Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2018

Lauri Karell

MIKROKONTROLLERIN ETÄPÄIVITYSJÄRJESTELMÄ

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka | Sulautetut ohjelmistot

2018 | 22 sivua

Lauri Karell

MIKROKONTROLLERIN ETÄPÄIVITYSJÄRJESTELMÄ

Opinnäytetyön tavoitteena oli kehittää käyttöjärjestelmättömälle mikrokontrollerille etäpäivitysjärjestelmä. Järjestelmän vaatimuksena oli, että laite automaattisesti jatkaa päivityksen aiheuttamasta vikatilanteessa huolimatta toimintaansa vanhalla ohjelmalla. Rajoitteena oli myös laitteen vähäinen muistin määrä.

Työssä kuvataan järjestelmän osapuolet, niiden väliset tiedonsiirron menetelmät ja päivitykseen liittyvät eri prosessit vaihe vaiheelta.

Päivitettävän laitteen muisti on jaettu niin, että sinne mahtuu bootloader, eli esilaataja, vanha ohjelma sekä uusi ohjelma. Bootloaderin, tehtävä on lukea pääohjelmien metatiedot ja niiden perusteella valita niistä toinen ajettavaksi. Ajettava ohjelma suorittaa laitteen tavallisten toimintojen, kuten esimerkiksi paikannin sijaintitietojen lähettämisen, lisäksi päivitykseen liittyvät toimenpiteet, joita ovat oman metatiedon päivittäminen, päivitysten tarkistaminen ja lataaminen. Tilanteissa, joissa virheellistä ohjelmaa ei voida ajaa, laitteen vahtiajastin käynnistää laitteen uudelleen, jolloin bootloader merkitsee ohjelman vialliseksi eikä enää yritä ajaa sitä.

Opinnäytetyö täytti asetetut tavoitteet. Työssä on kuitenkin tietoturvaan liittyviä riskejä, joiden takia järjestelmää tulee tällaisenaan käyttää vain tuotteen kehitysvaiheessa.

ASIASANAT:

mikro-ohjaimet, esineiden internet, sulautetut ohjelmistot

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communication Technology | Embedded software

2018 | 22 pages

Lauri Karell

OVER-THE-AIR UPDATES FOR A BARE METAL DEVICE

The goal of the thesis was to develop an over-the-air update system for a memory constrained device that does not have an operating system. It was required that the device keeps working normally despite update failure.

The thesis describes the device, the server and the methods used for transferring data between them. It also explains the design of the different updating procedures.

The requirements were met by dividing the device's memory into sections. These sections are the bootloader section and two main program sections. The second main program section is reserved for the updated program. It is up to the bootloader to decide which of the two programs to run by reading the metadata associated with the programs. The main program is required to keep its associated metadata up to date, and to download new firmware updates from the server. In case the updated program fails to run and the watchdog resets the device. The bootloader then detects the reset and marks the program as faulty.

The thesis meets all the requirements. It should be noted that the system in its current state has some security threats and is, thus, intended only for the development phase of the product.

KEYWORDS:

microcontrollers, Internet-of-Things, embedded software

SISÄLTÖ

KÄYTETYT LYHENTEET

ASCII	American Standard Code for Information Interchange, merkkikoodauksen standardi
CRC	Cyclic Redundancy Check, tekniikka, jonka avulla huomataan virheitä digitaalisessa datassa
GPS	Global Positioning System, paikannusjärjestelmä
GSM	Global System for Mobile Communications, matkapuhelinjärjestelmä
JTAG	Joint Test Action Group, määritelty standardi ohjelmiston ja laitteiston testaukseen
SQL	Structured Query Language, kyselykieli relaatiotietokannoille
SWD	Serial Wire Debug, ohjelmointiportti
TCP	Transmission Control Protocol, internetin yleisin tietoliikenneprotokolla
TCP/IP	Transmission Control Protocol / Internet Protocol, usean tietoverkkoprotokollan yhdistelmä
UART	Universal Asynchronous Receiver-Transmitter, piiri, jonka avulla laite voi lähettää ja vastaanottaa tietoa
UML	Unified Modeling Language, yleiskäyttöinen mallinnuskieli
USB	Universal Serial Bus, sarjaväylä, jota käytetään usein liittämään oheislaitteita tietokoneeseen

