OULUN AMMATTIKORKEAKOULU

RUDOLF-LAITEOHJELMISTON PÄIVITYS

Thomas Grönroos
Opinnäytetyö (AMK)
Syksy 2024
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma
Ohjelmistokehitys

Tekijä(t): Thomas Grönroos
Opinnäytetyön otsikko: Rudolf-laiteohjelmiston päivitys
Työn ohjaaja(t): Ville Majava
Työn valmistumislukukausi ja -vuosi: Syksy 2024
Sivumäärä: 20

Opinnäytetyö tavoitteena oli päivittää ja modularisoida Anicare Oy:n vanha nRF9160-piirin laiteohjelmisto uuden nRF Connect SDK -version myötä sekä samalla lisätä tuki uudelle nRF9161-piirille. Työ jaettiin kolmeen vaiheeseen: Työt aloitettiin aikaisemman laiteohjelmiston ominaisuuksien kartoituksella ja moduuleiksi jakamisella. Seuraavaksi aloitettiin moduulien luominen ja yksittäistestaus. Viimeisessä vaiheessa uudelle laiteohjelmistolle luotiin testaussuunnitelma ja laiteohjelmisto testattiin suunnitelmaa vasten.

Työn lopputulokseksi saatiin modulaarinen laiteohjelmisto, joka tukee sekä nRF9160-piiriä että nRF9161-piiriä. Jatkokehityskohteita olisi lisätä yksikkötestit varmentamaan erillisten moduulien toiminnallisuutta, lisätä Flash-muistiin tallennettavat lokit, joilla kentällä olevien laitteiden virheiden tutkimista voitaisiin helpottaa. Sekä lisätä tulevaa nRF Connect SDK v2.9.0 varten toimiva *sysbuild*, sillä v2.9.0 poistaa nykyisen *child_image*-rakennustoiminnallisuuden. Myös ohjelman verkkokommunikaatioon käytettävää virtaa voitaisiin koittaa optimoida vaihtamalla kevyempään verkkoprotokollaan ja laitteelle helpommin käsiteltävään dataformaattiin.

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Information technology
Option of Software development

Author(s): Thomas Grönroos
Title of thesis: Rudolf-firmware upgrade
Supervisor(s): Ville Majava
Term and year when the thesis was submitted: Fall 2024
Number of pages: 20

The topic of this thesis was to update and modularize Anicare Oy's old nRF9160 chip's firmware with the release of nRF Connect SDK v2.7.0. The upgrade also added support for nRF9161 chip. The thesis work was conducted in three phases, first the old firmware was charted and modularized, after which the new modules were built and tested separately. Lastly a test plan was created bast on the firmware and the new firmware was tested against it. As a result, a new modularized firmware was created that includes support for both nRF9160 and nRF9161.

SISÄLLYS

SANASTO

child_image	nRF Connect SDK tarjoama ominaisuus, joka sallii useamman projektin rakentamisen yhdeksi laitekuvaksi (1).
DK	Development Kit, eli kehitysalusta on yleisesti laiteläheisessä kehityksessä käytetty (yleensä) laitevalmistajan tarjoama alusta, joka on helposti ohjelmoitavissa ja testattavissa.
FOTA	Firmware over the air, on käsite jossa laitteisto/järjestelmä voidaan päivittää lataamalla uusi laiteohjelmisto verkon yli.
IoT	Internet of Things, eli esineiden internetti on tietotekninen verkko, joka kerää ja välittää data täysin automaattisesti (2).
NVS	Non-Volatile Storage, on elektroniikassa muistityyppi, joka säilyttää informaationsa ilman jatkuvaa virran syöttöä.
RTOS	Real-time operating system, eli reaaliaikainen käyttöjärjestelmä on käyttöjärjestelmä, joka suorittaa sovellus tehtäviä annettujen aikarajoitteiden puutteessa (3).
SDK	Software development kit, on ohjelmoinnissa käytetty alustakohtainen paketti työkaluja (4).
SiP	System-in-Package, on kokoelma eri tarkoitukseen suunniteltuja sulautettuja piirejä kasattuna yhdeksi vakio paketiksi, joka suorittaa koko järjestelmän tai alijärjestelmän tehtävät (5, s. 4).
Sysbuild	Zephyr projektin tarjoama ominaisuus, jolla voidaan yhdistää useampi rakennus järjestelmä yhteen ja

halutessa tuottaa yksi yhtenäinen laitekuva näiden rakennus järjestelmien lopputuloksesta (6).

