auton sähköinen ohjaus jossa ajaja painaa jarrua on mentävä suoritukseen välittömästi keskeyttäen kaikki muut prosessit.



***Kuva 5**. Ennustavan ja yhteistyövuoronnuksen erot [54]*

4. VERTAILUUN VALITUT JÄRJESTELMÄT

Valitut järjestelmät ovat kaikki avoimen lähdekoodin järjestelmiä. Tämä päätös on tehty sen takia, että järjestelmien tarjonta on todella laaja ja näistä järjestelmistä on helppoa saada tarvittavat tiedot. Avoimen lähdekoodin järjestelmässä myös järjestelmätuen saatavuus on yleisesti pienempi ongelma koska tarvittavat muutokset voi myös tehdä itse.

Seuraavassa esitellään vertailtavat käyttöjärjestelmät pintapuolisesti ja luvussa 5 käydään tarkemmin läpi käyttöjärjestelmien ominaisuuksia ja eroja.

4.1 Arm Mbed OS 5

Arm Mbed OS on nimensä mukaisesti Arm:n prosessoria käyttäville mikrokontrollereilla suunnattu käyttöjärjestelmä. Mbedin virallisilta sivuilta löytyvä lista tuetuista laitteista sisältää mikrokontrollereita aina halvimmista Cortex-M0 sarjasta hieman järeämpään Cortex-M4 sarjaan. Pienimmät Cortex-M0 kontrollerit sisältävät 32 kilotavun tallennustilan ja 8 kilotavun keskusmuistin ja järeimmissä Cortex-M4 kontrollereissa on 2 megatavua tallennustilaa ja 384 kilotavua keskusmuistia.

Mbed OS sisältää valmiit toteutukset seuraaville teknologioille: Bluetooth LE, Wi-Fi, 6LoWPAN Sub-GHZ Mesh, NFC, Thread, LoRA LPWAN, RFID, Ethernet ja mobiiliverkkoyhteydet. Näille teknologioille löytyy valmiit kirjastot, kehitysalustat, tutoriaalit ja esimerkit. Sisäänrakennettuna on myös tuki digitaalisille ja analogisille IO-pinneille (engl. General Purpose Input/Output, GPIO), keskeytyksille, portti ja väylä IO:lle, pulssinleveysmodulaatio (engl, Pulse-Width Modulation, PWM) ja I2C-, SPI- ja sarjaporttiväylille.

Mbed OS:n kehittämiselle on myös selaimessa toimiva versio, jolla on helppo tehdä yksinkertainen testiohjelma käytetylle kehitysalustalle ja Mbed CLI (engl. Command-line interface, komentorivityökalu) ajettavaksi tietokoneella. Mbed CLI:n avulla projektille on helppo lisätä kirjastoja joko Mbedin omasta kirjastosta tai mistä tahansa Git-repositoriosta.

Mbed OS käyttää ytimenään Arm Keil CMSIS-RTOS RTX ydintä [6, 7]. Ydin on rojaltivapaa ja avoimella lähdekoodilla. Lähdekoodin saa kaikkien Keil MDK versioiden mukana, myös ilmaisen. Ydin käyttää vähimmillään noin 5 kilotavua tallennustilaa [6]. Ominaisuuksiin kuuluu säikeistys, lukot, semaforit (engl. semaphore) ja deterministinen vuoronnin. [7]

4.2 FreeRTOS

FreeRTOS:n ensimmäinen versio on julkaistu vuonna 2003 Richard Barryn toimesta ja myöhemmin kehitystä on hoitanut Real Time Engineers Ltd. 2017 FreeRTOS:n kehittäminen siirtyi Amazon Web Service:lle (AWS).

FreeRTOS käyttöjärjestelmän ytimenä toimii avoimeen lähdekoodiin perustuva FreeRTOS-ydin, joka on suunniteltu käytettäväksi mikrokontrollereiden kanssa. Käyttöjärjestelmän tilavaatimus on kuudesta kahteentoista kilotavua, joten näiltä osin se soveltuu hyvin pienemmillekkin mikrokontrollereille.

Käyttöjärjestelmän ominaisuuksiin kuuluvat deterministinen optimoitava vuoronnin, lukot, semaforit, viestijonot (engl. queue), tehokkaat ohjelmalliset ajastimet ja skaalautuvuus. Kotisivuilla mainitaan FreeRTOS:n viestijonojen olevan sekä yksinkertaisia että joustavia pienessä tilassa ja toimivat pohjana kaikelle kommunikaatiolle ja synkronisoinnille säikeiden ja taskien välillä. [31]

4.2.1 Amazon FreeRTOS

Amazon FreeRTOS (a:FreeRTOS) on Amazonin AWS:lle räätälöimä FreeRTOS versio jolle Amazon tarjoaa valmiita ohjelmistokomponentteja käytettäväksi Amazon Web Servicesin (AWS) kanssa. Nämä sisältävät muun muassa komponentteja verkkoliikenteelle Ethernetin, langattoman lähiverkon tai mobiiliverkon yli. Amazonin tarjoamissa esimerkeissä on muun muassa AWS IoT liikenne käyttäen MQTT:tä ja OTA -päivitykset. [3]

4.3 ChibiOS

ChibiOS on Giovanni Di Sirion kehittämä ja ylläpitämä reaaliaikakäyttöjärjestelmä jonka tavoitteena on olla suorituskykyisin käyttöjärjestelmä kaikista saatavilla olevista. Muita ominaisuuksia on täysin staattinen arkkitehtuuri, ennustettava ja deterministinen vuoronnus, pieni koko, täysi HAL tuki (engl. Hardware Abstraction Layer, laitteistoabstraktiokerros) ja MISRA 2012 yhteensopivuus [57].

ChibiOS on ensimmäinen esiteltävistä joka sisältää useampia lisenssitasoja. Täysi ohjelmisto on ilmaiseksi käytössä, jos myös julkaistu ohjelma julkaistaan avoimena GPL3 lisenssin alla. Apache 2.0 lisenssillä on saatavilla rajoitettu versio kaupalliseen käyttöön 500 laitteeseen saakka ja koko järjestelmän lähdekoodeineen saa käyttöönsä maksullisella lisenssillä jossa on kaksi tasoa, 1000 - 5000 laitteelle tarkoitettu kehityslisenssi ja rajoittamaton tuotantolisenssi.

ChibiOS tarjoaa kaksi eri versiota käyttöjärjestelmäytimestä, ChibiOS RT ja ChibiOS NIL. Nämä ytimet jakavat saman API:n (engl. Application Programming Interface, ohjelmointirajapinta), mutta eroavat sisäiseltä toiminnaltaan. NIL on tarkoitettu erittäin pienille

mikrokontrollereille ja vie maksimissaan yhden kilotavun tallennustilaa. Kuitenkin NIL:n tavoitteena on olla täysi RTOS eikä pelkästään minimalistinen toteutus vuorontimelle.

NIL:n ytimestä on karsittu joitakin ChibiOS RT:ssä olevia ominaisuuksia kuten tuki virtuaalisille ajastimille, säikeiden väliselle synkroniselle viestitykselle, ajonaikainen statistiikka, trace bufferit ja tuki CMSIS RTOS:lle. [19]

4.4 Zephyr

Zephyr Project on Linux Foundationin avoimen lähdekoodin käyttöjärjestelmä. Myös sen tavoitteena on olla paras ja turvallisin käyttöjärjestelmä resursseiltaan rajoitetuille IoT ratkaisuille. Koska Zephyr Project on täysin avoimen lähdekoodin järjestelmä, sen vahvuuksiksi listataan mahdollisuus kaikkien osallistua järjestelmän kehitykseen ilman, että yksi henkilö tai yritys vastaa uuden päivityksen tai ominaisuuden hyväksymisestä.

Koska koko ohjelma on lisensoitu Apache 2.0 lisenssillä niin sen käyttäminen, muokkaaminen ja uudelleen julkaiseminen on täysin ilmaista, eikä Zephyr Projectilla ole maksullista versiota. Tämä tosin myös tarkoittaa, että järjestelmällä ei ole erillistä asiakaspalvelua vaan kaikki tuki on löytyy virallisesta dokumentaatiosta tai keskustelualueilta. [63]

4.5 Mongoose OS

Irlantilaisen Cesanta yhtiön Mongoose OS on myös avoimen lähdekoodin järjestelmä, mutta tarjoaa ilmaisen Apache 2.0 lisenssin lisäksi kaupallista versiota. Osa Mongoose OS:n kirjastoista ovat suljetun lähdekoodin komponentteja, mutta maksullisella lisenssillä myös näiden komponenttien lähdekoodi on rajoittamattomasti käytössä.

Mongoose OS on lähtökohtaisesti tarkoitettu IoT-sovelluksille ja sisältää sisäänrakennettuna muun muassa toteutuksen TCP- ja UPD-yhteyksille, MQTT:lle ja tiedostojärjestelmille. Mongoose OS:n pienin asennus on hieman suurempi kuin tämän vertailun pienempien järjestelmien ja vaatii minimissään 113 kilotavua tallennustilaa. [46]

4.6 Contiki

Myös Contiki OS on täysin avoimen lähdekoodin järjestelmä, jonka lisenssi on 3-pykäläinen BSD-tyylinen lisenssi [22]. Lisenssi sallii ohjelmiston käyttämisen vapaasti kaupallisessa tarkoituksessa, mutta lisenssi on liitettävä lähdekoodiin ja lisenssin lisäämisen käännetyn ohjelmiston mukana. Kaikki Contiki OS:n viralliseen versioon lisättävät osat on lisensoitava tällä lisenssillä, mutta käyttäjät voivat julkaista oman ohjelmansa suljettuna.

Contiki OS on suunniteltu IoT-laitteille ja sisältää tarvittavat verkkoyhteyspinot ja tuen useille käytössä oleville sulautetuille langattomille protokollille ja lähiverkoille. Koska Contiki on tarkoitettu verkoitetuksi järjestelmäksi niin lisäksi on kehitetty työkalu Cooja useamman laitteen yhtäaikaiseen simulointiin.

Contiki OS:n tyypilliselle ratkaisulle kerrottu tilavaatimus on 30 kilotavua ja alle 10 kilotavua RAM-muistia. [23]

4.7 Phoenix RTOS

Phoenixin ensimmäinen prototyyppi on julkaistu 2001 Varsovan teknillisen yliopiston toimesta (engl. Warsaw University of Technology) ja siitä johdettu Phoenix-RTOS 2 julkaistiin ensimmäisen kerran vuonna 2005. Käytössä on sama lisenssi kuin Contiki OS:llä eli 3-pykäläinen lisenssi jossa sallitaan lähdekoodin käyttäminen ja uudelleen levittäminen, sillä ehdolla, että lähdekoodin ja käännetyn ohjelmiston mukana toimitetaan kyseinen lisenssi. Nykyään Phoenix RTOS:n kehittämisestä vastaa Phoenix Systems yhtiö.