1 JOHDANTO

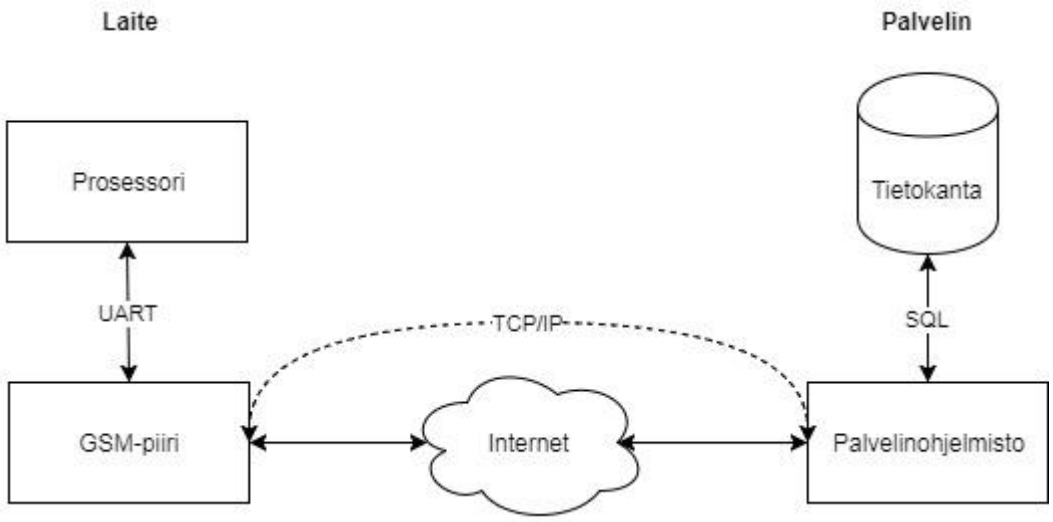
Palvelujen ylläpitäminen vaatii jatkuvaa ohjelmistokehitystä. Esimerkiksi tuotteen vaatimusten muutokset, käytössä ilmenneet ohjelmaviat ja uudet toiminnallisuudet vaativat laiteohjelmiston päivittämistä.

Kun laitteita on vain vähän, laiteohjelmiston päivittäminen on vielä mahdollista toteuttaa esimerkiksi vaihtamalla päivitetty laite vanhan tilalle tai päivittämällä laiteohjelmisto paikan päällä kannettavan tietokoneen kanssa. Kun taas laitteet ovat fyysisesti vaikeissa paikoissa, kuten paikannin ajoneuvon hanskalokeron päällä tai muualla katteiden sisällä, tai kun laitteita on paljon, ohjelmiston päivitys kannattaa toteuttaa etänä.

Sulautettujen järjestelmien etäpäivittämiseen liittyy monia haasteita. Päivitettävä laite vaatii aina jonkinlaisen yhteyden palvelimeen, josta laiteohjelmistopäivitys voidaan ladata. Usein laitteilla on myös kustannus- ja tilarajoitusten myötä heikko suorituskyky sekä rajallinen määrä muistia, jolloin niiden ei ole aina mahdollista hyödyntää etäpäivitystä valmiiksi tukevaa käyttöjärjestelmää. Lisäksi monet laitteet ovat itsenäisiä, jolloin käyttäjää ei välttämättä ole käynnistämässä niitä uudelleen vikatilanteissa.

Opinnäytetyön tavoitteena on luoda käyttöjärjestelmättömälle mikrokontrollerille etäpäivitysjärjestelmä, jota voidaan käyttää laiteohjelmiston päivittämiseen niin kehitys- kuin tuotantovaiheessakin.

Järjestelmän kaksi osapuolta ovat päivitettävä laite ja palvelin. Osapuolet ja niiden väliset yhteydet on kuvattu kuvassa 1.



Kuva 1. Järjestelmän osapuolet.

Opinnäytetyössä aluksi kuvaillaan osapuolten eri komponentteja ja avataan niiden merkitystä työssä. Tämän jälkeen käsitellään päivitykseen liittyvät eri prosessit ja kuvataan ne vaihe vaiheelta. Lopuksi vilkaistaan järjestelmän kehitysympäristöä ja sitä, kuinka järjestelmää voidaan testata.

2 PÄIVITETTÄVÄ LAITE

Päivitysjärjestelmään sopivalta laitteelta vaaditaan mahdollisuus muodostaa yhteys palvelimeen sekä mahdollisuus ajaa ladattu ohjelma. Itse laitteen käyttötarkoitus voi olla mikä tahansa.

Kehityksessä käytetty laite on GPS-paikantimen prototyyppi. Päivitykselle merkitykselliset komponentit piirilevyllä ovat STM32F103 prosessori ja SIM800 GSM-moduuli. Laitteessa on myös 5 V:lla toimiva sarjaportti, jota käytettiin kustannussyistä testausvaiheissa korvaamaan GSM-yhteys.

2.1 Prosessori

STM32F103RB on STMicroelectronicsin kehittämä 32-bittinen ARM Cortex-M3 -prosessori. Prosessorin maksimikellotaajuus on 72 MHz, mutta työssä käytetyllä laitteella prosessorin kellotaajuudeksi on asetettu 48 MHz. Käytettävää Flash-muistia prosessorilla on 128 kt. Se on laitteen ainoa muisti, jossa data säilyy virrattomanakin.