1 JOHDANTO

Opinnäytetyön tavoitteena oli päivittää Anicare Oy:n nRF9160-pohjaisten laitteiden ohjelmisto nRF Connect SDK:n (Software Development Kit) versiosta v2.3.0 versioon v2.7.0 sekä samalla rakentaa ohjelmistosta modulaarisempi ja lisätä tuki nRF9161-alustalle. Anicare Oy on oululainen teknologiayhtiö, joka valmistaa IoT-laitteita (Internet of Things) eläinten terveydentilan seurantaan ja paikantamiseen. Yritys on perustettu 2019 ja aloitti toimintansa porojen paikannuksella. Nykyisin yritys valmistaa laitteita datan keruuseen sekä villieläinten tutkimuspuolella että karjatalouden puolella eläinten terveydentilan seurantaan.

Laitteen ohjelmisto tukee useita eri ominaisuuksia eri käyttökohteisiin. Porotaloudessa laitteella voidaan paikantaa poroja ja seurata, onko poro elossa. Villieläinten tutkintaa varten laiteita voidaan konfiguroida erilaisilla paikannustiheyksillä ja karjataloudessa laitteen kiihtyvyysanturilla voidaan kerätä dataa, jonka analysoinnilla voidaan seurata eläimen terveyttä.

Pääsyynä ohjelmistopäivitykselle oli nykyisen ohjelman jäykkyys. Tämä tarkoittaa, että ohjelma on vaikea muokata ja ylläpitää, koska kaikki ohjelman komponentit ja ominaisuudet on sidottu toisiinsa kiinni. Tämän takia on vaikea ohjelmaan lisätä uusia ominaisuuksia ilman isoja muutoksia ja eri ongelmien korjaus saattaa johtaa muihin ongelmiin jossain muualla ohjelmassa.

Tästä syystä ohjelmisto päätettiin kirjoittaa uusiksi käyttäen modulaarista rakennetta pohjana. Modulaarinen rakenne on tapa toteuttaa ohjelma, jossa ohjelman eri osat ja ominaisuudet on jaettu omiin moduuleihin, jotka toteuttavat jonkun ominaisuuden. Tämä helpottaa ohjelman ylläpitämistä ja päivittämistä, sillä itsenäisten moduulien päivittämisellä ei ole vaikutusta tai vain pieni vaikutus muihin moduuleihin, myös uusien omaisuuksien lisääminen on yksinkertaisempaa, sillä ne voidaan rakentaa omillaan muista moduuleista. (7.)

2 TEORIA

2.1 Ohjelmistoarkkitehtuuri

Ohjelmistoarkkitehtuurilla kuvataan ohjelmiston rakennetta. On olemassa useita erilaisia ohjelmistoarkkitehtuureja, joilla on omat hyvät ja huonot puolensa. Esimerkkejä eri ohjelmistoarkkitehtuureista on komponenttipohjainen, monoliittinen applikaatio, tapahtumapohjainen, MVC (Model-View-Controller). (8.)

Arkkitehtuurit ovat yleisesti ottaen jaettavissa kahteen kokonaisuuteen monoliittinen ja hajautettu, esimerkiksi komponenttipohjainen arkkitehtuuri on hajautettu rakenne. Tarkoittaen, että ohjelmisto on jaettu moduuleihin. Komponenttipohjaisen arkkitehtuurin etu on joustavuus, eli ohjelmistoa on helpompi päivittää ja ylläpitää. Vastaavasti monoliittisen applikaatio on monoliittinen arkkitehtuuri ja sen suurin etu on yksinkertaisuus. (9.)