Phoenix RTOS on toteutettu mikroytimellä joka hoitaa muistihallinnan, säikeistyksen ja prosessien hallinnan ja säikeiden välisen yhteydenpidon ja synkronoinnin. Ydin on jaettu viiteen osaan jotka ovat HAL, virtuaalisen muistin hallinta (engl. virtual memory management), prosessien ja säikeiden hallinta (engl. process and thread management), alajärjestelmien testaus (engl. test for other subsystems) ja yleiset palvelurutiinit (engl. common routines). [52]

5. VERTAILUT

Tässä luvussa vertaillaan käyttöjärjestelmien eri ominaisuuksia yhteenvetotaulukoiden avulla. Koska käyttöjärjestelmien julkaisijat pyrkivät esittämään oman käyttöjärjestelmänsä mahdollisimman hyvässä valossa, kaikkia ominaisuuksia ei löydy kunnolla kaikille järjestelmille niiden kotisivuilta.

Taulukossa 1 esitellään järjestelmät yleisesti ja tulevissa taulukoissa pyritään avaamaan yksittäisiä ominaisuuksia tarkemmin. Esimerkiksi vuoronnus on usein hybridi ja käyttäjä voi itse valita mitä vuoronnusta käytetään tai mahdollisesti näiden yhdistelmiä.

Taulukossa 2 käydään läpi taulukon 1 parametrit ja mitä ne tarkoittavat.

Loput tästä luvusta on jaettu kappaleisiin ominaisuuksien mukaan. Kappaleessa 5.1 käydään läpi käyttöjärjestelmien verkko-ominaisuudet. Kappaleessa 5.2 esitellään käyttöjärjestelmille saatavilla oleva tuki.

Käyttöjärjestelmien tilavaatimuksista on esimerkki kappaleessa 5.3 ja lisäksi kappaleessa 5.3.1 esitellään käyttöjärjestelmien tukemia kehitysympäristöjä.

Lopuksi kappaleessa 5.4 esitellään käyttöjärjestelmien tukemat mikrokontrollerit.

5.1 Verkkotuki

Taulukossa 3 käydään läpi verkko-ominaisuuksia jos tuki on mainittu käyttöjärjestelmän kotisivuilla ja dokumentaatiossa. Useimmille järjestelmille on kuitenkin tarjolla kolmannen osapuolen tuki suurelle osalle näistä ominaisuuksista. Tässä taulukossa ei kuitenkaan oteta kantaa eri yhteysprotokolliin kuten HTTP, FTP, SSH, CoAP (engl. The Constrained Application Protocol) [13] tai MQTT.

Taulukossa 4 on avattu mitä protokollien nimet tarkoittavat.

LwIP on avoimen lähdekoodin kevyt TCP/IP-toteutus ja käyttöjärjestelmät, joille tämä on merkitty verkko-ominaisuudeksi, tukee kaikkia toteutuksia joita lwIP:llä on tuki. LwIP tukee kaikki käytetyimpiä verkkoprotokollia, mutta ei anna tukea verkkolaitteille vaan verkkolaitteet tarvitsevat omat ajurinsa ja toteutuksensa. [30]

Osalla käyttöjärjestelmistä on myös suoraan kehittäjän tuki eri pilvipalveluille. Mongoose OS mainostaa kotisivuillaan natiivia tukea AWS:lle, Microsoft Azurelle, Google Cloud Platformille ja IBM Watsonille [46].

Taulukko 1. *Käyttöjärjestelmien perustietoja*

	Mbed OS 5	FreeRTOS	ChibiOS	Zephyr	Mongoose OS	Contiki	Phoenix
Julkaisuvuosi	2009	2003	2007	2016	2016	2003	2001
Nykyinen versio	mbed-os-5.9.0	V10.0.1	18.2.1	zephyr-v1.12.0	2.3	3.0	3.0
Lisenssi	Apache2.0	MIT	GPL3 Apache2.0 Commercial	Apache2.0	Apache2.0 Commercial	3-clause BSD	3-clause BSD
Tuetut kielet	C,C++	C	C	C	C,C++,JS	C	C
Alustat	MCU	MCU, CPU	MCU	MCU	MCU	MCU	MCU
Simulaattori	Kyllä	Kyllä	Kyllä	-	-	Kyllä	-
Ytimen arkkitehtuuri	Hybrid	-	Hybrid	-	Hybrid	Hybrid	Monolithic or Micro[1]
Vuoronnin	Preemptive	Hybrid	Priority-based	Hybrid	-	Hybrid	-
Prioriteetti- tasojen määrä	7	Säädettävä	5	Säädettävä	Säädettävä	-	-

[1] Phoenix RTOS versio 2 on monoliittinen ja versio 3 on mikroytimellä toteutettu

Taulukko 2. *Parametrien selitykset*

Parametri	Selitys
Julkaisuvuosi	Vuosi milloin järjestelmän ensimmäinen versio on julkaistu virallisesti.
Nykyinen versio	Käyttöjärjestelmän uusin versio kirjoitushetkellä.
Lisenssi	Lisenssit millä järjestelmä on julkaistu.
Tuetut kielet	Ohjelmointikielet joita käyttöjärjestelmä tukee.
Alustat	Järjestelmän tukemat alustat: MCU on mikrokontrollerit ja CPU prosessoripohjaiset järjestelmät, joissa on erillinen prosessori, muisti ja lisälaitteet, esimeriksi PC.
Simulaattori	Tuki simulaattoreille tai onko käyttöjärjestelmälle saatavilla simulaattori. Lähteenä käyttöjärjestelmien viralliset verkkosivut. Viiva tarkoittaa, että tietoa ei ollut saatavilla tai käyttöjärjestelmää tukevaa simulaattoria ei ole.
OS Arkkitehtuuri	Käyttääkö käyttöjärjestelmä monoliittista-, mikro- vai yhdistelmäydintä (engl. hybrid kernel), jossa on ominaisuuksia monoliittisesta- ja mikroytimestä. Vain jos julkaisija on ilmoittanut ytimen tyypin.
Vuoronnin	Vuoronnustyyppi, voi olla preemptive, cooperative tai näiden yhdistelmä.
Prioriteettitasojen määrä	Preemptive vuoronnuksen säikeiden ja prosessien prioriteettitasojen määrä. Säädettävä on käyttäjän määritettävissä.

5.2 Asiakaspalvelu ja tuki

Taulukossa 5 esitellään käyttöjärjestelmälle tarjotut tukikanavat jotka löytyvät kotisivuilta tai jos käyttöjärjestelmän lähdekoodi on GitHubissa niin listattuna on myös GitHubin ongelmien ilmoitusalue. GitHub Issues on usein hyvä paikka tarkastaa, onko ongelmasta olemassa mahdollisesti avoin ohjelmistovirheilmoitus tai mahdollisesti korjaus ongelmaan.

Kaikkiin järjestelmiin on saatavilla useampia esimerkkiohjelmistoja joko käyttöjärjestelmän kotisivuilta tai suoraan ladattavan käyttöjärjestelmäpaketin mukana. Myös referenssimanuaali on saatavilla kaikille käyttöjärjestelmille. FreeRTOS on järjestelmistä ainut, jonka referenssimanuaali on pdf-muodossa, muilla kyseinen manuaali löytyy verkkosivuina.

GitHubin ongelmat sivulla käyttäjät voivat lisätä mahdollisia ohjelmistovirheitä ja ongelmia joihin muut voivat vastata tai kehittäjä voi reagoida korjaamalla ohjelmistovirheen. Taulukossa 6 on listattuna ongelmien määrä mitä GitHubiin on ilmoitettu kirjoitushetkellä. Taulukon sarake GitHub Repository viittaa käyttöjärjestelmän repositorioon.

Phoenix RTOS:n lähdekoodi löytyy GitHubista, mutta tällä hetkellä siellä ei ollut yhtään ongelmaa tai ohjelmistovirhettä lisättynä GitHubin Issues-sivulle. Phoenix RTOS:n kohdalla ei myöskään löytynyt muita tukikanavia kuin dokumentaatio käyttöjärjestelmän kotisivulta ja itse lähdekoodi ja sen kommentit.

Taulukko 3. *Verkko-ominaisuudet*

Käyttöjärjestelmä	Verkko-tekniikat ja protokollat
Mbed OS 5	6LoWPAN, Bluetooth, Cellular, Ethernet, LoRA, NFC, WiFi, DHCP, TCP/IP, UDP
FreeRTOS	WiFi, TCP/IP, UDP
ChibiOS	lwIP, uIP
Zephyr	Bluetooth, Ethernet, SLIP, DHCP, TCP/IP, UDP
Mongoose OS	Bluetooth, Cellular, Ethernet, WiFi, DHCP, TCP/IP, UDP
Contiki	6LoWPAN, uIP
Phoenix RTOS	lwIP

5.3 Esimerkki ROM käytöstä

Kuvaajassa 6 on esitetty kuinka paljon tilaa käännetty esimerkki ohjelmisto vie laitteen ROM:sta ja taulukossa 7 on tarkemmin esitelty mikä esimerkkiohjelma tähän on käytetty ja paljonko itse esimerkkiohjelmassa on koodirivejä. Mongoose OS:n kohdalla ohjelmaa ei ole käännetty itse vaan tieto on Mongoosen GitHubista löytyvästä käyttöoppaasta [16]. FreeRTOS esimerkissä koodirivien määrä on arvioitu noin 10 %:n virhemarginaalilla.

Phoenix RTOS kohdalla ei myöskään ole käännetty ohjelmistoa itse vaan ladattu kotisivuilta valmiiksi käännetty tiedosto ytimelle ja katsottu tämän koko kilotavuina.

Taulukon ohjelmat ei myöskään ole suoraan verrannollisia koska esimerkkiohjelmat on valittu käyttöjärjestelmien omista kirjastoista ja taulukko onkin tarkoitettu suuntaa antavaksi. Mbed OS 5:n, ChibiOS:n, Zephyrin ja Contikin kohdalla on käytetty yksinkertaisia Hello World -esimerkkejä jotka joko tulostavat tekstiä sarjaporttiin tai ohjaavat kehitysalustasta löytyvää ledivaloa.

FreeRTOS kohdalla on käytetty Amazon FreeRTOS tarjoamaa AWS IoT Coreen yhdistävää esimerkkiä. Tässä esimerkissä on siis käytössä langaton verkkoyhteys ja TLS-salattu MQTT-yhteys AWS:n pilvipalveluun. Tämän takia FreeRTOS esimerkki on lähempänä todellista ohjelmistoratkaisua kuin muut käännetyt esimerkit. FreeRTOS:n kotisivuilta kuitenkin pelkän ytimen kooksi minimalistisilla asetuksilla ja neljällä prioriteetilla on ilmoitettu viidestä kymmeneen kilotavua [31].

5.3.1 Kehitysympäristöt

Käyttöjärjestelmien kehitystyökalut vaihtelevat suuresti. Mbed tarjoaa tehokkaan komentorivityökalun Mbed CLI:n muodossa, joka kertoo tarkasti kuinka paljon muistia eri komponentit vievät, esimerkki Mbed CLI:n muistitaulukosta kuvassa 8.