2.2 GSM-piiri

SIM800 on SIMCom Wireless Solutionin piiri, jossa on nykyaikaisille matkapuhelimille tyypilliset toiminnallisuudet, kuten tiedonsiirto, äänipuhelujen soittaminen sekä tekstiviestien lähetys ja vastaanotto. Piiri käyttää tiedonsiirtoon matkapuhelinverkkoa, ja sen maksiminopeus on 85,6 kb/s. (STMicroelectronics 2018)

Piiriä ohjataan AT-komennoilla sarjaportin välityksellä. Piiri käyttää yleisimpien komentojen lisäksi SIMComin omia AT-komentoja, jotka laajentavat piirin toiminnallisuutta tavallisten puhelinoperaatioiden ulkopuolelle. Yksi työn kannalta merkittävä näistä toiminnallisuuksista on TCP-yhteyden muodostaminen ja tiedonsiirto sen yli.

2.3 Vahtiajastin

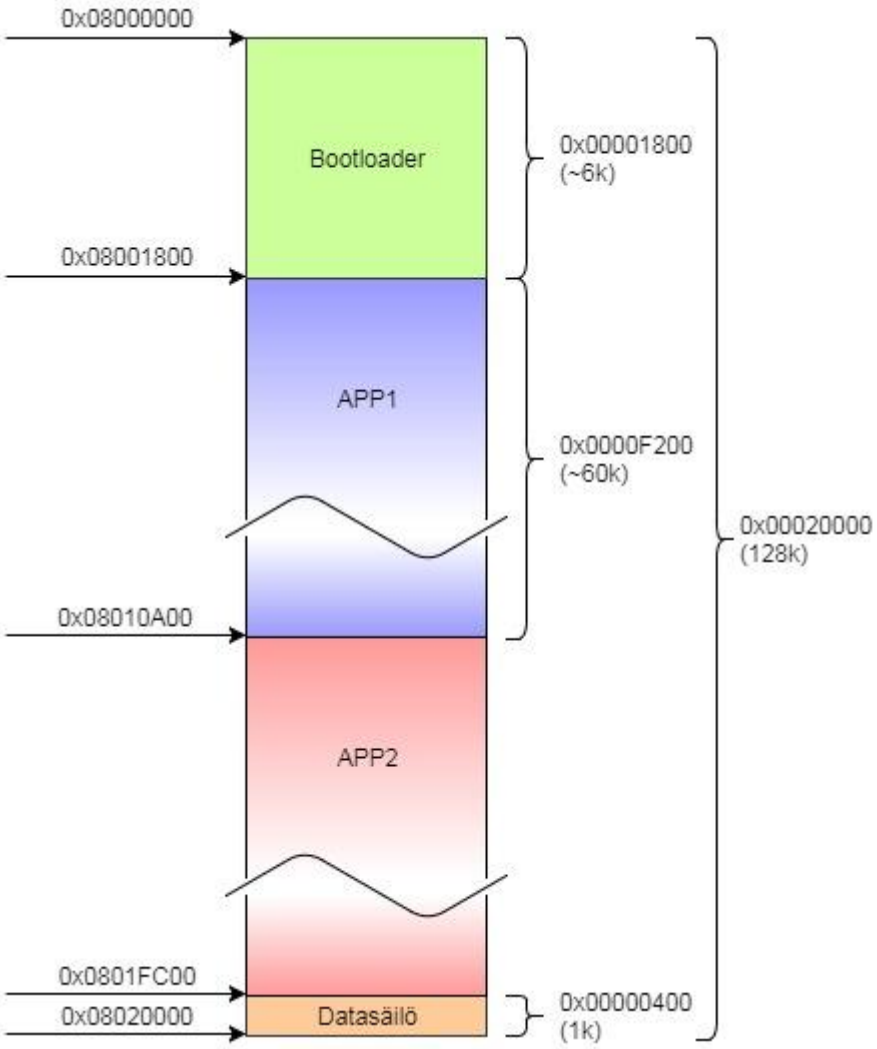
Watchdog timer eli vahtiajastin on laite, joka aiheuttaa prosessorin uudelleenkäynnistymisen laiteohjelmiston vikatilanteissa. Tavallisesti vahtiajastimella on laskuri, joka

laskee alaspäin määrätystä arvosta kohti nollaa. Prosessori lähettää ajastimelle säännöllisesti signaalin, joka palauttaa laskurin arvon alkuarvoon. Jos laskuri milloinkaan saavuttaa nollan, vahtiajastin olettaa laiteohjelmiston jumiutuneen ja aiheuttaa prosessorille uudelleenkäynnistymisen. (Barr 2001)

STM32F1-sarjan prosessoreissa on kaksi sisäänrakennettua vahtiajastinta. Independent watchdog eli itsenäinen vahtiajastin toimii tavallisen vahtiajastimen tapaan. Itsenäisellä vahtiajastimella on vain sille omistettu kello, mikä tarkoittaa että se toimii vaikka pääkello lopettaisi toiminnan. Window watchdog eli ikkunallinen vahtiajastin puolestaan on ajoitukseltaan tarkempi. Tässä yhteydessä ikkunalla tarkoitetaan aikaväliä jolloin signaali vastaanotetaan hyväksytysti. Tällöin se huomaa myös laiteohjelmiston epätavalliset hidastumiset tai nopeutumiset. (STMicroelectronics 2018)

