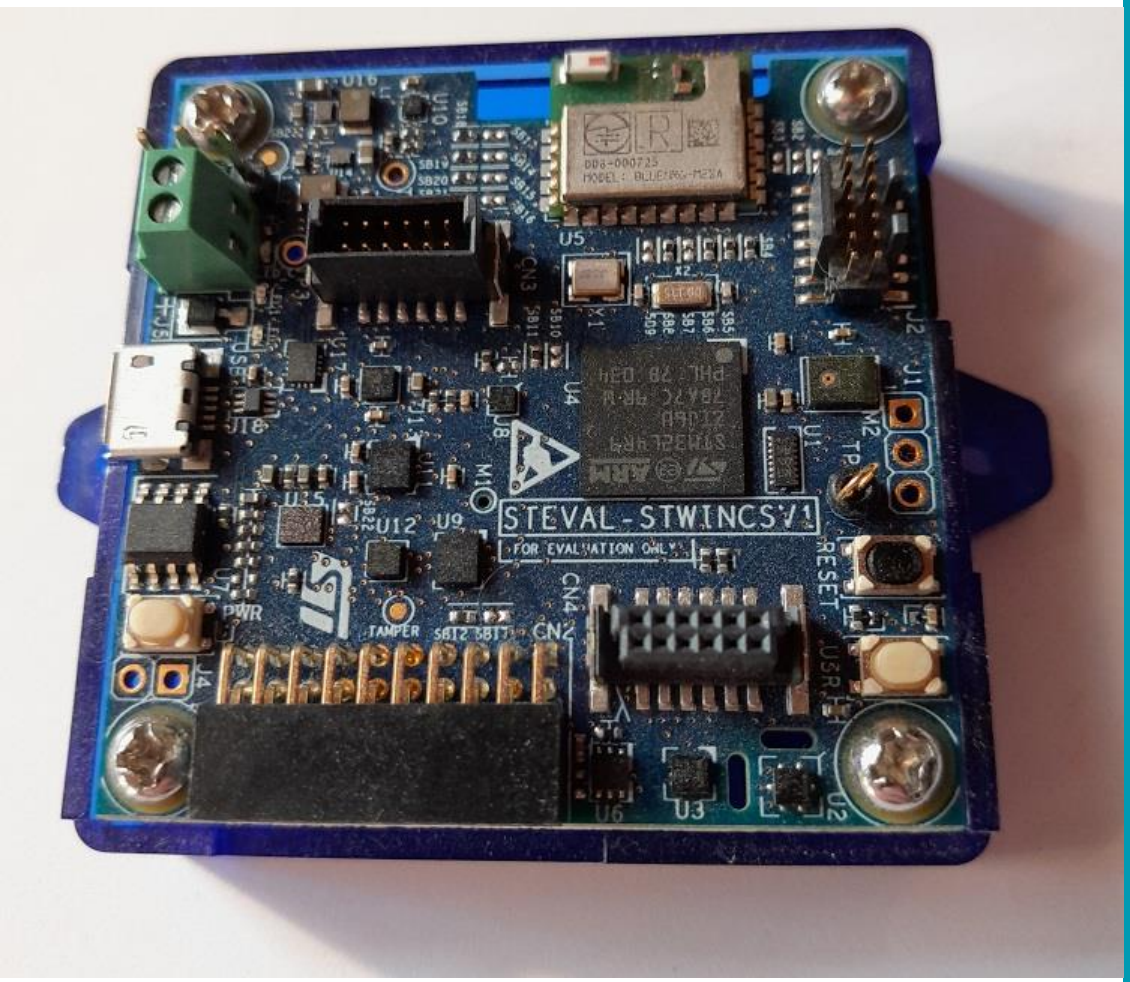
Tanya Jaakkola

Tekoäly osana teollisuuden älykästä kunnonvalvontaa ja ennakoivaa vikadiagnostiikkaa



Insinööri (AMK)

Tieto- ja viestintätekniikka

Syksy 2021 / Kevät 2022

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Jaakkola Tanya

Työn nimi: Tekoäly osana teollisuuden älykästä kunnonvalvontaa ja ennakoivaa vikadiagnostiikkaa