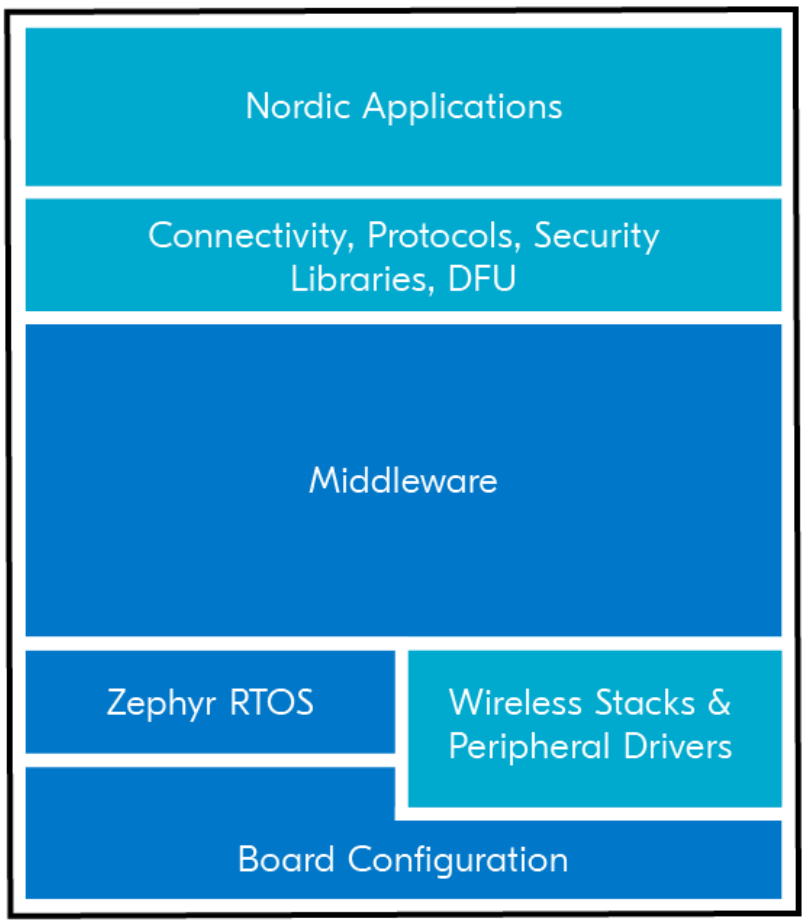
Uuden laiteohjelmiston pääpiirteeksi valittiin modulaarisuus. On olemassa useampia eri arkkitehtuureja, jotka ovat modulaarisia rakenteeltaan. Esimerkiksi komponenttipohjainen ja mikropalvelut ovat modulaarisia arkkitehtuureja. Tässä tapauksessa kaikkein läheisin arkkitehtuuri on komponenttipohjainen arkkitehtuuri, sillä ohjelmisto on jaettu useisiin ohjelman sisäisiin moduuleihin, jotka toteuttavat ohjelman eri ominaisuuksia. Mikropalveluissa taas ohjelmisto on yleensä rakennettu useasta pienemmästä erillisestä ohjelmasta, jotka kommunikoivat keskenään käyttäen jotain kevyttä kommunikaatioprotokollaa.

2.2 Kehitysympäristö

Uuden laiteohjelmiston kehitykseen käytettiin Nordic Semiconductorin nRF9160- ja NRF9161DK:ta (Development Kit), DK:t pohjautuvat vastaavasti nRF9160- ja nRF9161-SiP:hin (System-in-Package). DK:n ohjelmointiin käytettiin Nordicin nRF Connect SDK:ta, joka on rakennettu Zephyr RTOS:n (Real-time Operating System) päälle.