Amazon FreeRTOS:n kehitystyökalut taas vaihtelevat käytetyn kehitysalustan mukaan. Tässä työssä kehitysalusta on LPC54018 IoT Module ja käyttöjärjestelmän kääntämiseen

Taulukko 4. *Yhteystyypit aakkosjärjestyksessä*

Ominaisuus	Termi avattuna	Selitys
6LoWPAN	IPv6 over Low Power Wireless Personal Area Networks	Matalan tehon likiverkkoyhteys käyttäen IPv6 [56]
Bluetooth	-	Lyhyen matkan langaton yhteys
Cellular	-	Mobiiliverkkoyhteys, esimerkiksi 3G
DHCP	Dynamic Host Configuration Protocol	Protokolla jolla voidaan asettaa TCP/IP-verkon ominaisuudet automaattisesti [14]
Ethernet	Common Local Area Network connection type	Yleinen langallisen lähiverkon teknologia
LoRa	Long Range	Alle gigahertsin taajuuksilla toimiva pitkän matkan langaton verkkoyhteys [11]
lwIP	Lightweight TCP/IP	Kevyt TCP/IP protokolla
NB-IoT	Narrowband IoT	Mobiiliverkkoa käyttävä pienen datakapasiteetin yhteys [18]
NFC	Near Field Communication	Lyhyen matkan langaton yhteys, vain muutamia senttimetrejä [24]
SLIP	IP over serial line	Zephyr OS käytetty protokolla jolla simulaattori voi ottaa yhteyden Zephyr OS-laitteisiin [63]
TCP/IP	Transmission Control Protocol/Internet Protocol	Pakollinen protokolla internettiin yhdistämiseksi [14]
UDP	User Datagram Protocol	Tiedonsiirtoprotokolla, UDP:llä tiedon eheyttä ei tarkisteta [55]
uIP	-	Ilmainen pieni TCP/IP-toteutus, tukee protokollia TCP, UDP, IP ja ARP [26]
WiFi	Wireless Local Area Network technology	Langaton lähiverkko

Taulukko 5. *Asiakaspalvelu ja tuki*

Käyttöjärjestelmä	Tukikanavat
Mbed OS 5	Discussion board, Email support [8], GitHub Issues
FreeRTOS	Discussion board, FAQ
ChibiOS	Discussion board
Zephyr	Wiki [62], GitHub Issues, Online Meetings
Mongoose OS	Discussion board, GitHub Issues, Commercial Technical Support [15], Consultancy Services
Contiki	Wiki [21], GitHub Issues
Phoenix RTOS	-

Taulukko 6. *GitHub Issues määrä*

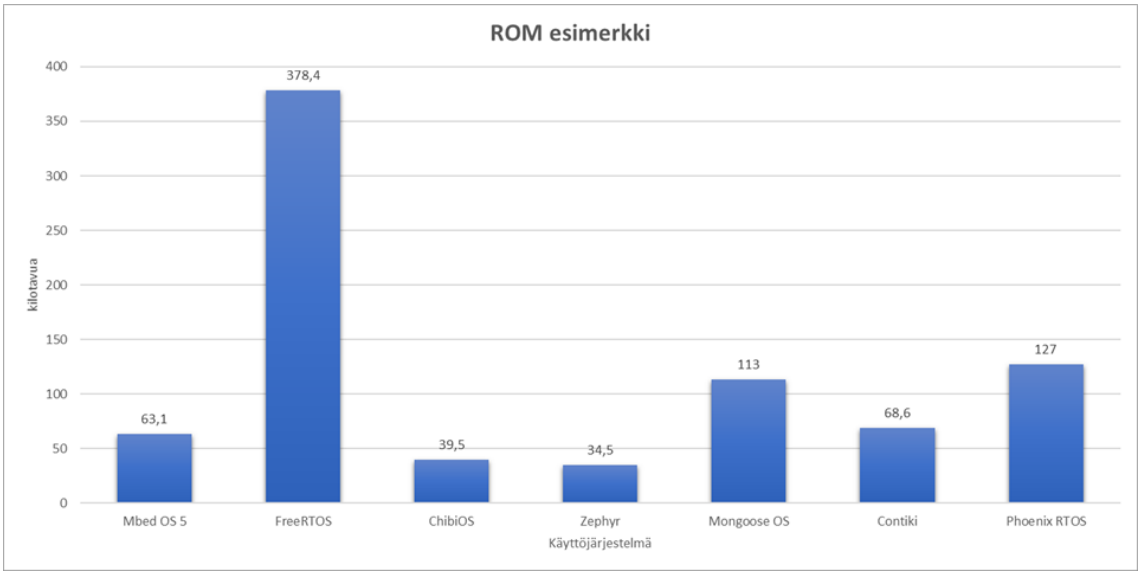
Käyttöjärjestelmä	Määrä	GitHub Repository
Mbed OS 5	332	ARMmbed/mbed-os
Amazon FreeRTOS	27	aws/amazon-freertos
Zephyr	863	zephyrproject-rtos/zephyr
Mongoose OS	43	cesanta/mongoose-os
Contiki	352	contiki-os/contiki

on käytetty NXP:n MCUXpressoa joka on ilmainen kehitysympäristö NXP:n mikrokontrollereille [48].

Contiki OS taas tarjoaa mahdollisuutta ladata virtuaalikoneen kuvatiedosto (engl. virtual machine image file), joka sisältää kaikki kääntämiseen ja simuloimiseen tarvittavat ohjelmat valmiiksi asennettuna. [23]

ChibiOS:n ja Zephyrin kääntämiseen on käytetty virtuaalikoneessa suoritettavaa Ubuntu 16.4 LTS käyttöjärjestelmää. ChibiOS kääntämiseen on tässä käytetty GNU Makea ja Makefileja. Zephyrin käännösympäristön asentamiseen on Pythonilla tehdyt työkalut ja projektien kääntämiseen on tässä käytetty CMakea ja Ninjabuildia.

Taulukossa 8 on kerättynä käyttöjärjestelmien tukemat kehitysympärisöt. Tässä taulukossa Eclipse tarkoittaa myös siihen pohjautuvia kehitysympäristöjä. Näitä on esimerkiksi NXP:n MCUXpresso, Silicon Labsin Simplicity Studio ja Texas Instrumentsin Code Composer Studio. Usein mikrokontrollerivalmistajan tarjoama kehitysympäristö perustuu Eclipseen. Kuvassa 7 on esimerkki Eclipseen pohjautuvasta IDE:stä. Ulkoasua voi muokata suhteellisen vapaasti ja tässä on lisätty vakionäkymän lisäksi FreeRTOS työt (engl. tasks) ja muuttujien seuranta virheenjäljitystä varten.



***Kuva 6**. ROM määrä eri järjestelmissä esimerkkiohjelmilla*

***Taulukko 7**. Käännetty esimerkkiohjelma kuvaajan 6 ROM käytölle*

Käyttöjärjestelmä	Ohjelma	Koodi-rivit	Alusta mille käännetty
Mbed OS 5	Mbed OS 5 Blinky	20	NUCLEO-F429ZI
FreeRTOS	Amazon FreeRTOS MQTT Example	1000	LPC54018 IoT Module
ChibiOS	RT-STM32F429ZI-NUCLEO144	83	NUCLEO-F429ZI
Zephyr	hello world example	13	X86 Qemu
Mongoose OS	example-no-libs-c app	-	-
Contiki	hello world example	15	Zolertia Z1
Phoenix RTOS	phoenix ia32.elf	-	IA32

5.4 Tuetut mikrokontrollerit

Tässä luvussa, taulukoissa 9, 10 ja 11, taulukoidaan kaikki käyttöjärjestelmän kotisivuilla listatut mikrokontrollerit, joille luvataan tuki. Huomattavaa on, että mikrokontrollereille voi yhdelle mallille olla useampia paketointeja ja näitä eri paketointien nimiä on pyritty poistamaan itse mikrokontrollereiden nimistä. Paketointi määrittää vain mikrokontrollerin fyysisen koon ja ulostulopinnien määrän ja paikan eikä vaikuta ohjelmalliseen toteutukseen.

STMicroelectronicsin nimeämiskäytännön takia esimerkiksi STM32F030, STM32F031, STM32F042, STM32F070, STM32F072 ja STM32F091 merkitään muodossa STM32F0xx. STMicroelectronicsin nimeämisessä alkuosa STM32 tarkoittaa 32-bittisiä prosessoreita ja sitä seuraava F0, F1, F2, F3, F4 ja F7 merkitsee sen olevan Cortex M -sarjan prosessori ja numero kertoo mikä Cortex M se on. Eli jos tuki on esimerkiksi STM32F031F6 niin sama

Taulukko 8. *Tuetut kehitysympäristöt*

Käyttöjärjestelmä	Kehitysympäristöt
Mbed OS 5	Mbed CLI, Kiel uVision, IAR, Eclipse, Visual Studio Code
FreeRTOS	IAR, Keil uVision, Eclipse, Rowley Crossworks, Softune
ChibiOS	Eclipse
Zephyr	Eclipse, JS Web IDE
Mongoose OS	Mos CLI, Visual Studio Code
Contiki	IAR
Phoenix RTOS	-

HAL-ajuri toimii joko ilman muutoksia tai erittäin pienillä muutoksilla minkä tahansa STM32F0-mikrokontrollerin kanssa.

Taulukoissa on poikittaisviivalla erotettu valmistajat toisistaan ja poikittaisviivan jälkeen ensimmäisenä tulee valmistaja ja sitä seuraavat rivit ovat mikrokontrollereita.

Taulukoista nähdään, että kehittäjien ilmoittama tuki eri mikrokontrollereille vaihtelee suuresti. Mbed OS 5:n laaja tuki Arm:n mikrokontrollereille selittyy sillä, että käyttöjärjestelmä on Arm:n kehittämä. Zephyrin kohdalla taas on laaja kehittäjien yhteisö ja käyttöjärjestelmän muokkaamisen helppous [63, 62].

ChibiOS tuetuissa käyttöjärjestelmissä on merkattu vain suoraan ChibiOS:n HAL:n tukemat järjestelmät. ChibiOS:n ydin ei ole järjestelmäriippuvainen paitsi kellolähteiden osin, joten sen kääntäminen uusille alustoille on suhteellisen helppoa. Tässä kuitenkin menetetään osa reaaliaikakäyttöjärjestelmän tuomista hyödyistä, koska muilta osin ohjelma on kirjoitettava suoraan valitulle mikrokontrollerille.

Phoenix RTOS:n kohdalla taas löytyi huomattavan vähän tietoa järjestelmän tukemista mikrokontrollereista. Käytännössä sivuilla mainitaan tuki vain RISC-V prosessoreille, IA32 eli PC-prosessoreille ja

Mikrokontrollereista taas selkeästi näistä valituista käyttöjärjestelmistä löytyy parhaiten tuki STMicroelectronicsin STM32-sarjalaisille ja näille löytyykin laaja tuki Mbed OS 5:llä, FreeRTOS:llä, ChibiOS:llä ja Zephyr:llä ja STM32L1xx:lle on tuki myös Contikilla. Mielenkiintoisena on myös Nordic Semiconductorsin nRF52832, jota tukee Mbed OS 5, Zephyr, Mongoose OS ja Contiki.