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: tekoäly, koneoppiminen, ennakoiva vikadianostiikka, teollisuuden älykäs kunnonvalvonta, Industry 4.0, STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B, konenäkö

Opinnäytetyö on tehty Kajaanin Ammattikorkeakoulun tilauksesta. Opinnäytetyölle keskeisiä aihepiirejä ovat tekoäly, koneoppiminen, konenäkö, ennakoiva vikadiagnostiikka, teollisuuden älykäs kunnonvalvonta sekä Industry 4.0. Opinnäytetyön yhtenä tavoitteena on luoda kattava teoriapohja opetusmateriaaliksi opinnäytetyön keskeisistä aihepiireistä. Opinnäytetyön toinen tavoite on perehtyä STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B-kehitysalustan ominaisuuksiin, koneoppimisytimeen sekä testata sen toimintaa ja laatia ohjeet kehitysalustan käytölle.

Opinnäytetyön teoriaosuuden taustamateriaalit kerättiin pääsääntöisesti verkkolähteistä ja materiaaleina käytettiin tutkielmia, artikkeleita ja eri yritysten esittämiä käsityksiä aihepiireistä. Materiaalia lähestyttiin siltä kannalta, että miten siitä saataisiin luotua opetusmateriaaliksi sopivaa aineistoa. Opinnäytetyössä testattavan kehitysalustan koneoppimisytimen ominaisuuksia testattiin keräämällä laitteella anturidataa, jota hyödyntäen etsittiin sille sopiva tekoälykirjasto, jota oli tarkoitus hyödyntää kehitysalustan tekoälyominaisuuksien testausta kunnonvalvontasovellusten kehitystä varten.

Opinnäytetyön tuloksena saatiin opetusmateriaaliksi tarkoitettu teoriapohja aiheista tekoäly, koneoppiminen, konenäkö, ennakoiva vikadiagnostiikka ja teollisuuden älykäs kunnonvalvonta sekä Industry 4.0. Opinnäytetyön toisena tuotoksena saatiin opas STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B-kehitysalustan ominaisuuksista, käyttöönotosta sekä laitteen koneoppimisytimen käytöstä.

Opinnäytetyön tulokset auttavat kouluttamaan asiantuntijoita tekoälyn hyödyntämiseen ennakoivassa vikadiagnostiikassa, teollisuuden älykkäässä kunnonvalvonnassa ja Industry 4.0-sovellusten kehityksessä. Opinnäytetyön tarkoitus on toimia opetusmateriaalina Kajaanin Ammattikorkeakoululla opinnäytetyön keskeisiin aihepiireihin liittyen.

Abstract

Author(s): Jaakkola Tanya

Title of the Publication: Artificial Intelligence as Part of Industrial Intelligent Condition Monitoring and Proactive Fault Diagnosis

Degree Title: Bachelor of Engineering, Information and Communication Technologies

Keywords: artificial intelligence, machine learning, proactive fault diagnosis, industrial intelligent condition monitoring, Industry 4.0, STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B, computer vision

The thesis was commissioned by Kajaani University of Applied Sciences. Key topics for the thesis were artificial intelligence, machine learning, computer vision, proactive fault diagnosis, industrial intelligent condition monitoring and Industry 4.0. One of the goals of the thesis was to create a comprehensive theoretical basis of said key topics to be used for educational purposes. The other goal of the thesis was to become familiar with the STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B development platform, its properties, its machine learning core and to test its functionality and compile instructions on using the development platform.

The background materials for the theoretical part of the thesis were collected mainly from online sources comprising on studies and articles on the key topics of the thesis. The development platform and its machine learning core was tested by collecting data from its environmental sensors and using the collected data as contextual data to find a suitable machine learning library to train the machine learning core to use for fault detection applications.

The end result of the thesis was material that is intended to be used as educational material in the future. The material was compiled from the key topics of the thesis. The other end result of the thesis was instructions on how to utilize the features and properties of the STEVAL-STWINKT1B development board.

The end results of the thesis can help to train experts in the use of artificial intelligence in proactive fault diagnosis, industrial intelligent condition monitoring and the development of Industry 4.0 applications. The purpose of the thesis is to serve as teaching material at Kajaani University of Applied Sciences in connection with the key topics of the thesis.