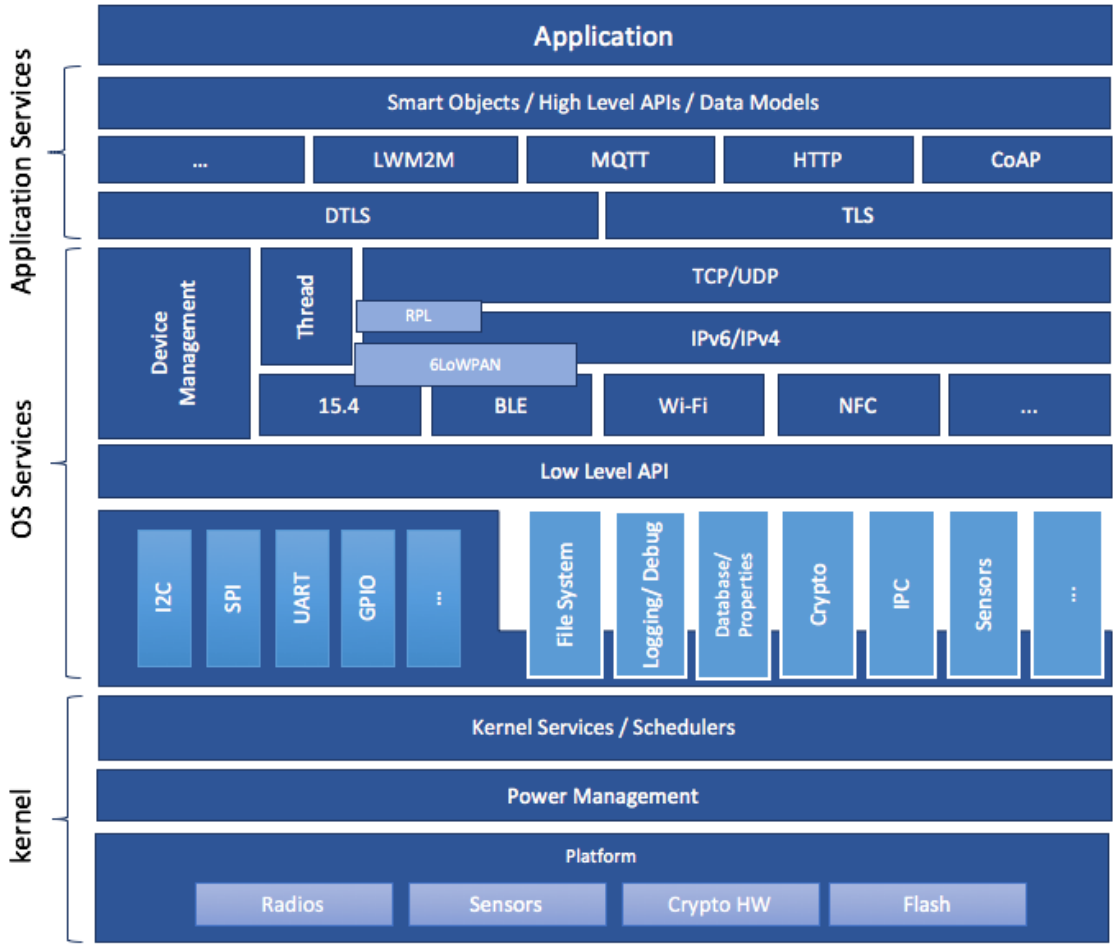
nRF9160- ja nRF9161-SiP ovat Nordic Semiconductorin kehittämiä vähävirtaisia mobiiliverkkossa toimivia IoT-laitteita. Molemmat piirit tukevat LTE-M- ja NB-IoT-verkkoja sekä sisältävät GNSS-vastaanottimen (Global Navigation Satellite System) radion sisään rakennettuna. Piirien prosessorina toimii 64 MHz:n ARM Cortex-M33. Piireillä on 1 megatavu sisäistä Flash-muistia ja 256 kilotavua SRAM-välimuistia. Tämän lisäksi nRF9161 tukee myös uudempaa 3GPP Release 14 -standardia ja DECT NR+ Mesh -verkkoa. (10; 11.)

nRF Connect SDK on Nordic Semiconductorin kehittämä kehitysalusta heidän nRF52-, nRF53-, nRF70- ja nRF91-sarjan piireille. Kuvassa 1 on yleisen tason kuvaus nRF Connect SDK:n arkkitehtuurista. nRF Connect SDK on rakennettu Zephyr RTOS:n ympärille ja sisältää useita eri malliapplikaatiota ja esimerkki toteutuksia eri protokollapinoista. nRF Connect SDK lisää muun muassa tuen Bluetooth Mesh-, ANT-, Thread-, Matter-, LTE-M- ja NB-IoT-protokollapinoille. (12.)



KUVA 1. nRF Connect SDK -arkkitehtuuri (13)

Zephyr on matalajalanjälkinen käyttöjärjestelmäydin, joka on suunniteltu muistirajoitteisille järjestelmille sekä mikrokontrollereille. Kuvassa 2 on ylätason kuvas Zephyrin arkkitehtuurista. Zephyr koostuu useista eri alijärjestelmistä, jotka on suunniteltu olemaan modulaarisia ja konfiguroitavissa sekä muokattavissa ja laajennettavissa erilaisten sulautettujen järjestelmien vaatimuksiin. Zephyr on rakennettu pääsääntöisesti C-kielellä ja käyttää West-rakennusjärjestelmää. Zephyrin konfigurointityökaluina toimivat pääasiassa Kconfig, CMake ja Devicetree. (14.)



KUVA 2. Zephyr-arkkitehtuuri (15)

2.3 Käytetyt ohjelmointikielet

Työssä käytettyjä ohjelmointikieliä olivat C, Python, Bash ja PowerShell. Projektin konfiguraation käytettiin pääsääntöisesti CMake- ja Kconfig-tiedostoja.

C-kieli on 1970-luvulla kehitetty yleiskäyttöinen ohjelmointikieli. C-kieltä käytetään erityisesti käyttöjärjestelmäydinten ja laiteajureiden kirjoittamiseen. C-kieli on myös hyvin suosittu sulautetuissa järjestelmissä. (16.)

Python on 1980-luvun loppu puolella kehitetty yleiskäyttöinen korkean tason ohjelmointikieli, joka on julkaistu vuonna 1991. Python on dynaaminen garbage-collected-kieli, joka tukee useita eri ohjelmointiparadigmoja kuten proseduraalista- ja olio-ohjelmointia. (17.)