Taulukot ovat kuitenkin vain suuntaa antavia ja ottavat kantaa vain siihen mille laitteille tuki on ilmoitettu käyttöjärjestelmän kehittäjän sivuilla. Kolmansien osapuolien tekemiä ratkaisuja on saatavilla, mutta kaikkien näiden listaaminen ei ole tämän työn tarkoitus vaan antaa kuva käyttöjärjestelmien kehittäjien tukemista alustoista.

Taulukko 9. *Mikrokontrollerituki, osa 1*

Valmistaja/MCU	Mbed OS 5	FreeRTOS	ChibiOS	Zephyr	Mongoose OS	Contiki	Phoenix
Altera							
Nios II Gen 2				x			
Analog Devices							
ADuCM3029	x						
ADuCM4050	x						
Cadence							
Tensilica		x		x			
Cortus							
APS3		x					
Cypress							
CY8C58LP		x					
PSoC 62				x			
Espressif							
ESP32					x		
ESP8266					x		
Infineon							
TC1782		x					
XMC1x00 [2]		x					
XMC4200		x					
XMC4500		x					
Intel							
Quark S1000				x			
Maxim integrated							
MAX32600	x						
MAX32625	x						
MAX32630	x						

[2] Sisältää XMC1000, XMC1100, XMC1200 ja XMC1300

Taulukko 10. *Mikrokontrollerituki, osa 2*

Valmistaja/MCU	Mbed OS 5	FreeRTOS	ChibiOS	Zephyr	Mongoose OS	Contiki	Phoenix
Microchip							
AT32UC3		x					
AT91SAM3		x					
AT91SAM4		x					
AT91SAM7		x					
AT91SAM9		x					
Atmega128 RFA1						x	
ATmega32		x					
ATSAM3X				x			
ATSAM4S16C				x			
ATSAMA5		x					
ATSAMD20		x		x			
ATSAMD21				x			
ATSAME70Q21				x			
ATSAMV7		x					
CEC1302		x					
PIC18		x					
PIC24		x					
PIC32		x				x	
Microsemi							
A2F200M3F-FGG484		x					
SmartFusion2		x					
NORDICSemi							
nRF51822	x			x			
nRF52832	x			x	x	x	
nRF52840				x			

Taulukko 11. *Mikrokontrollerituki, osa 3*

Valmistaja/MCU	Mbed OS 5	FreeRTOS	ChibiOS	Zephyr	Mongoose OS	Contiki	Phoenix
Nuvoton							
NANO130	x						
NUC472	x						
M2351KIAAE	x						
M453VG6AE	x						
M487JIDAE	x						
NXP							
i.MX6				x			x
i.MX7				x			
LPC1114FN28	x						
LPC11U24	x						
LPC11U35	x						
LPC11U68	x						
LPC1114		x					
LPC1549	x						
LPC1768	x	x					
LPC1830		x					
LPC2368		x					
LPC4088	x						
LPC4337	x						
LPC4350		x					
LPC54114				x			
LPC546XX	x						
LPC812	x						
LPC824	x						
MC1322x						x	
MC9S12C32		x					
MC9S12DP256B		x					

Taulukko 12. *Mikrokontrollerituki, osa 4*

Valmistaja/MCU	Mbed OS 5	FreeRTOS	ChibiOS	Zephyr	Mongoose OS	Contiki	Phoenix
NXP							
MCF51CN128		x					
MCF523x		x					
MK20DX128VLH5	x						
MK20DX256VLH7	x						
MK22FN512VLH12	x						
MK60DN512VMD10		x					
MK64FN1M0VLL12	x			x			
MK66FN2M0VMD18	x						
MKL05Z32VFM4	x						
MKL25Z128VLK4	x	x		x			
MKL27Z64VLH4	x						
MKL43Z256VLH4	x						
MKL46Z256VLL4	x						
Realtek							
RTL8195AM	x						
Renesas							
RL78		x				x	
RX100		x					
RX200		x					
RX600		x					
RX700		x					
RZ/A1H	x	x					
RZ/T		x					
V850ES		x					

Taulukko 13. *Mikrokontrollerituki, osa 5*

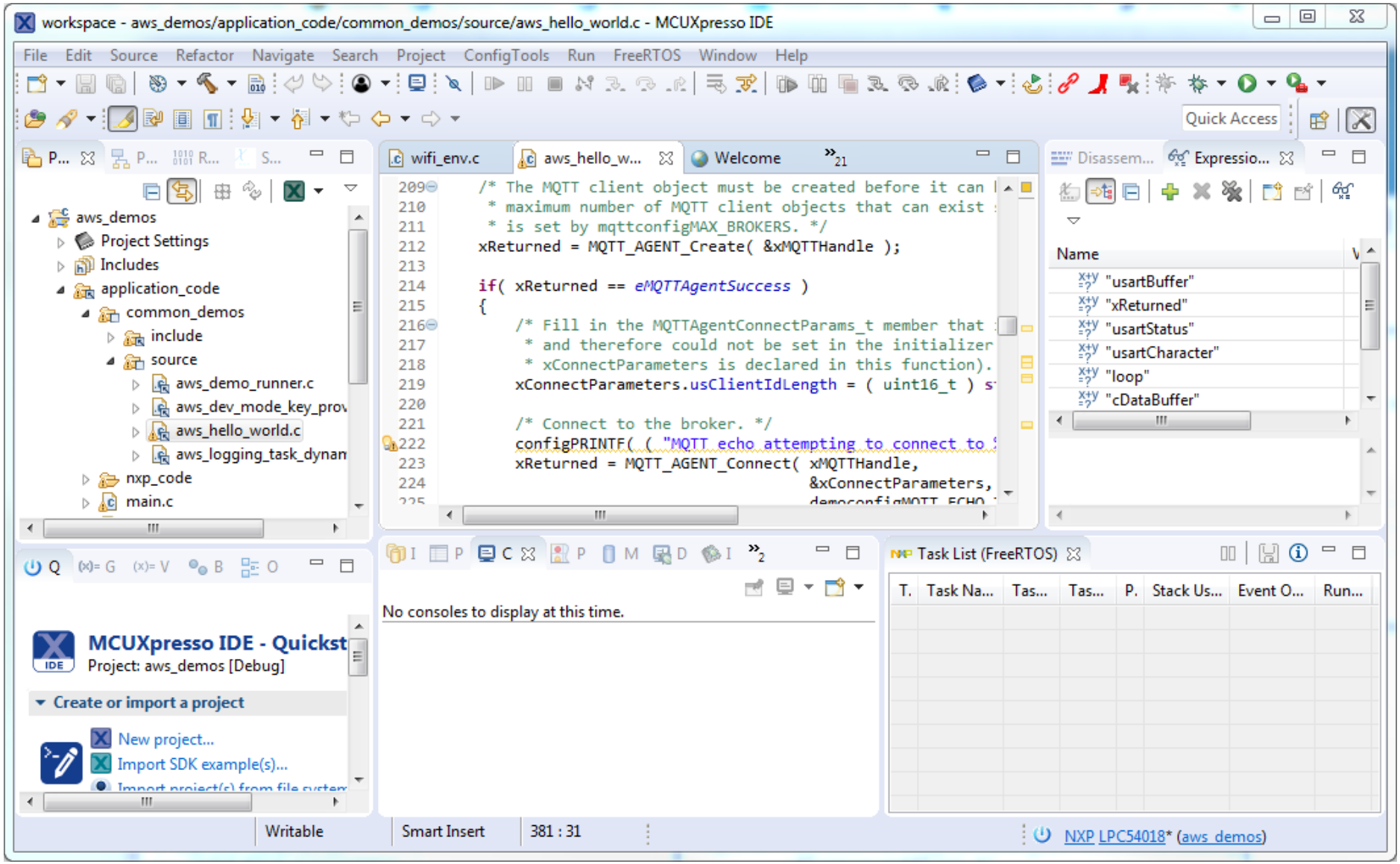
Valmistaja/MCU	Mbed OS 5	FreeRTOS	ChibiOS	Zephyr	Mongoose OS	Contiki	Phoenix
Silicon Labs							
EFM32		x					
EFM32HG322F64	x			x			
EFM32GG990F1024	x						
EFM32LG990F256	x						
EFM32PG1B200F256	x						
EFM32WG990F256	x			x			
EFM32ZG222F32	x						
EFR32FG14P233F256GM48				x			
EFR32MG12P332F1024GL125	x						
Spansion							
FM3		x					
STM							
STM32F0xx	x	x	x[3]	x			
STM32F1xx	x		x	x			
STM32F2xx	x		x	x			
STM32F3xx	x	x	x	x			
STM32F4xx	x	x	x	x			
STM32F7xx	x	x		x			
STM32L0xx	x			x			
STM32L1xx	x	x	x	x		x	
STM32L4xx	x			x			

[3] ChibiOS HAL support

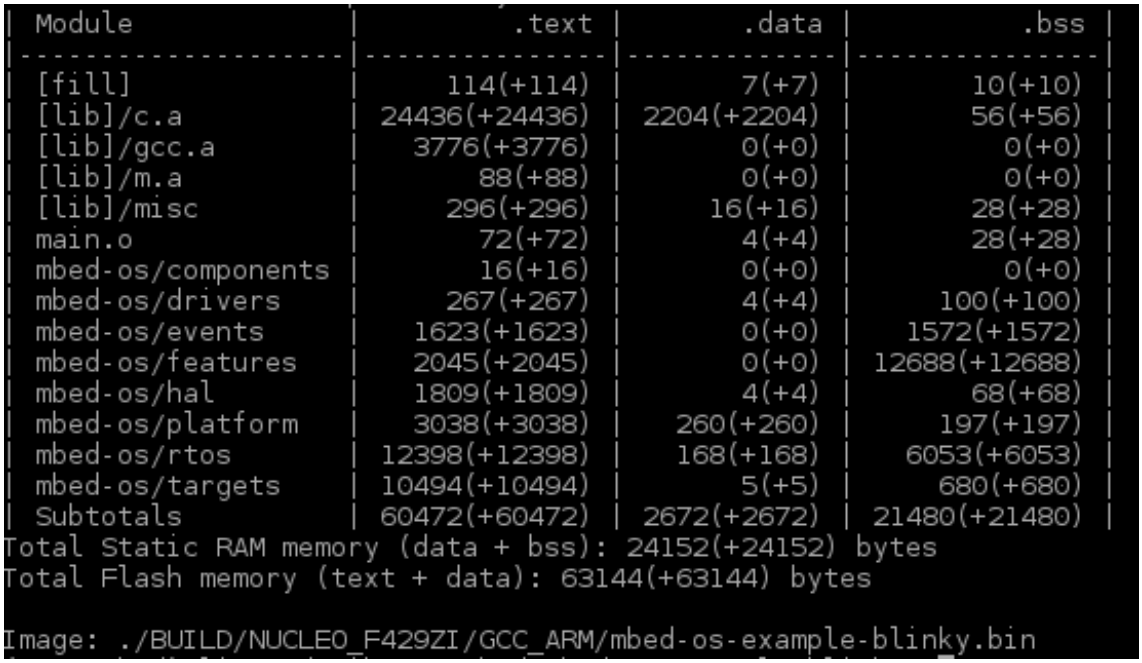
Taulukko 14. *Mikrokontrollerituki, osa 6*