Sisällys

1 Johdanto 1

2 Tekoäly, koneoppiminen ja konenäkö 3

2.1 Tekoäly 3

2.1.1 Tekoälyn kolme aaltoa 3

2.1.2 Heikko ja vahva tekoäly 4

2.1.3 Keinotekoiset neuroverkot ja syväoppiminen 5

2.2 Koneoppiminen 10

2.3 Konenäkö 12

3 Industry 4.0 ja ennakoiva vikadiagnostiikka 13

3.1 Ennakoiva vikadiagnostiikka ja älykäs kunnonvalvonta 13

3.2 Industry 4.0 17

4 Opinnäytetyössä käytettävät ohjelmistot 20

4.1 NanoEdge AI Studio by Cartesiam 20

4.2 STM32 ohjelmistot 22

4.2.1 STM32CubeProgrammer 22

4.2.2 STM32CubeIDE 22

4.2.3 STM32 ST-LINK utility 23

5 STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B 24

5.1 STEVAL-STWINKT1B -kehitysalusta 24

5.1.1 STEVAL-STWINKT1B -kehitysalustapaketin osat 24

5.1.2 Mikrokontrolleri 27

5.1.3 Virranhallinta 28

5.1.4 Anturit 29

5.2 STEVAL-STWINKT1B -laiteohjelmistopaketti 31

5.2.1 Serial_DataLog -esimerkkiprojekti 34

5.2.2 BLE_SampleApp -esimerkkiprojekti 37

6 STEVAL SensorTile Wireless Industrial Node ja FP-AI-NANOEDG1 45

6.1 Työssä käytetyt laitteistot ja ohjelmistot 47

6.2 Työvaiheet 49

6.2.1 Micro-SD -kortin formatointi 49

6.2.2 Anturidatan keräys ja käsittely 50

6.2.3 Tekoälykirjaston hakeminen 55
6.2.4 Projektin kääntäminen 61
6.2.5 Tekoälyn koulutus ja testaus 65
6.3 Tulosten käsittely 69

7 Pohdinta 73

Lähteet 78

1 Johdanto

Vuosikymmenien ajan teollisuuden kunnonvalvonnassa on hyödynnetty ennaltaehkäisevää huoltoa. Tällöin laitteet on huollettu tietyin väliajoin ja tietyillä menetelmillä riippumatta siitä, onko huollettavissa laitteissa vikaa, eikä huollon ulkoisia osia ole välttämättä tarkastettu. Tämä voi paikoitellen olla tehoton menetelmä suorittaa huoltoa, sillä huollon onnistuminen on tarkastettu lähinnä silmämääräisesti. Tällä tavoin ei voida olla täysin varmoja, onko huolto onnistunut, kuinka tarpeellinen huolto on ylipäätään ollut tai onko vikoja jäänyt huollon aikana huomaamatta. Ennaltaehkäisevän huollon ongelma onkin ollut huoltojen suuret toimintakustannukset, sillä ylimääräisiä huoltoja ollaan voitu joutua tekemään laitteen vioituttua huollosta huolimatta. [1, s. 2.]

Hyödyntämällä ennakoivaa vikadiagnostiikkaa ja teollisuuden älykästä kunnonvalvontaa voitaisiin pienentää huoltotöihin meneviä kustannuksia ja tehostaa tuotantoa. Näillä menetelmillä hyödynnettäisiin laitteistoon sulautettuja ympäristöantureita, joilla kerättäisiin arvokasta dataa laitteiden toiminnasta. Tämän datan avulla voidaan kehittää tekoälyn avulla ennakoivia malleja, jotka ilmoittavat kehittymässä olevasta viasta. Kun vika havaitaan hyvissä ajoin etukäteen, voidaan huolto suunnitella niin, että häiriötä tuotantoon tulee mahdollisimman vähän eikä mahdollisesti ympäristöä saastuttavia ja resursseja haaskaavia onnettomuuksia pääse tapahtumaan. Tällä tavoin saataisiin myös hyödynnettyä rajallisia resurssejamme tehokkaammin niin, että mahdollisimman vähän menee hukkaan. [1, s 1-6. ; 2.]