Bash ja PowerShell ovat skriptikieliä vastaavasti Bash- ja PowerShell-komentotulkeille. Bash on Unix-maailmassa erittäin yleinen skriptikieli eri tehtävien automatisaatioon ja PowerShell ajaa vastaavaa asemaa Windows-ympäristössä. Bash komentotulkki on julkaistu 7.6.1989. Se on rakennettu C-kielellä ja lisensoitu GPLv3:n alla. Bash on alustariippumaton, mutta ei yleisesti käytetty Windows-ympäristöissä. (18.) PowerShell on Microsoftin kehittämä automatisointi- ja konfigurointihallintaohjelma, joka muodostuu komentotulkista ja skriptikielestä. PowerShell on julkaistu ensimmäisen kerran 14.11.2006 Windows-komponenttina. 18.8.2016 PowerShellin lähdekoodi julkaistiin avoimena-lähdekoodina ja ohjelmasta tehtiin järjestelmäriippumaton. (19.)

3 LAITEOHJELMISTON TOTEUTUS

Laiteohjelmiston rakennus toteutettiin kolmessa osassa. Rakennus aloitettiin vanhan ohjelman kartoituksella ja uuteen ohjelmaan tarvittavien ominaisuuksien suunnittelulla. Toisessa osassa oli varsinainen laiteohjelmiston rakentaminen ja eri moduulien erillistestaus. Viimeiseksi luotiin laiteohjelmistosta testisuunnitelma ja laiteen ominaisuudet testattiin suunnitelmaa vasten.

3.1 Uuden arkkitehtuurin kuvaus

Uusi laiteohjelmisto on jaettu useisiin pieniin moduuleihin kuten *Modem*, *Network* ja *NVS* (Non-Volatile Storage), jotka voidaan ottaa ja poistaa käytöstä erikseen. Osa moduuleista on kuitenkin riippuvaisi toisten moduulien toiminnallisuudesta. Näiden moduulien päälle on sitten rakennettu ohjelmiston päälogiikka, joka on vastuussa eri tehtävien aloittamisesta ja eri moduulien tuottaman tiedon käsittelystä.

Projektin konfiguraation on jaettu moduulikohtaisiin Kconfig- ja CMake-tiedostoihin, joissa moduulit voivat lisätä uusia konfiguraatioasetuksia ja C-lähdekooditiedostoja. Näillä uusilla konfiguraatioasetuksilla voidaan sitten ottaa käyttöön eri moduuleja, niiden ominaisuuksia ja muuttaa eri ominaisuuksien asetuksia. Aikaisempaan projektiin nähden on myös projektin pääkonfiguraatiota helpotettu käyttämällä vain yhtä pääkonfiguraatiotiedosto ja laitekohtaisia overlay-tiedostoja, jotka ylikirjoittavat vain tarvittavia ominaisuuksia.

Uudessa arkkitehtuurissa laitteen Flash-muisti on jaettu kolmeen osaan, joista yksi osa on varattu kahteen tarkoitukseen. Flash-muistin pääosiot ovat *mcuboot_primary*, *nvs_storage* ja *mcuboot_secondary*, joka on jaettu *data_storage*-osion kanssa. *NVS*:lle varattu *nvs_storage* ja *data_storage* ovat tarkoitettu laitekonfiguraatiotietojen tallentamiseen (*nvs_storage*) ja väliaikaisten mittaustulosten tallentamiseen (*data_storage*). Applikaatiokoodi ja vakiomuuttuja on sijoitettu *mcuboot_primary*-osioon ja FOTA-päivityksiä (Firmware over the air) varten on *mcuboot_secondary*.

3.2 SDK:n päivitys

SDK:n päivittäminen versiosta 2.3.0 versioon 2.7.0 ei projektin kannalta suurempia muutoksia vaatinut. Isoin muutos oli versiossa 2.7.0 lisätty uusi ”hardware model v2”, joka sallii yhden laudan tukea useampaa eri prosessoria/prosessorikonfiguraatiota. Projektin kohdalla tämä tarkoittaa, että yrityksen mukautettupiiri voidaan määritellä projektissa vain kerran ja lisätä tähän sitten tarvittaessa vain tuki eri prosessoreille ja niiden ominaisuuksille.

Hardware model v2 vaati jo olemassa olevien mukautetun piirin tiedostojen uudelleenorganisointi ja -nimeämistä sekä perusinformaation uudelleen konfigurointia. Tämän lisäksi uuden mallin käyttäminen edellytti laudan prosessorien määrittämisen kokonaan uusiksi käyttäen uusia konfiguraatiotiedostoja.