Valmistaja/MCU	Mbed OS 5	FreeRTOS	ChibiOS	Zephyr	Mongoose OS	Contiki	Phoenix
Synopsys							
ARC EM[4]				x			
Texas Instruments							
CC2530						x	
CC2650				x		x	
CC3220		x		x	x		
LM3S		x					
MSP430		x				x	
MSP432		x		x			
RM48		x					
TMS570		x					
Toshiba							
TMPM066FWUG	x						
TMPM3H6FWFG	x						
Xilinx							
MicroBlaze		x					
PowerPC 405		x					
PowerPC 440		x					
UltraScale MPSoC		x					
Zynq		x					
WIZnet							
W7500	x						

[4] Sisältää EM4, EM6, EM5D, EM7D, EM9D ja EM11D ytimet



Kuva 7. *Esimerkki Eclipseen pohjautuvasta IDE:stä, NXP:n MCUXpresso*



Module	.text	.data	.bss
[fill]	114(+114)	7(+7)	10(+10)
[lib]/c.a	24436(+24436)	2204(+2204)	56(+56)
[lib]/gcc.a	3776(+3776)	0(+0)	0(+0)
[lib]/m.a	88(+88)	0(+0)	0(+0)
[lib]/misc	296(+296)	16(+16)	28(+28)
main.o	72(+72)	4(+4)	28(+28)
mbed-os/components	16(+16)	0(+0)	0(+0)
mbed-os/drivers	267(+267)	4(+4)	100(+100)
mbed-os/events	1623(+1623)	0(+0)	1572(+1572)
mbed-os/features	2045(+2045)	0(+0)	12688(+12688)
mbed-os/hal	1809(+1809)	4(+4)	68(+68)
mbed-os/platform	3038(+3038)	260(+260)	197(+197)
mbed-os/rtos	12398(+12398)	168(+168)	6053(+6053)
mbed-os/targets	10494(+10494)	5(+5)	680(+680)
Subtotals	60472(+60472)	2672(+2672)	21480(+21480)

Total Static RAM memory (data + bss): 24152(+24152) bytes
Total Flash memory (text + data): 63144(+63144) bytes

Image: ./BUILD/NUCLEO_F429ZI/GCC_ARM/mbed-os-example-blinky.bin

Kuva 8. *Mbed CLI:n tekemä taulukko muistin käytöstä*

6. SUOSITUKSET

Käyttöjärjestelmän valitsemiseen vaikuttaa usein pelkkien teknisten rajoitteiden lisäksi inhimilliset rajoitteet. Vaikka jokin käyttöjärjestelmä voi nopeasti kuulostaa soveltuvan omaan sovellukseen niin on myös otettava huomioon mahdolliset määräajat ja sovelluksen tekijät. Jos useampi käyttöjärjestelmä on sopiva ominaisuuksien puolesta niin valitsemisessa kannattaa ottaa huomioon tekijän ammattitaito ja kokemus näiden käyttöjärjestelmien kanssa ja tehdä lopullinen valinta tämän pohjalta.

Yleinen esimerkki nykyään on laitteen liittäminen pilvipalveluun kuten Amazon Web Services tai Google Cloud Services. Sovelluksessa pitää kuitenkin ottaa pilvipalveluominaisuuksien lisäksi tarvitut reaaliaikaominaisuudet. Pilvipalveluiden käyttämät liikenteen salaukset ovat raskaita ja vievät paljon laskenta-aikaa muilta suoritettavilta ohjelmilta.

Sulautettujen järjestelmien projekteissa voi tulla vastaan tilanteita missä tuotteelle ei ole vielä valittu edes käytettävää mikrokontrolleria. Tämän kaltaisissa projekteissa kannattaa käyttöjärjestelmän ja mikrokontrollerin valitseminen tehdä yhdessä. Valitaan toteutukseen sopiva mikrokontrollerityyppi ja -valmistaja sen mukaan mitä järjestelmältä vaaditaan ohjelmallisesti. Esimerkiksi jos halutaan käyttöön pilvipalvelut ja järjestelmä tekee mittauksia joilla ei ole reaaliaikavaatimuksia niin voidaan valita käyttöjärjestelmä, jolla pilvipalveluiden toteutus on helpointa ja mikrokontrolleri siten, että käyttöjärjestelmällä on täysi tuki kyseiselle kontrollerille valmiina.

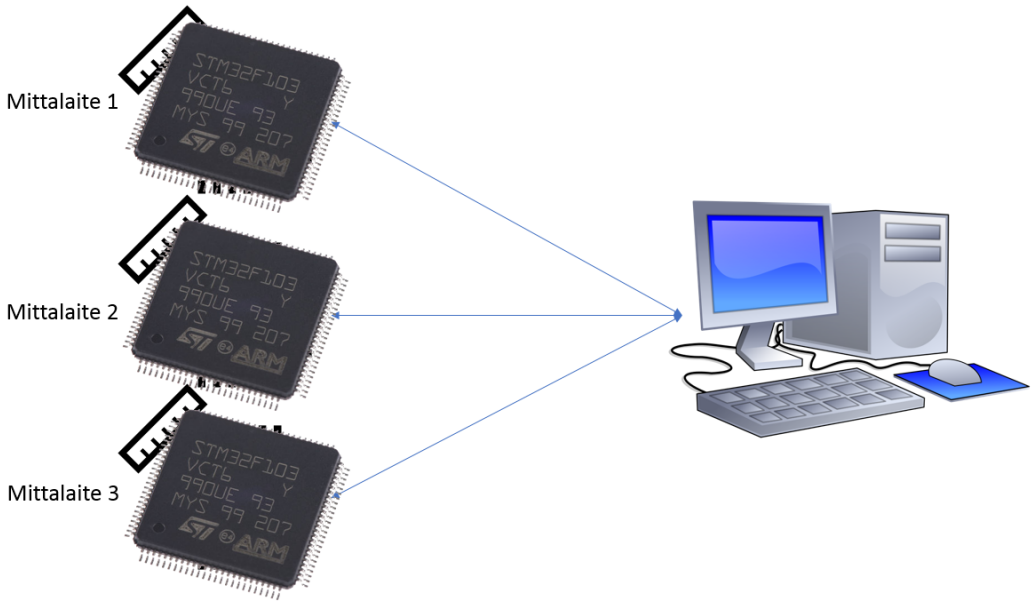
Kappaleissa 6.1 ja 6.2 on kaksi yksinkertaista IoT-sovellusta joissa laitteella ei ole tarkkoja reaaliaikaominaisuuksia. Kappaleessa 6.3 tarvitaan jo tarkempaa ajoitusta, koska ledien ohjaamiseen tarvitaan tarkkaa ajoitusta ja kaikki samassa nauhassa olevat ledit täytyy ohjelmoida kerralla ilman keskeytyksiä.

6.1 Esimerkki 1: Tietokoneeseen yhdistetty mittalaite

Kuvassa 9 on yksinkertainen esimerkki jossa mittalaitteita yhdistetään tietokoneeseen. Projektin vaatimukset esitetään kappaleessa 6.1.1. Käyttöjärjestelmältä vaadittuja ominaisuuksia rajoitetaan kappaleessa 6.1.2.

6.1.1 Laiteominaisuuksien määrittäminen

Useita mittalaitteita pitää pystyä yhdistämään ja ohjaamaan yhdellä tietokoneella helposti. Yhdistämisen täytyy olla niin yksinkertaista, että loppukäyttäjä voi liittää laitteen tietokoneeseen ja käynnistää mittauksen.



Kuva 9. *Sulautettuja mittalaitteita yhdistettynä tietokoneeseen*

Mittalaitetta ohjataan yksinkertaisilla komennoilla, jossa laitteella aloitetaan mittaus ja mittauksen jälkeen laite palauttaa tuloksen tietokoneelle. Mittaukselle voi antaa erilaisia parametreja tarvittaessa. Mittauksen aikana laitteen ei tarvitse vastata tietokoneelta tuleviin käskyihin.

Tulevaisuudessa laitteille halutaan verkkotoiminnallisuus, jolloin laitteita ohjataan etänä mistä tahansa ja kaikki mittaustulokset kerätään pilvipalveluun analysointia varten. Tämän takia elektroniikkasuunnittelija ja tilaaja on päättänyt, että laitteeseen laitetaan Ethernet, jolloin laitteelle voidaan myös syöttää virta tuon Ethernetin yli [41].

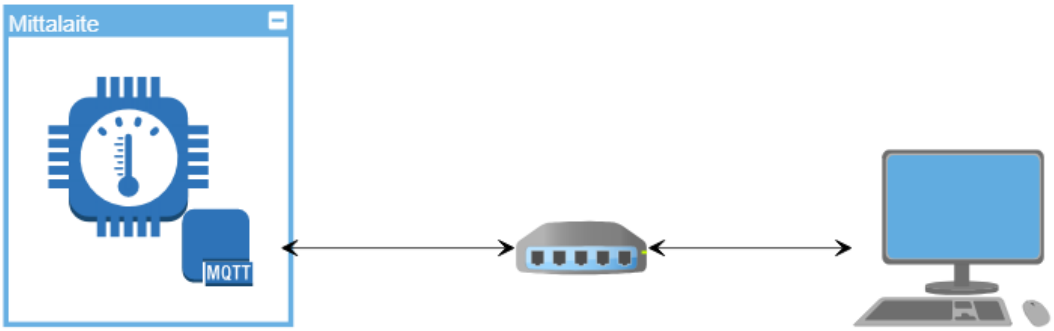
Mikrokontrollerina on päätetty käyttää STMicroelectronicsin STM32F4-sarjalaista, mutta tarkkaa kotelointia ei ole vielä päätetty.

6.1.2 Käyttöjärjestelmän ominaisuuksien rajoittaminen

Koska mikrokontrolleri on jo päätetty niin voimme suoraan rajoittaa käyttöjärjestelmät niihin joilla on luvattu tuki kyseiselle mikrokontrollerille. Taulukosta 13 voidaan suoraan katsoa STM32F4xx kohdalta, että kyseistä mikrokontrolleria tukee Mbed OS 5, FreeRTOS, ChibiOS ja Zephyr.

Koska laite on suunniteltu toimimaan verkossa niin mittalaitteiden yhdistäminen tietokoneeseen tehdään verkkokytkimen kautta, jolloin laitteita voidaan yhdistää yhtä aikaa niin monta kuin kytkimessä on paikkoja. Kuvassa 10 on esitetty uusi tapa yhdistää mittalaite tietokoneeseen.

Yhteysprotokollan on oltava kevyt ja hyvin tuettu. MQTT on ohjelmallisesti ja verkkoliikenteen kannalta kevyt ja on laajalti tuettu [53] ja käytössä sulautetuissa järjestelmissä.