2.4 Laiteohjelmisto

Laiteohjelmisto on se kokonaisuus, joka muodostuu laitteen ohjelmista ja niiden välisistä vuorovaikutuksista. Työssä laiteohjelmisto koostuu esilataajasta, kahdesta pääohjelmasta ja pysyvästä datasäilöstä. Laiteohjelmisto on tallennettu kokonaisuudessaan prosessorin sisäiseen flash-muistiin, joka on esitetty kuvassa 2.



Kuva 2. Prosessorin muistikartta.

2.4.1 Bootloader

Jokaisella laiteohjelmistolla on jokin aloituspiste, josta prosessori käynnistyessään aloittaa ohjelman ajamisen. Yksinkertaisimmillaan tämä on laitteelle kehitetyn monoliittisen ohjelman main-metodi. Toisinaan vaatimuksena voi olla ohjelman oikeellisuuden tarkistaminen ennen ajoa, laitteen fyysisen toimivuuden varmistaminen tai muistin salauksen purkaminen. Tämän kaltaisia toimenpiteitä tekevää aloitusohjelmaa kutsutaan bootloaderiksi eli esilataajaksi.

Työssä bootloader aluksi lukee datasäilöstä kahden pääohjelman metatiedot, ja niiden pohjalta valitsee toisen pääohjelmista ajettavaksi. Tätä päätösprosessia on kuvattu tarkemmin luvussa 5.1.

2.4.2 Pääohjelmat ja datasäilö

Laitteelle on varattu tilaa kahdelle pääohjelmalle, jotta päivityksen virhetilanteissa voidaan palata vanhempaan pääohjelmaan. Pääohjelman vaatimukset päivitysjärjestelmän osalta ovat oman tilan päivittäminen datasäilöön, ohjelmapäivitysten tarkistaminen ja niiden lataaminen.

Datasäilöön tallennetaan pääohjelmien metatiedot, jotka sisältävät pääohjelmien versionumerot ja tilat. Ohjelman tila voi olla ”empty”, ”unverified” tai ”working”. Tila on ”empty” eli tyhjä, jos pääohjelmaa ei ole tai se on todettu virheelliseksi. Tila ”unverified” eli varmistamaton tarkoittaa, että laite on ladannut ohjelman onnistuneesti, mutta sitä ei ole vielä ajettu onnistuneesti. Kun ohjelma on varmistettu toimivaksi, tila on ”working” eli toimiva.

Toimivuuden kriteerit määritellään laitekohtaisesti. Näitä kriteerejä voisi olla esimerkiksi täyden ohjelmasyklin onnistunut suoritus ja onnistunut yhteyden muodostus palvelimeen.

3 PALVELIN

Palvelin kuuntelee laitteita, vastaa niiden pyyntöihin ja tallentaa niiden lähettämiä tietoja. Laitteiden pyynnöt voivat olla esimerkiksi käyttäjän määrittämien asetusten hakeminen, tämänhetkisen sijainnin tallentaminen tai uuden laiteohjelmistopäivityksen lataus.

3.1 Ohjelma

Palvelimen yksi osa on ohjelma, joka odottaa laitteiden yhteydenmuodostusta, vastaa niiden pyyntöihin ja käsittelee tietokantaa. Ohjelman voi käsitellä usean laitteen pyyntöjä samanaikaisesti, ja se toimii yhtäjaksoisesti ympäri vuorokauden viikon jokaisena päivänä.

Ohjelma on Windows-käyttöjärjestelmällä pyörivä Windows Forms -sovellus, joka on kehitetty .NET Framework -ohjelmistoalustalla.

3.2 Tietokanta

Palvelimen toinen osa on tietokanta, johon laitteen lähettämiä tietoja tallennetaan. Näitä tietoja voidaan hyödyntää myös muissa palveluissa, kuten karttasovelluksessa ajoneuvojen paikantamisessa. Tietokantana toimii Microsoft SQL Server 2008, joka on relaatiotietokanta.