3.3 Ohjelmarakenteen päivitys

Laiteohjelmiston rakenteen päivittäminen aloitettiin edellisen version rakenteen läpikäymisellä ja pilkkomisella eri ominaisuuskokonaisuuksiin, josta sitten rakennettiin ylätason etenemissuunnitelma. Tämän etenemissuunnitelman perusteella isot kokonaisuudet jaettiin pienemmiksi palasiksi, jotka voitiin sitten alkaa toteuttaa uudessa laiteohjelmistossa eri moduuleiksi.

Verkkokommunikaatio jaettiin kahteen pääosaan, laitteen modeemin hallintaan ja verkkoyhteyden hallintaan. Uuden ohjelman Flash-moduuli on jaettu kahteen alamoduuliin, jotka ovat vastuussa oman Flash-alueen alustamisesta ja tarjoavat tarvittavat käyttöliittymät datan kirjoittamiselle ja lukemiselle alueeltaan. Kiihtyvyysanturi- ja GNSS-datan keruuosioissa luotiin ohjelman käyttämien sensoreiden ajurit ja käyttöliittymät. Laitteen GNSS-vastaanotin on nRF91 piireillä modeemin sisään rakennettu ja jakaa suoritusajan verkkokommunikaation kanssa. Tästä syystä, kun laite hakee GPS-paikkaa, täytyy muiden verkkoprosessien odottaa haun loppumista. Kiihtyvyysanturina toimii erillinen lis2dw12-piiri, joka on liitetty laitteeseen käyttäen I2C-sarjaliikenneväylää (Inter-Integrated Circuit).

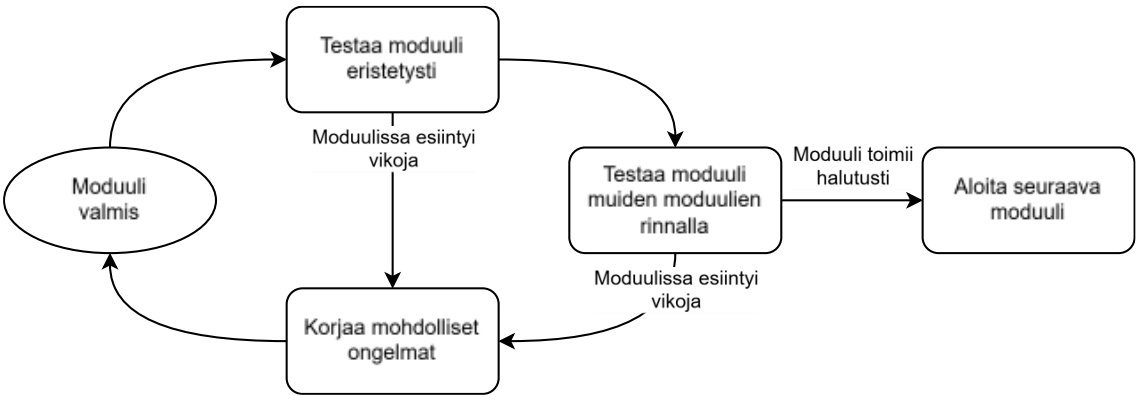
3.4 Keskeisimmät ominaisuudet

Laiteohjelmiston keskeisimpiä ominaisuuksia ovat verkkokommunikaatio, paikannus, paikannussykli ja kuolinilmaisu. Rakenteellisina ominaisuuksina kaikkein keskeisin on uuden laiteohjelman modulaarisuus. Ohjelmisto-ominaisuuksilla on eri tärkeys eri käyttökohteissa, mutta kaikissa tapauksissa verkkokommunikaatio on olemassa datan tuomiseen ja laitteiden päivittämiseen.

Porotalouden puolella laitteen keskeisimmät ominaisuudet ovat paikannus ja kuolinilmaisu. Kuolinilmaisu on ominaisuus laitteella, joka ilmoittaa onko eläin elossa, mahdollisesti kuollut taikka kuollut. Tätä käytetään porotaloudessa, jotta poron omistajat voivat käydä keräämässä poron kuolemakorvauksen. Villieläinten tutkimuksessa paikannussyklin vaihdettavuus ja datan tuonnin luotettavuus ovat isommassa asemassa kuin kuolinilmaisu. Karjataloudessa laitetta käytetään eläinten hyvinvoinnin seurantaan käyttäen kiihtyvyysdataa.