Kuva 10. *Sulautettu mittalaite yhdistettynä tietokoneeseen Ethernetin yli*

Alkuvaiheessa laitteet ottavat yhteyttä mittaustietokoneella pyörivään MQTT-palvelimeen eli brokeriin.

Taulukosta 3 nähdään, että Ethernet tuki löytyy Mbed OS 5:stä ja Mongoose OS:stä jolloin näillä tiedoilla valinnaksi soveltuisi Mbed OS 5. Koska tässä tapauksessa ei tarvitse ottaa huomioon reaaliaikaominaisuuksia vaan laite kuuntelee uusia viestejä MQTT:ltä vain, jos mittaus ei ole jo käynnissä ja estää kaikki keskeytykset itse mittauksen ajaksi.

6.1.3 Suositus käyttöjärjestelmäksi

Mbed OS 5 johon on lisätty toiminnallisuus MQTT:lle Eclipsen Paho [28] pohjautuvalla kirjastolla soveltuu tähän tarkoitukseen. Mikrokontrollerin tarkempi valinta tehdään ominaisuuksiltaan hieman yli tämän sovelluksen tarpeiden tulevia pilvipalvelulaajennoksia ajatellen.

6.2 Esimerkki 2: Yhteyslaite tiedon keräämiseksi pilveen

Projektin tarkoituksena on luoda yhteysväylä valmiin laitteen ja pilvipalvelun välille. Mittalaite käyttää USART-väylää tiedon siirtämiseen. Laitteiden on tarkoitus pystyä siirtämään tietoa molempiin suuntiin. Alkuperäinen laite ohjaa uutta laitetta yksinkertaisilla käskyillä, joilla voidaan tarkistaa yhteyden tila, määrittää millä tavalla yhteys verkkoon luodaan, mihin palvelimelle yhdistetään ja käskeä lähettämään kerättyä tietoa pilvipalveluun.

Yhteyslaite taas voi pyytää tarvittaessa asetukset verkkoyhteyden muodostamiselle ja palvelimeen yhdistämiselle jos laite jostain syystä käynnistyy uudelleen ja menettää aiemmat asetuksensa.

Yhteyden pilvipalveluun pitää olla salattu ja käyttää tunnettua protokollaa. Tämän takia prototyyppiin päätetään kokeilla Amazonin pilvipalvelua joka käyttää IoT-laitteiden yhdistämiseen joko HTTP:tä tai MQTT:tä.

6.2.1 Laiteominaisuuksien määrittäminen

Tässä vaiheessa yhteyslaitteena päätetään käyttää valmista kehitysalustaa (engl. Development board) soveltuvuusselvityksen (engl. Proof of concept, PoC) tekemiseksi. Tässä suurin rajoittava tekijä on koko koska alustan pitää olla tarpeeksi pieni mahtuakseen valmiin laitteen koteloon.

6.2.2 Käyttöjärjestelmän ominaisuuksien rajoittaminen

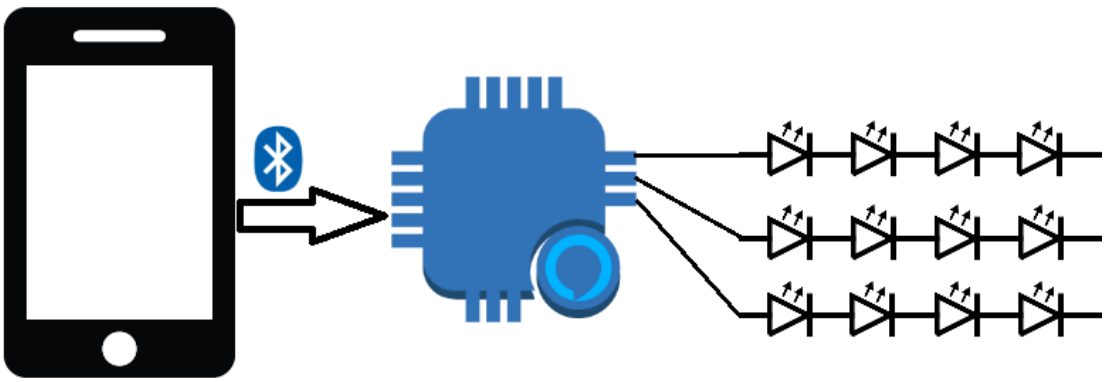
Tässä suurin ja selkein tekijä on Amazonin pilvipalveluiden käyttäminen. Mongoose OS ja Amazon FreeRTOS tarjoavat suoraan tuen AWS IoT:n MQTT:lle joten selkeästi kannattaa käyttää jompaa kumpaa näistä järjestelmistä.

6.2.3 Suositus käyttöjärjestelmäksi ja kehitysalustaksi

Tässä tapauksessa kannattaa ensin valita käyttöjärjestelmä jossa on mahdollisimman paljon valmiina eli valmiina kirjastot verkkoliikenteelle ja MQTT:lle. Tähän soveltuu Amazon FreeRTOS jolloin voidaan ladata suoraan Amazonin sivuilta paketti missä on asetukset AWS IoT MQTT:hen yhdistämiseen.

Tällä pohjalla valitaan Amazonin tukemista kehitysalustoista sopivan kokoinen laite jossa on myös valmiina langaton lähiverkko.

6.3 Esimerkki 3: Bluetooth-ohjattu ledikontrolleri



Kuva 11. *Esimerkiksi älypuhelimella bluetooth yhteys mikrokontrolleriin*

Tarkoituksena on tehdä kuvan 11 mukainen kontrolleri WS2813 ledeistä tehdylle matriisille johon voidaan joko valita valmiita valoprofiileja tai ohjelmoida uusia Bluetoothin yli. WS2813 on RGB-ledi ja siinä on valmiina pieni mikropiiri (engl. Integrated Circuit, IC) ja yksittäistä ledinauhaa ohjataan yhden digitaalisen ulostulon avulla.

Ledeistä on tarkoitus tehdä 1500 ledin matriisi jossa jokainen ledi voidaan erikseen ohjelmoida näyttämään eri väriä. Myös värien virkistystaajuudeksi halutaan 20 päivitystä sekunnissa. Kyseessä on siis eräänlainen erittäin matalan resoluution näyttö.

6.3.1 Laiteominaisuuksien määrittäminen

Yhden ledin ohjaamiseen tarvitaan kolme tavua eli yhteensä 24 bittiä dataa joka kirjoitetaan pinniin maksimissaan 800 kbps nopeudella [61]. Eli tässä huomataan, että mitä pidempi ledinauha on kyseessä niin sitä kauemmin sen kokonaan uudelleen ohjelmoiminen kestää. Tämä täytyy ottaa huomioon elektroniikkaa suunnitellessa, mutta ei vaikuta itse käyttöjärjestelmän valintaan.

Laitteelta vaaditaan myös suurta määrää keskusmuistia, yksi kuva vie tilaa 36 kilobittiä eli 4,5 kilotavua. Pienemmillä mikrokontrollereilla on yleensä vain joitakin kymmeniä kilotavuja keskusmuistia. Muistinkäyttöön tulee myös Bluetoothin käyttämä muisti eli muistia olisi hyvä olla vähintään 256 kilotavua tai enemmän.

Bluetoothin tekemä keskeytys siis pitää joko voida laittaa mikrokontrollerin keskeytysjonoon tai Bluetooth yrittää ottaa aktiivisesti haltuun suoritusta ohjelmointien välillä.

Animaatioita ei kuitenkaan tallenneta muistiin kuva kerrallaan vaan ohjelma kerrallaan jolloin tarvitaan vain muutamia kuvia ja algoritmi millä seuraavat kuvat muodostetaan.

6.3.2 Käyttöjärjestelmän ominaisuuksien rajoittaminen

Koska ledien ohjaamisessa on suhteellisen tarkat aikarajoitukset siitä kuinka pitkään pinnin pitää olla ylhäällä ja alhaalla niin ledien ohjelmoimisen ajaksi kaikki keskeytykset pitää voida laittaa pois päältä. Koska ledien ohjaus on myös koko ajan päällä niin sitä ei voi asettaa korkeammalle prioriteettitasolle kuin Bluetooth-yhteyttä, tämä ei antaisi Bluetoothille koskaan suoritusaikaa.

6.3.3 Suositus käyttöjärjestelmäksi ja kehitysalustaksi

Koska järjestelmä tarvitsee Bluetoothin niin taulukosta 3 nähdään, että kehittäjän tuki tälle löytyy Mbed OS 5, Zephyr ja Mongoose OS. Mongoose OS:n mikrokontrolleritуki on kuitenkin heikompi kuin Zephyrin tai Mbed OS 5 ja Mongoose OS:ssä on maksullisia kirjastoja joten se voidaan rajoittaa pois.

Voidaan myös todeta, kumpi tahansa käyttöjärjestelmä soveltuu tarkoitukseen, mutta Mbed OS 5 tuki on hieman laajempi. Jos kehittäjällä ei ole valmiiksi kokemusta kummastakaan käyttöjärjestelmästä niin tällä perusteella Mbed OS 5 ja sen kattava esimerkkikirjasto soveltuu tarkoitukseen paremmin.

6.4 Esimerkki 4: Venttiilien ohjaus- ja seurantajärjestelmä

Järjestelmässä on yhden mikrokontrollerin ohjattavana yhdestä viiteen venttiiliä vesiputkistossa. Venttiilejä ohjataan askelmoottorilla jonka avulla voidaan tarkasti määrittää kuinka paljon venttiiliä avataan vai suljetaanko se mahdollisesti kokonaan. Järjestelmään on myös mahdollista lisätä mittalaite jolle voidaan antaa asetuksia venttiilin automaattista ohjausta varten. Mittalaite keskustelee laitteen kanssa SPI:tä käyttäen.

Itse kontrolleria voidaan ohjata siihen kiinnitetyn näytön käyttöliittymästä tai verkkoyhteyden yli.

Tuote on suunniteltu kaupalliseksi.

6.4.1 Laiteominaisuuksien määrittäminen

Elektroniikan puolesta laite siis tarvitsee ulostuloja ja sisääntuloja moottorien tarkkaan ohjaamiseen ja lisäliittimiä mittalaitteiden asentamiseksi. Lisäksi tarvitaan ohjaus näytölle ja verkkoyhteydelle.

Verkkoyhteys toteutetaan versiosta riippuen joko mobiiliverkkoyhteytenä tai langattoman lähiverkon avulla jos sellainen on saatavilla. Laitteeseen on siis asennettava lisälaitteet näille molemmille. Verkkoyhteyden asetukset voidaan määrittää joko paikalliselta käyttöliittymältä tai asetukset voidaan tallentaa muistikortille, josta ne luetaan laitteen käynnistyessä.

Mikrokontrollerilta siis vaaditaan useampia UART- ja SPI-väyliä ja ulos- ja sisääntulopinnejä.

6.4.2 Käyttöjärjestelmän ominaisuuksien rajoittaminen

Käyttöjärjestelmän tarvitsee tässä tapauksessa tukea sekä mobiiliyhteyttä ja langattomia lähiverkkoja.