Relaatiotietokannassa tieto tallennetaan riveinä tauluihin. Nämä taulut kuvaavat yleensä olioita, kuten asiakkaita, joiden ominaisuudet ovat määritelty kolumneiksi. Kolumnilla on aina nimi, tietotyyppi ja maksimikoko, sekä mahdollisesti erikseen määriteltyjä sääntöjä. Yksi rivi vastaa yhtä olion instanssia. (Oracle 2018)

4 TIEDONSIIRTO

4.1 TCP/IP

TCP/IP (Transmission Control Protocol / Internet Protocol) on maailman yksi yleisimmin käytetyistä tietoliikenneprotokollien yhdistelmistä. Se määrittelee käytettävät standardit tiedon reitittämiselle ja verkkojen yhdistämiselle. Tällä tavalla TCP/IP eriyttää yhteyden ylläpidon toteutuksen sen käytöstä, jolloin myös useat erilaiset verkot ja koneet voidaan yhdistää keskenään. (IBM 2018)

TCP on tietoliikenneprotokolla joka huolehtii, että paketit saapuvat kohteeseen ehjänä ja samassa järjestyksessä kuin missä ne ovat lähetetty. (RFC 793/1981)

4.2 UART

UART (Universal Asynchronous Receiver-Transmitter) on piiri, jonka avulla kaksi laitetta voi vastaanottaa ja lähettää tietoa. Tieto välitetään kahden johdon avulla, jotka ovat *RX* ja *TX*. Mikrokontrollereissa on usein UART sisäänrakennettuna, mutta sarjaliikenteen voi myös toteuttaa myös ”bit-banging”-menetelmällä, jolloin prosessori suoraan ohjaa linjoja. (SparkFun 2018)

4.3 Sovelluskerroksen pakettimuoto

Jokainen viesti alkaa kahdella tavulla, jotka kertovat lähetyksen pituuden. Nämä tavut ovat verkon tavujärjestyksen mukaisesti big-endian -muodossa, joka tarkoittaa että merkittävin arvo on tavuissa ensimmäisenä.

Viestin pituuden jälkeen loppuosa voi olla joko rakenteellinen ASCII-koodattu viesti, tai binäärinen viesti. ASCII-koodattuja viestejä käytetään tavallisessa lyhyessä keskustelussa, jotta luettavuus, virheiden ratkaiseminen ja lokin pitäminen on helpompaa. Binääristä dataa käytetään vain kun lähetettävää tietoa on huomattavasti suurempi määrä, kuten ohjelmistopäivityksen latauksessa.

Jokainen laitteen lähettämä viesti sisältää laitteelle palvelimen asettaman tunnisteen, mikä tarkoittaa että keskustelu on tilaton. Tällöin viestin lähettäjä on helppo yhdistää

oikeaan laitteeseen, ja viestien välittämiseen käytetty yhteys voidaan eriyttää rajapinnaksi. Palvelimen on kuitenkin mahdollista vastata laitteen eri pyyntöihin, kuten laitteen asetusten hakuun ja uuden ohjelmaversion hakuun.

4.4 Tarkistussumma

CRC (Cyclic Redundancy Check) on menetelmä, jolla voidaan huomata tiedon virheellisyys helposti. Menetelmä perustuu polynomisen jakolaskun jakojäännökseen, jota kutsutaan tarkistussummaksi. Menetelmää käytetään usein siirrettävän tai tallennetun tiedon eheyden tarkistamiseen. (STMicroelectronics 2013)