3.5 Testaus

Ohjelmiston testaus suoritettiin kahdessa osassa. Ensimmäisessä osassa testaus toteutettiin saman aikaisesti uusien moduulien ja ominaisuuksien luonnin aikana. Kuvassa 3 on esitelty kehitysvaiheentestaus. Kun moduuli oli saatu testattavaan kuntoon, se testattiin eristetysti muista moduuleista, joita ei ole merkitty moduulin riippuvuuksin. Tämän jälkeen moduuli testattiin muiden moduulien rinnalla.



KUVA 3. Moduulien kehitysvaiheentestauskaavio

Testauksen toisessa osassa koko ohjelmistosta luotiin testaussuunnitelma, jota vastaan sitten kaikki laitteen ominaisuudet testattiin. Toisessa testausvaiheessa koko ohjelma testattiin, sekä nNR9161- ja nRF9160DK:lla että yrityksen mukautetulla levyllä. Tässä vaiheessa tehtiin myös virranmittaukset uudelle laiteohjelmistolle, jolla varmistettiin, että laitteen virran kulutus ei ole kasvanut päivityksen ohessa.

4 TULOKSET

Uuden laiteohjelman toteutuksessa ilmennyt suurin ongelma oli, että laiteohjelman uusi Flash-osiointi ei ole yhteen sopiva vanhan laiteohjelman kanssa. Koska uusi Flash-osiointi ei toimi vanhan version kanssa, täytyi uuteen versioon lisätä väliaikaisesti tuki vanhalle Flash-osioinnille, tukemaan vanhoja laitteita. Väliaikainen osiointi voidaan sitten jättää pois käytöstä, kun kaikki vanhalla versiolla rakennetut laitteet ovat saapuneet elinkaarensa loppuun.

SDK:n päivittämisen yhteydessä myös huomattiin, että vanha laiteohjelmisto ei ole aikaisemmin itse alustanut ”Modem Library” -kirjastoa, vaan on turvautunut aikaisemman version tarjoamaan automaattiseen alustukseen. Tästä syystä uudessa versiossa täytyi muistaa ”Modem Library” alustaa manuaalisesti ennen nRF91 laitteiden modeemin käyttöä.

Varsinaisen laiteohjelmiston lisäksi projektin aikana luotiin uusia skriptejä helpottamaan testaus- ja kehitysprosessia sekä päivitettiin vanhoja Bash skriptejä toimimaan uuden ohjelmarakenteen kanssa. Tämän lisäksi päivitetyistä skripteistä tehtiin uudet Windowsilla toimivat natiivi versiot käyttäen PowerShell-skriptejä.

5 POHDINTA

Projektin päätavoitteena oli päivittää Anicare Oy:n laiteohjelmisto nRF Connect SDK:n versioon v2.7.0 sekä lisäksi oli tavoitteena lisätä tuki nRF9161-alustalle ja modularisoida vanha laiteohjelmisto.

SDK:n päivitys ja nRF9161 alustan tuen lisääminen saavutettiin täysin, mutta koska vanhan laiteohjelmiston ominaisuudet olivat niin toisiinsa sidottuja, on modulaarisuudessa edelleen kohtia, joita voitaisiin parantaa.

Vanhan laiteohjelman FOTA-päivittämisen tukemisessa oli joitain haasteita. Uudessa laiteohjelmassa Flash-muisti on osioitu uusiksi paremmin tukemaan laitteen nykyisiä ja mahdollisia tulevia käyttö kohteita, mutta uusi Flash-osiointi ei ole yhteen sopiva vanhan laiteohjelmiston osioinnin kannassa. Koska uusi osiointi ei ole yhteen sopiva vanhan laiteohjelmiston osioinnin kanssa, täytyy tämä ongelma kiertää käyttämällä vanhaa Flash-osiointia nykyisille laitteille ja asentaa vain kaikki tulevat laitteet käyttämällä uutta osiointia.

Kokonaisuutenaan projekti saavutti asetetut tavoitteet ilman suurempia ongelmia. Lopputuloksena saatiin ohjelmaversio, jota voidaan päivittää ja muokata eri käyttötarkoituksiin huomattavasti helpommin kuin aikaisempaa laiteohjelmistoa. Uudella versiolla on tuki nRF9160- ja nRF9161-piireille sekä vanhan ohjelman päivittämiselle jo kentällä oleville laitteille.

Projektin jatkokehityksen kannalta oleellisia asioita olisi kentällä olevien laitteiden lokien tallentaminen Flash-muistiin, jotta laiteilta voitaisiin sitten lukea tarvittaessa virhekoodeja ja muuta informaatiota ongelmien ratkaisemiseksi.