Venttiilien ohjaaminen vaatii järjestelmältä luotettavaa keskeytyksien käsittelijää, sekä mahdollisuutta estää keskeytykset venttiilien säätämisen ajaksi.

6.4.3 Suositus käyttöjärjestelmäksi

Kaikki järjestelmät tukevat reaaliaikajärjestelmien perusominaisuuksia joten näiltä osin mikä tahansa järjestelmä soveltuu tähän.

Verkkovaatimukset rajoittaa valintaa taulukon 3 perusteella Mbed OS 5 ja Mongoose OS käyttöjärjestelmiin.

Mbed OS 5 tuki taulukon 5 mukaan olisi kattavampi, mutta jos Mongoose OS:n lisensoi kaupallisella lisenssillä niin mahdollisuus käyttää käyttöjärjestelmän kehittäjän tarjoamaa asiakastukea vähentää näitä eroja huomattavasti.

Koska tässä vaiheessa ei ole vielä tarkemmin määritelty miten verkkopalvelu tarkalleen toteutetaan ja käytetäänkö mahdollisesti jotain pilvipalvelua niin Mongoose OS olisi järkevämpi valinta, koska sillä on laaja tuki eri pilvipalvelualustoille.

Koska taas käyttöjärjestelmän valinta on osa itse tuotteen kehitysprosessia niin kumpi tahansa käyttöjärjestelmä näistä kahdesta on soveltuva ja lopullinen valinta kannattaa tehdä joko tuotteessa työskentelevän ohjelmoijan kokemuksen mukaan tai suoraan valitun mikrokontrollerin pohjalta.

6.5 Esimerkki 5: Yksinkertainen moottorien kauko-ohjaus

Järjestelmällä ohjataan neljää harjatonta tasavirtamoottoria (engl. brushless DC electric motor) CAN-väylää käyttäen. Ohjaussignaali tulee itse toteutetulta radiovastaanottimelta SPI-väylää pitkin. Moottoreissa on sisäänrakennettu mikropiiri ohjausta varten.

Laitteen ohjelmistoa on tarkoitus käyttää useampien eri mikrokontrollereiden kanssa eli tarkoitus on toteuttaa moottorien ohjaaminen ja ohjaussignaalin lukeminen kirjastoina, jotta näitä voitaisiin käyttää mahdollisimman helposti myös tulevissa projekteissa vaikka mikrokontrolleri vaihtuisikin.

6.5.1 Laiteominaisuuksien määrittäminen

Työhön on jo valittu STM32F0-sarjan mikropiiri josta löytyy tarvittavat väylät moottorien ajamiseen ja radio-ohjaimen kuunteluun. Elektroniikan osalta laite on yksinkertainen.

6.5.2 Käyttöjärjestelmän ominaisuuksien rajoittaminen

Kaikki käyttöjärjestelmät tukevat tässä tarvittavia tiedonsiirtoväyliä joten se ei rajoita käyttöjärjestelmän valintaa, joten pääasiallinen rajoite on tuo valittu mikrokontrolleri.

6.5.3 Suositus käyttöjärjestelmäksi

STM32F0-sarjalle löytyy tuki käyttöjärjestelmiltä Mbed OS 5, FreeRTOS, ChibiOS ja Zephyr. Koska tarkoitus on saada mahdollisimman laaja tuki eri mikrokontrollereille niin ChibiOS rajoittaisi näistä huomattavasti eniten koska valmista tukea ei ole kuin STMicroelectronicsin mikrokontrollereille.

Mbed OS 5 tuki on rajoittunut täysin Arm:n ydintä käyttäviin mikrokontrollereihin joten jos ei ole täysin varmaa, että tulevaisuudessakin tullaan käyttämään vain näitä mikrokontrollereita niin tätä ei kannata ottaa rajoittavaksi tekijäksi.

FreeRTOS:lla ja Zephyrilla on molemmilla laaja tuki eri mikrokontrollereille joten valinta on järkevää tehdä näiden kahden välillä.

6.6 Esimerkki 6: Langaton anturiverkko

Tarkoituksena on saada rakennettua hajautettu langaton anturiverkko jonka eri solmut (engl. node) keskustelevat suoraan toisille solmuille langattomasti. Osa solmuista on yhdyssolmuja, eli laitteita joilla on myös yhteys internetiin, jotta anturiverkkoa on mahdollista lukea ja ohjata sen ulkopuolelta.

Anturiverkkoihin liittyy monia erittäinkin monimutkaisia vaatimuksia joista kuitenkin yksi tärkeimmistä on se, että anturit toimivat paristoilla hyvinkin erilaisissa ympäristöissä.

6.6.1 Laiteominaisuuksien määrittäminen

Koska laitteet toimivat paristoilla niin niiden virran käyttäminen olisi minimoitava. Virran käyttämistä voidaan rajoittaa muun muassa pitämällä laitteiden langattomien lähettimien tehot pieninä. Tämä myös rajoittaa sitä kuinka kaukana toisistaan solmut voivat olla ja kuinka nopeasti dataa solmuverkossa voidaan siirtää.

Virtavaatimus myös rajoittaa pois kaikki tehokkaammat mikrokontrollerit eli mikrokontrollerin teho ja koko on erittäin rajoittunut. Esimerkki vähätehoisista mikrokontrollereista on STM32L0-sarja jotka ovat erittäin pienen tehon mikrokontrollereita [58]. Tämän sarjan mikrokontrollereiden suurimmaksi ROM määräksi on ilmoitettu 192 kilotavua ja RAM:ia löytyy 20 kilotavuun saakka.

Laitteiden välisessä verkkoliikenteessä päätetään käyttää 6LoWPAN joka on tarkoitettu juurikin pienen tehon langattomiin likiverkkoihin.

6.6.2 Käyttöjärjestelmän ominaisuuksien rajoittaminen

Koska käytössä on STM32L0-sarjan mikrokontrolleri, niin tämä tuo myös huomattavat muisti- ja tilarajoitteet. Toinen rajoittava tekijä on 6LoWPAN:n käyttö verkkoyhteydessä.

6.6.3 Suositus käyttöjärjestelmäksi

Tässä työssä esitettyjen tietojen pohjalta ja taulukon 3 vain Mbed OS 5 ja Contiki on tuki 6LoWPAN:lle.

Taulukosta 11 taas nähdään, että vain Mbed OS 5:llä on kehittäjän tuki tälle mikrokontrollerisarjalle ja näiden tietojen pohjalta se vaikuttaa parhaalta käyttöjärjestelmältä tarkoitukseen.

Kuitenkin anturiverkon käyttöjärjestelmä valitseminen on huomattavasti monimutkaisempi projekti. V. Juvenin diplomityö [39] käsittelee aihetta laajasti ja johtopäätöksessä todetaankin, että kaikkein tärkeintä on minimoida käyttöjärjestelmän ja sen ytimen viemät resurssit anturiverkossa.

Tämä esimerkki on valittuna esimerkiksi siitä, että pelkästään käyttöjärjestelmien kehittäjien tarjoamilla tiedoilla ei aina ole järkevää tehdä rajoittamista vaan sovelluksiin täytyy erikseen tutkia tarkemmin mikä käyttöjärjestelmä toimii parhaiten resursseiltaan niukassa järjestelmässä. Mitä enemmän resursseja on käytössä, sitä suoraviivaisempi valintaprosessikin usein voi olla.

7. YHTEENVETO

Sulautettujen järjestelmien kehityksen monimutkaistuessa reaaliaikakäyttöjärjestelmän käyttäminen on usein aikaa säästävä ja järkevä vaihtoehto. Varsinkin asioiden Internetin yleistyessä ja ohjelmien kasvaessa yhä monimutkaisemmiksi mahdollisuus virheisiin kasvaa. Myös laitteiden tietoturvan kannalta on hyvä käyttää jo valmiiksi käytettyä ja testattua toteutusta, esimerkiksi juuri verkkoyhteyksille.

Tässä työssä on käyty läpi seitsemän käyttöjärjestelmän ominaisuuksia ja avattu niitä taulukoiden avulla. Lisäksi on esitetty esimerkkejä siitä, kuinka käyttöjärjestelmän valintaprosessia voi lähestyä. Kuitenkin valintaprosessiin vaikuttavat useat muut tekijät ja työn tavoitteena olikin esitellä mahdollisia käyttöjärjestelmiä ja antaa osviittaa tiedon hakemisesta.

Jos käyttöjärjestelmää etsitään järjestelmään, jossa resursseja on tarjolla riittävästi ja virrankulutus ei ole ongelma, niin valinta on helppo suorittaa. Mitä vähemmän resursseja on ja mitä tiukemmat vaatimukset tehonkulutuksen suhteen on, sitä vaikeammaksi käyttöjärjestelmän valitseminen menee. Tosin kun tutkimus on kerran tehty niin sen hyvä dokumentoiminen auttaa tulevaisuudessa samankaltaisien projektien suhteen.

Sovellukseen voi myös olla useampia sopivia käyttöjärjestelmiä joten pelkästään sovelluksen nykyhetki ei ratkaise sitä, mikä käyttöjärjestelmä kannattaa valita. Valinnassa kannattaa myös tehdä arviointi siitä, mikä suunnitellun järjestelmän käyttöikä on. Jos käyttöiäksi on suunniteltu vuosikymmeniä niin kannattaa käyttöjärjestelmää katsoa vapaan lähdekoodin puolelta. Vapaan lähdekoodin ohjelmistossa vaikka virallinen tuki olisikin käyttöjärjestelmälle jo loppunut, niin kuka tahansa voi kehittää uusia ominaisuuksia tai päivityksiä järjestelmään. Suljetun käyttöjärjestelmän kohdalla tämä ei onnistu ja kehittäjän tuen loppuessa, uudessa päivityksessä voi joutua päivittämään myös käyttöjärjestelmän eri versioon. Pahimmassa tapauksessa koko vanha ohjelma täytyy kirjoittaa uudelleen alusta saakka.

Kokonaisuudessaan sulautettujen järjestelmien ohjelmistokehitys vaatii tekijältään huomattavaa perehtymistä asiaan. Tämän työn tarkoituksena oli tutustua reaaliaikakäyttöjärjestelmätarjontaan ja avata hieman sitä ajatteluketjua, jota käyttöjärjestelmää päättäessä joutuu käymään läpi.

LÄHTEET

[1] A. Ali, S. A. Ladhake, A Comparison of Scheduling Mechanisms in commercial Real-Time Operating Systems, International Journal of Advanced Research in Computer Science, vsk. 3, nro 3, Copyright - Copyright International Journal of Advanced Research in Computer Science May 2012; Last updated - 2014-11-24, tou. 2012.

[2] Amazon, AWS IoT, Product Overview. Saatavissa (viitattu 13.9.2018): https://aws.amazon.com/iot/

[3] Amazon, FreeRTOS Homepage. Saatavissa (viitattu 25.9.2018): https://aws.amazon.com/freertos/

[4] T. N. B. Anh, S. L. Tam, Real-Time Operating Systems for Small Microcontrollers, IEEE Micro, vsk. 29, nro 5, mar. 2009, s. 30–45.