5 PÄIVITYSPROSESSI

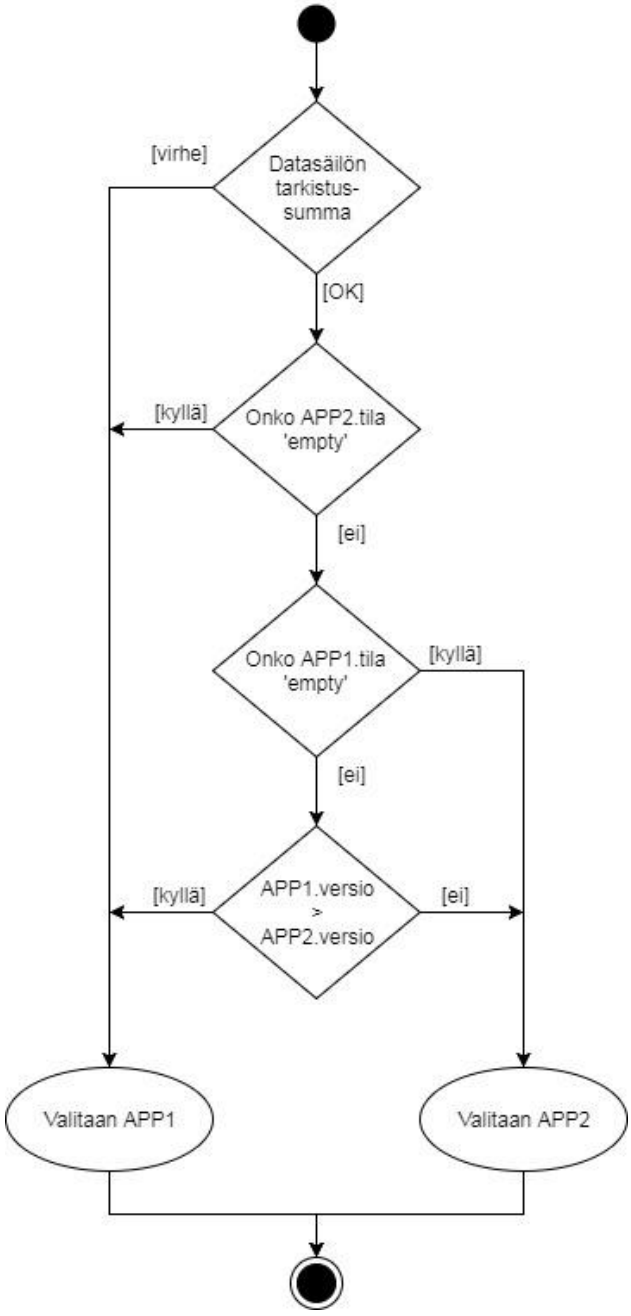
5.1 Pääohjelman valinta ja ajo

Ensimmäiseksi tarkistetaan datasäilön ehjyys laskemalla sen tarkistussumma ja vertaamalla sitä luettuun arvoon. Jos tarkistussummat eivät täsmää, pääohjelmaksi valitaan muistista ensimmäinen. Tarkistussumman laskentaan käytetty algoritmi on STM32:n CRC-tiivistealgoritmi.

Ehjästä datasäiliöstä luetaan pääohjelmien tilat ja versionumerot. Jos yksi pääohjelma on tilaltaan ”empty”, valitaan vastakkainen. Jos taas molemmat ovat tilaltaan ”empty”, valitaan muistista ensimmäinen.

Seuraavaksi verrataan versionumeroita, ja valitaan niiden perusteella uudempi ohjelma.

Tämä prosessi on mallinnettu UML-kaaviossa kuvassa 3.



Kuva 3. Aktiviteettikaavio pääohjelman valintaprosessista.
Pääohjelmat ovat ”APP1” ja ”APP2”, muistista ensimmäinen ja toinen.

Lopuksi tarkistetaan tarvittaessa myös viimeksi ajetun ohjelman viallisuus. Jos valitun ohjelman tila on ”unverified”, luetaan prosessorin rekisteristä aiheuttiko vahtiajastin uudelleenkäynnistymisen, jotta viallista ohjelmaa ei yritetä ajaa useasti. Virhetilanteessa muokataan ohjelman tilaksi ”empty” ja ajetaan vastakkainen ohjelma.

5.2 Päivityksen tarkistaminen

Laite muodostaa palvelimelle kyselyn, josta ilmenee laitteen tämänhetkisen ohjelman tunniste. Palvelin vastaa kyselyyn ohjelman uusimmalla versionumerolla. Laite aloittaa päivitysprosessin vain jos palvelimen versio on uudempi.