Laitteen verkkokommunikaatiota voitaisiin koittaa optimoida käyttämällä kevyempää kommunikaatioprotokollaa ja kevyempiä datapaketteja. Nykyinen JSON-paketin muodostaminen ja lähettäminen kuluttaa kaikkein eniten laitteen virrasta, jos laite saa GNSS-sijainnin nopeasti. Jos dataformaatti olisi pienempi ja helpommin laitteelle käsitellä, säästäisi tämä lähetykseen tarvittavaa aikaa ja näin myös laitteen virtaa.

Tulevaisuudessa täytyy myös saada Zephyrin Sysbuild-ominaisuus toimimaan mukautetulla levyllä, sillä nRF Connect SDK v2.9.0 poistaa tuen nykyiselle child_image-ominaisuudelle ja korvaa sen Sysbuildilla. Tämän lisäksi projektille kannattaa tulevaisuudessa kehittää yksikkötestit, helpottamaan virheiden huomaamista ja varmistamaan, että ohjelma toimisi suunnitellusti.

LÄHTEET

1. Nordic Semiconductor 2024. Multi-image builds using child and parent images. Luettavissa: https://docs.nordicsemi.com/bundle/ncs-2.7.0/page/nrf/config_and_build/multi_image.html. Luettu: 9.10.2024
2. Wikipedia 2024. Internet of things. Luettavissa: https://fi.wikipedia.org/wiki/Esineiden_internet. Luettu: 3.9.2024.
3. Wikipedia 2024. Real-time operating system. Luettavissa: https://en.wikipedia.org/wiki/Real-time_operating_system. Luettu: 4.9.2024.
4. Wikipedia 2024. Software development kit. Luettavissa: https://en.wikipedia.org/wiki/Software_development_kit. Luettu: 29.8.2024.
5. Pfahl, Jr., R. C. & Adam, J. 2005. System in Package Technology. iNEMI. Luettavissa: https://thor.inemi.org/webdownload/Industry_Forums/Productronica_2005/Floor_Pres/SIP.pdf. Luettu: 3.10.2024.
6. Nordic Semiconductor 2024. Sysbuild (System build). Luettavissa: https://docs.nordicsemi.com/bundle/ncs-2.7.0/page/zephyr/build/sysbuild/index.html. Luettu 9.10.2024.
7. Bråtegren, K. & Kulläng, R. 2024. Software Modularity – an introduction to Strategic Software Modularization. Modular management. Luettavissa: https://www.modularmanagement.com/blog/software-modularity. Luettu: 3.10.2024.
8. David Garlan and Mary Shaw 1994. An Introduction to Software Architecture. Carnegie Mellon University, School of Computer Science. https://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf. Luettu: 8.10.2024
9. Chandler Harris 2024. Microservices vs. monolithic architecture, When monoliths grow too big it may be time to transition to microservices. Atlassian Luettavissa: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith. Luettu: 8.10.2024
10. Nordic Semiconductor 2024. nRF9160. Luettavissa: https://www.nordicsemi.com/Products/nRF9160. Luettu: 3.9.2024.
11. Nordic Semiconductor 2024. nRF9161. Luettavissa: https://www.nordicsemi.com/Products/nRF9161. Luettu: 3.9.2024.

12. Nordic Semiconductor 2024. nRF Connect SDK. Luettavissa: https://docs.nordicsemi.com/bundle/ncs-2.7.0/page/nrf/index.html. Luettu: 4.9.2024.
13. DevAcademy. nRF Connect SDK structure and content. Luettavissa: https://academy.nordicsemi.com/courses/nrf-connect-sdk-fundamentals/lessons/lesson-1-nrf-connect-sdk-introduction/topic/nrf-connect-sdk-structure-and-content/. Luettu: 26.11.2024
14. Zephyr 2024. Introduction. Luettavissa: https://docs.zephyrproject.org/latest/introduction/index.html. Luettu: 4.9.2024.
15. Zephyr. Zephyr Security Overview. Luettavissa: https://docs.zephyrproject.org/3.7.0/security/security-overview.html. Luettu: 26.11.2024
16. Wikipedia 2024. C (programming language). Luettavissa: https://en.wikipedia.org/wiki/C_(programming_language). Luettu: 3.9.2024.
17. Wikipedia 2024. Python (programming language). Luettavissa: https://en.wikipedia.org/wiki/Python_(programming_language). Luettu: 3.9.2024.
18. Wikipedia 2024. Bash (Unix shell). Luettavissa: https://en.wikipedia.org/wiki/Bash_(Unix_shell). Luettu: 3.9.2024.
19. Wikipedia 2024. PowerShell. Luettavissa: https://en.wikipedia.org/wiki/PowerShell. Luettu: 3.9.2024.