[5] Apache Software Foundation, Apache Licenses, 2004. Saatavissa (viitattu 10.9.2018): http://www.apache.org/licenses/

[6] ARM Keil, CMSIS-RTOS Keil RTX. Saatavissa (viitattu 14.9.2018): http://www2.keil.com/mdk5/cmsis/rtx/

[7] ARM Mbed, Mbed OS Homepage. Saatavissa (viitattu 14.9.2018): https://www.mbed.com/en/platform/mbed-os/

[8] ARM Mbed, Mbed OS Support. Saatavissa (viitattu 18.10.2018): https://os.mbed.com/support/

[9] Arm Mbed, mbed TLS. Saatavissa (viitattu 28.9.2018): https://tls.mbed.org/

[10] Arm MBED, Pelion Device Management, Product Overview. Saatavissa (viitattu 13.9.2018): https://cloud.mbed.com/product-overview

[11] A. Augustin, J. Yi, T. Clausen, W. M. Townsley, A Study of LoRa: Long Range & Low Power Networks for the Internet of Things, Sensors, vsk. 16, nro 9, Copyright - Copyright MDPI AG 2016; Last updated - 2017-09-15, 2016, s. 1466.

[12] J. Beningo, From bare-metal to RTOS: 5 Reasons to use an RTOS. Saatavissa (viitattu 1.10.2018): https://www.embeddedrelated.com/showarticle/1008.php

[13] C. Bormann, CoAP - Constrained Application Protocol. Saatavissa (viitattu 30.10.2018): http://coap.technology/

[14] A. Butterfield, G. E. Ngondi, A Dictionary of Computer Science, tam. 2016. Saatavissa: http://www.oxfordreference.com/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975

[15] Cesanta, Mongoose OS - Support and Integration Services. Saatavissa (viitattu 18.10.2018): https://mongoose-os.com/support.html

[16] Cesanta, mongoose-os-docs/intro.md at master. Saatavissa (viitattu 22.10.2018): https://github.com/cesanta/mongoose-os-docs/blob/master/userguide/intro.md

[17] T. B. Chandra, P. Verma, A. K. Dwivedi, Operating Systems for Internet of Things: A Comparative Study, teoksessa: Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, New York, NY, USA, 2016, ACM, ICTCS '16, Udaipur, India, s. 47:1–47:6.

[18] J. Chen, K. Hu, Q. Wang, Y. Sun, Z. Shi, S. He, Narrowband Internet of Things: Implementations and Applications, IEEE Internet of Things Journal, vsk. 4, nro 6, Dec, 2017, s. 2309–2314.

[19] ChibiOS, ChibiOS Homepage. Saatavissa (viitattu 25.9.2018): http://www.chibios.org/

[20] B. Cole, Rethinking C++ for embedded systems design, hel. 2015. Saatavissa (viitattu 19.9.2018): https://www.embedded.com/electronics-blogs/cole-bin/4438613/Rethinking-C--for-embedded-systems-design

[21] Contiki, Contiki Wiki. Saatavissa (viitattu 18.10.2018): https://github.com/contiki-os/contiki/wiki

[22] Contiki OS, Contiki Open Source License. Saatavissa (viitattu 19.9.2018): http://www.contiki-os.org/license.html

[23] ContikiOS, ContikiOS Homepage. Saatavissa (viitattu 25.9.2018): http://www.contiki-os.org/

[24] K. Curran, A. Millar, C. M. Garvey, Near Field Communication, International Journal of Electrical and Computer Engineering, vsk. 2, nro 3, Copyright - Copyright IAES Institute of Advanced Engineering and Science Jun 2012; Last updated - 2013-09-13, kes. 2012, s. 371.

[25] N. DiGiose, Barr Group's 2018 Embedded Systems Safety and Security Survey Reveals an Internet of Insecure Things, hel. 2018. Saatavissa (viitattu 10.9.2018): https://goo.gl/ucZSFa

[26] A. Dunkels, uIP - A Free Small TCP/IP Stack, Paper by Adam Dunkels on uIP TCP/IP Stack, November, 2001. Saatavissa (viitattu 21.10.2018): http://www.dunkels.com/adam/download/uip-doc-0.5.pdf

[27] Eclipse Foundation, Eclipse Public License. Saatavissa (viitattu 10.9.2018): https://www.eclipse.org/legal/

[28] Eclipse Paho, Eclipse Paho - MQTT and MQTT-SN software. Saatavissa (viitattu 26.10.2018): http://www.eclipse.org/paho/

[29] Free Software Foundation, GNU Licenses. Saatavissa (viitattu 10.9.2018): https://www.gnu.org/licenses/licenses.html

[30] Free Software Foundation, lwIP - A Lightweight TCP/IP stack - Summary. Saatavissa (viitattu 21.10.2018): https://savannah.nongnu.org/projects/lwip/

[31] FreeRTOS, FreeRTOS Homepage. Saatavissa (viitattu 25.9.2018): https://www.freertos.org/

[32] C. Haber, Simplify the Development of Secure Connected Nodes Using Cryptography-Enabled Microcontroller with DICE Architecture, June, 2018. Saatavissa (viitattu 23.10.2018): https://www.microchip.com/pressreleasepage/secure-connected-nodes-CEC1702-DICE

[33] D. Herity, Modern C++ in embedded systems – Part 1: Myth and Reality, hel. 2015. Saatavissa (viitattu 26.9.2018): https://www.embedded.com/design/programming-languages-and-tools/4438660/Modern-C--in-embedded-systems---Part-1--Myth-and-Reality

[34] G. V. Hulme, Embedded system security much more dangerous, costly than traditional software vulnerabilities, huh. 2012. Saatavissa (viitattu 10.9.2018): https://www.csoonline.com/article/2131478/critical-infrastructure/embedded-system-security-much-more-dangerous--costly-than-traditional-softwa.html

[35] IAR, IAR Integrated Software Solutions. Saatavissa (viitattu 1.10.2018): https://www.iar.com/iar-embedded-workbench/add-ons-and-integrations/integrated-software-solutions/

[36] IAR Systems, IAR Homepage. Saatavissa (viitattu 26.9.2018): https://www.iar.com/

[37] IAR Systems, IAR Support: Detecting and avoiding stack overflow in embedded systems. Saatavissa (viitattu 26.9.2018): https://www.iar.com/support/resources/articles/detecting-and-avoiding-stack-overflow-in-embedded-systems/

[38] W. Jiang, Z. Guo, Y. Ma, N. Sang, Measurement-based research on cryptographic algorithms for embedded real-time systems, Journal of Systems Architecture, vsk. 59, lok. 2013, s. 1394–1404.

[39] V. Juven, Lightweight Event-driven Real Time Operating System for Resource Constrained Connectivity, pro gradu -työ, Tampere University of Technology, Korkeakoulunkatu 10, 33720 Tampere, mar. 2017. Saatavissa: https://dspace.cc.tut.fi/dpub/handle/123456789/24796

[40] J. Liedtke, On micro-kernel construction, ACM SIGOPS Operating Systems Review, vsk. 29, nro 5, jou. 1995, s. 237–250.

[41] S. Maniktala, Power Over Ethernet Interoperability, McGraw-Hill Education, 2013.

[42] Massachusetts Institute of Technology, MIT License. Saatavissa (viitattu 10.9.2018): https://opensource.org/licenses/MIT

[43] C. Maxfield, How embedded hardware development has changed over the past 20 years, mar. 2017. Saatavissa (viitattu 10.9.2018): https://www.embedded.com/electronics-blogs/max-unleashed-and-unfettered/4459096/How-embedded-hardware-development-has-changed-over-the-past-20-years

[44] Microchip Technology Inc., Hardware Crypto Engine. Saatavissa (viitattu 23.10.2018): https://www.microchip.com/design-centers/embedded-security/technology/hardware-crypto-engine

[45] Mongoose OS, Mongoose OS. Saatavissa (viitattu 13.9.2018): https://mongoose-os.com/

[46] MongooseOS, MongooseOS Homepage. Saatavissa (viitattu 25.9.2018): https://mongoose-os.com

[47] NXP, MCUXpresso Software and Tools. Saatavissa (viitattu 26.9.2018): https://www.nxp.com/support/developer-resources/software-development-tools/mcuxpresso-software-and-tools:MCUXPRESSO

[48] NXP, Welcome | MCUXpresso SDK Builder. Saatavissa (viitattu 21.10.2018): https://mcuxpresso.nxp.com/en/welcome

[49] J. Olsson, 6LoWPAN demystified, Texas Instruments, tekn. rap., lok. 2014. Saatavissa: http://www.ti.com/lit/wp/swry013/swry013.pdf

[50] J. M. T. Onkila, Käyttöjärjestelmän edut langattomassa anturiverkossa, pro gradu -työ, Tampere University of Technology, Korkeakoulunkatu 10, 33720 Tampere, kes. 2011. Saatavissa: http://dspace.cc.tut.fi/dpub/handle/123456789/20581

[51] Pen Test Partners, Security Blog. Saatavissa (viitattu 28.9.2018): https://www.pentestpartners.com/security-blog/

[52] Phoenix Systems, Phoenix-RTOS Homepage. Saatavissa (viitattu 30.9.2018): http://phoenix-rtos.com/

[53] R. Ramos, Which IoT protocol should you use for your design?, November, 2016. Saatavissa (viitattu 25.10.2018): https://www.embedded.com/electronics-blogs/say-what-/4442973/Which-IoT-protocol-should-you-use-for-your-design-

[54] D. S, What are “co-operative” and “pre-emptive” scheduling algorithms?, October, 2013. Saatavissa (viitattu 23.10.2018): https://www.rapitasystems.com/blog/cooperative-and-preemptive-scheduling-algorithms

[55] N. M. Seel (toim.), UDP, Springer US, Boston, MA, 2012, s. 3363–3363. Saatavissa: https://doi.org/10.1007/978-1-4419-1428-6_2472

[56] Z. Shelby, C. Bormann, I. Books24x7, 6LoWPAN: The Wireless Embedded Internet, 1. aufl. p., Wiley, US, 2009.

[57] G. D. Sirio, ChibiOS RT Features. Saatavissa (viitattu 17.9.2018): http://www.chibios.org/dokuwiki/doku.php?id=chibios:product:rt:features

[58] STMicroelectronics, STM32L0 - Arm Cortex M0 Ultra-low-power MCUs. Saatavissa (viitattu 2.11.2018): https://www.st.com/en/microcontrollers/stm32l0-series.html

[59] Texas Instruments, Code Composer Studio. Saatavissa (viitattu 26.9.2018): http://www.ti.com/tool/CCSTUDIO

[60] R. Vlaming, Femto OS. Saatavissa (viitattu 10.9.2018): http://www.femtoos.org/

[61] Worldsemi, Worldsemi Co., Limited. Saatavissa (viitattu 1.11.2018): http://www.world-semi.com/

[62] Zephyr Project, Zephyr Project Documentation. Saatavissa (viitattu 18.10.2018): https://docs.zephyrproject.org/latest/index.html

[63] Zephyr Project, Zephyr RTOS Homepage. Saatavissa (viitattu 25.9.2018): https://www.zephyrproject.org/