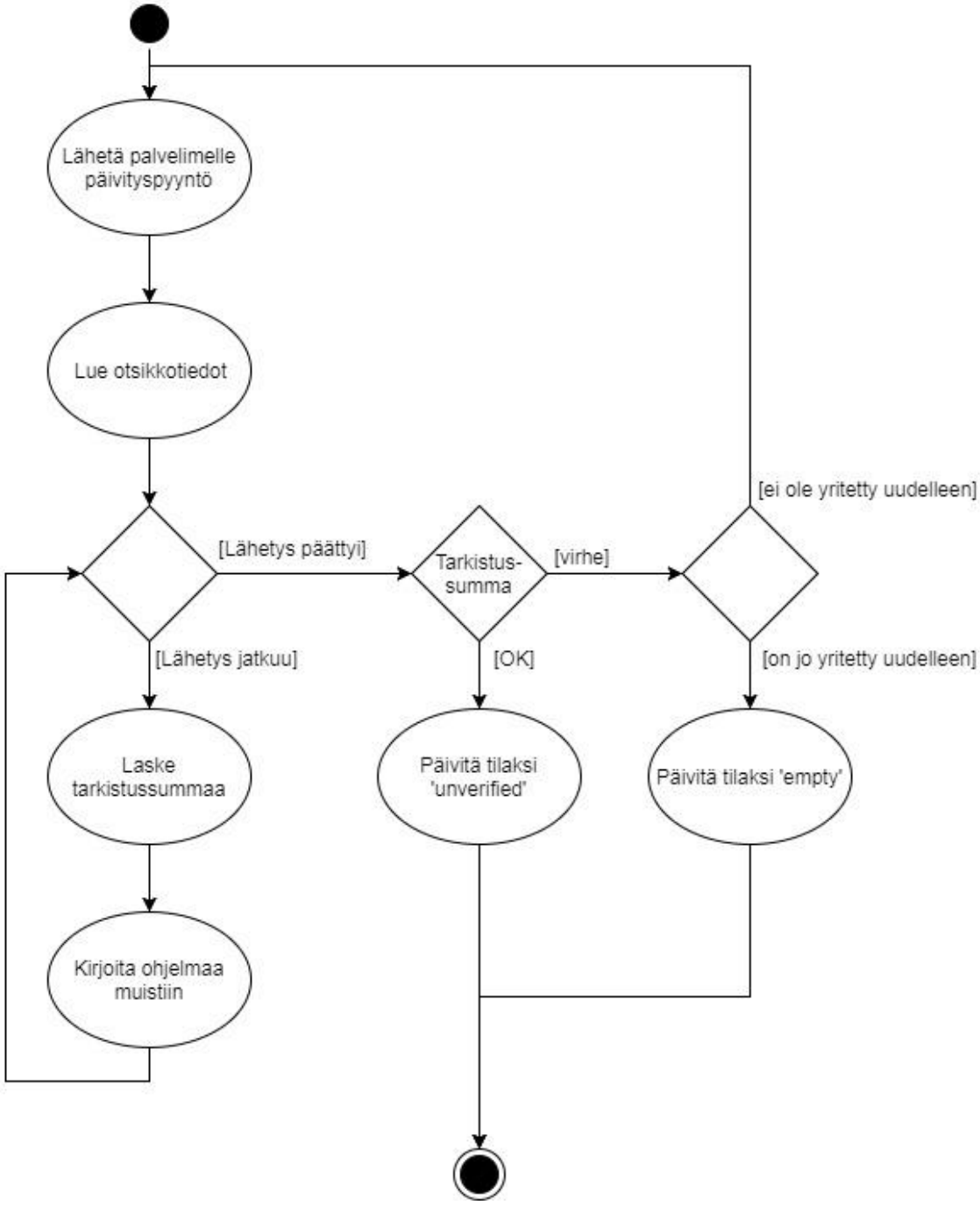
5.3 Uuden ohjelman lataus ja tarkistus

Laite muodostaa palvelimelle kyselyn, josta ilmenee ladattavan ohjelman tunniste ja versionumero. Palvelin vastaa kyselyyn ensin ohjelman otsikkotiedoilla, jotka ovat ohjelman koko ja sen tarkistussumma. Tämän jälkeen palvelin lähettää ohjelman binäärisenä 255 tavun kokoisissa paketeissa.

Laite kirjoittaa ohjelman paketti kerrallaan flash-muistiin ja laskee samalla tarkistussummaa. Kun koko ohjelma on vastaanotettu, laite tarkistaa tarkistussummat. Virhetilanteessa laite yrittää yhden kerran uudelleen.

Onnistuneen päivityksen jälkeen laite päivittää datasäilöön uuden ohjelman metatietoihin version, ja tilaksi ”unverified”. Jos taas päivitys epäonnistui toistamiseen, tilaksi tulee ”empty”.

Päivitysprosessi on mallinnettu UML-kaaviossa kuvassa 4.



Kuva 4. Aktiviteettikaavio päivitysprosessista.

6 KEHITYS

6.1 Kehitysympäristö

Piirilevyllä on SWD-portti, joka kytketään *SWDIO* ja *SWDCLK* linjoilla ST-Link/V2-ohjelmoijaan, joka on puolestaan kytketty tietokoneeseen USB-liitännällä. Tämän portin kautta prosessori on ohjelmoitavissa ja debugattavissa.

SWD-portti (Serial Wire Debug) on nastamäärältään kevyempi ohjelmointiliitäntä kuin IEEE1149.1 yhteensopiva perinteinen JTAG-portti. Tätä kahden nastan liitäntää on hyödyllistä käyttää laitteissa, joissa nastojen määrä on rajallinen, kuten ARM Cortex-M3 mikrokontrollereissa. (Ashfield ym. 2006)

Laiteohjelmistoa kehitetään C-kielellä ja ohjelmointiympäristönä toimii KEIL µVision 5. Tämän ohjelmointiympäristön avulla on mahdollista debugata laiteohjelmistoa liikkumalla lähdekoodissa rivi kerrallaan sekä lukemaan prosessorin rekistereitä ja muistialueita reaaliajassa.

Palvelimen ohjelma on kehitetty .NET Framework –ohjelmistoalustan päälle. Ohjelmointiympäristönä toimii Visual Studio 2017 ja ohjelmointikielenä C#. Palvelimen ohjelmaa ei kehitetty alusta pitäen, vaan sitä laajennettiin työn aikana kattamaan myös etäpäivittäminen.

6.2 Testaus

Laiteohjelmistoa testataan ohjelma kerrallaan. Käytännössä laitteen flash-muisti tyhjennetään aluksi, minkä jälkeen sinne ladataan bootloader ja pääohjelma. Tämän jälkeen voidaan aloittaa kumman tahansa ohjelman lähdekoodin debuggaaminen esimerkiksi breakpointtien avulla.

Palvelimen ohjelma on kehitetty testivetoisesti. Sille on erillinen NUnit-testauskehykselle kehitetty testausohjelma, johon kuuluu sekä yksikkötestejä, että integraatiotestejä. Tätä testausohjelmaa laajennettiin kattamaan myös päivitykseen liittyviä komponentteja työn aikana.

Päivityksen testauksen helpottamiseksi TCP-yhteys voidaan korvata sarjaportilla. Tällöin säästetään sekä puhelinoperaattorien tiedonsiirtokustannuksissa, että ajallisesti yhteydenmuodostuksessa. Sarjaliikennettä on näin myös mahdollista seurata logiikka-analysaattorilla.

6.3 Jatkokehitys

Tietoturvan kannalta järjestelmässä on aukkoja. Uuden päivityksen alkuperää ei ole mahdollista varmentaa, joten laitteiden kaappaus on mahdollista ohjaamalla laitteen yhteys väärennettyyn palvelimeen. Tällöin laitteelle voidaan ladata väärennetty laiteohjelmisto, jolloin hyökkääjä saa laitteen täysin haltuunsa, ja ainoa keino korjata laite on päivittää laiteohjelmisto fyysisellä yhteydellä.

Laitteen autenttisuutta ei ole myöskään mahdollista varmistaa. Tällöin hyökkääjä voi tekeytyä haluamakseen laitteeksi, ja lähettää väärennettyä tietoa.

Molemmat tietoturvauhat ovat torjuttavissa salausmenetelmillä. Tällöin on kuitenkin huomioitava laitteen suorituskyky ja muistin määrä, sillä salausalgoritmit ovat usein laskennallisesti vaativia ja kryptografiset avaimet suuria kooltaan.

Laitteelle olisi myös mahdollista kehittää vektoritaulu, jolloin laiteohjelmiston rakennetta voisi myöhemmin muokata tarpeen mukaan.

7 YHTEENVETO

Työn tarkoituksena oli luoda etäpäivitysjärjestelmä käyttöjärjestelmättömälle laitteelle, jolla on vain rajallinen määrä muistia. Laitteen tuli jatkaa toimintaansa myös päivityksen epäonnistuessa, ja sen tuli voida tehdä se ilman minkäänlaista käyttäjän syötettä.

Työssä kuvailtiin päivitettävä laite, palvelin, yhteystavat sekä päivitykseen liittyvät prosessit vaihe vaiheelta. Lopuksi esiteltiin kehitysympäristö, testausmenetelmät ja pohdittiin jatkokehitysmahdollisuuksia.

Työn tulos saavutti kaikki asetetut tavoitteet. Laitteen varmaan toimintaan ja itsenäisyyteen päästiin jakamalla laitteen muisti bootloaderille ja kahdelle pääohjelmalle. Hyödyntämällä tarkistussummaa lähetyksessä sekä vahtiajastinta ohjelman ajossa vikatilanteissa voitiin palata toimimaan vanhalla ohjelmalla. Jaetusta muistista huolimatta tilaa pääohjelmalle jäi kuitenkin riittävästi.

Järjestelmää voidaan nykyisellään hyödyntää monien erilaisten laitteiden kehitys- ja testausvaiheissa. Tulevaisuudessa järjestelmää voidaan myös kehittää tietoturvan osilta, jolloin se soveltuu käytettäväksi myös tuotteen elinkaaren tuotanto-, ylläpito- ja huoltovaiheissa.

LÄHTEET

Ashfield E.; Field I.; Harrod P.; Houlihane S.; Orme W. & Woodhouse S. 2006. Serial Wire Debug and the CoreSight Debug and Trace Architecture. Viitattu 23.4.2018 https://www.arm.com/files/pdf/Serial_Wire_Debug.pdf.

IBM 2018. Transmission Control Protocol/Internet Protocol. Viitattu 16.4.2018 https://www.ibm.com/support/knowledgecenter/en/ssw_aix_71/com.ibm.aix.networkcomm/tcpip_intro.htm.

Barr, M. 2001. Introduction to Watchdog Timers. Viitattu 9.4.2018 https://www.embedded.com/electronics-blogs/beginner-s-corner/4023849/Introduction-to-Watchdog-Timers.

Oracle 2018. About Relational Databases. Viitattu 16.4.2018 https://docs.oracle.com/cloud/latest/db112/CNCPT/intro.htm.

RFC 793/1981. Transmission Control Protocol. The Internet Engineering Task Force (IETF). Viitattu 16.4.2018 https://tools.ietf.org/html/rfc793.

SparkFun 2018. Serial Communication. Viitattu 16.4.2018 https://learn.sparkfun.com/tutorials/serial-communication/all.

STMicroelectronics 2013. AN4187 Application note. Rev 1. Viitattu 16.4.2018 http://www.st.com/content/ccc/resource/technical/document/application_note/39/89/da/89/9e/d7/49/b1/DM00068118.pdf/files/DM00068118.pdf/jcr:content/translations/en.DM00068118.pdf.

STMicroelectronics 2018. RM0008 Reference manual. Rev 18. Viitattu 10.4.2018 http://www.st.com/content/ccc/resource/technical/document/reference_manual/59/b9/ba/7f/11/af/43/d5/CD00171190.pdf/files/CD00171190.pdf/jcr:content/translations/en.CD00171190.pdf.