On erityisen tärkeää saada koulutettua asiantuntijoita hyödyntämään ennakoivaa vikadiagnostiikkaa, teollisuuden älykästä kunnonvalvontaa ja tekoälyä teollisten kunnonvalvontasovellusten kehityksessä. Tällöin voitaisiin optimoida paremmin teollista tuotantoa vastaamaan sekä asiakkaan että tuottajan tarpeita ja vähentää ympäristön kuormitusta, jo koulutuksen aikana.

Tämä opinnäytetyö on tehty Kajaanin Ammattikorkeakoulun tilauksesta. Opinnäytetyölle keskeisiä käsitteitä ovat tekoäly, koneoppiminen, konenäkö, Industry 4.0, ennakoiva vikadiagnostiikka ja teollisuuden älykäs kunnonvalvonta. Opinnäytetyön tarkoituksena on laatia kattava teoriapohja näille käsitteille, jota voitaisiin jatkossa hyödyntää koulutusmateriaalina.

Toinen opinnäytetyön keskeisistä teemoista on tutustua STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B-kehitysalustaan sekä esitellä sen ominaisuuksia. Tavoitteena on luoda käyttöohjeet kehitysalustan käyttöönottoa varten, esitellä kehitysalustan sulautettujen antureiden ominaisuuksia sekä laatia laajemmat käyttöohjeet laitteen koneoppimisytimen käyttöä varten.

Opinnäytetyö suoritettiin etänä syksyn 2021 sekä kevään 2022 aikana. Työ tehtiin Windows 10 käyttöjärjestelmäisellä koneella. Opinnäytetyön kappaleessa 2 käydään läpi tekoälyyn, koneoppimiseen sekä konenäköön liittyvä teoria ja vastataan kysymyksiin, kuten mitä tekoäly on, miten se toimii, minkä tyyppistä tekoälyä on olemassa ja mikä ero on tekoälyllä ja koneoppimisella. Kappaleessa 3 tutustutaan teollisen maailman käsityksiin, kuten ennakoiva vikadiagnostiikka, Industry 4.0 ja teollisuuden älykäs kunnonvalvonta sekä niiden merkitykseen alati kehittyvässä teollisuudessa ja teknologiassa.

Kappaleessa 4 esitetään opinnäytetyössä käytettävät ohjelmistot, jotka ovat oleellisia STEVAL-STWINKT1B -kehitysalustan käytölle. Kappaleessa 5 esitellään STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B -kehitysalusta, sen sulautetut anturit ja muut ominaisuudet. Kappaleessa 5 myös käydään yksinkertaisten laiteohjelmistoesimerkkien avulla läpi, miten kehitysalustaan muodostetaan yhteys, millä tavoin siitä voidaan kerätä dataa ja millä tavoin data voidaan saada näkyville.

Kappaleessa 6 esitellään STEVAL-STWINKT1B -kehitysalustan koneoppimisytimen ominaisuuksia sekä käydään läpi vaihe vaiheelta, kuinka koneoppimisydin otetaan käyttöön, eli miten sitä varten kerätään dataa, miten kerättyä dataa hyödynnetään koneoppimiskirjaston etsimisessä, kuinka koneoppimismalli otetaan käyttöön kehitysalustalla, miten se koulutetaan ja kuinka kehitysalustaa tämän jälkeen voidaan käyttää ennakoimaan vikoja prosesseissa. Kappaleessa 6 käydään myös läpi kehitysalustan testauksen tulokset.

Kappaleessa 7 käydään läpi opinnäytetyön tuloksia, mikä meni hyvin ja mikä ei. Kappaleessa 7 pohditaan myös ehdotuksia jatkokehitystä varten, tehdään yhteenveto opinnäytetyössä käydyistä asioista ja mietitään ennakoivan vikadiagnostiikan ja teollisen älykkään kunnonvalvonnan tulevaisuutta sekä opinnäytetyön roolia tässä.

2 Tekoäly, koneoppiminen ja konenäkö

2.1 Tekoäly

Euroopan Parlamentin Tutkimuspalvelu on määritellyt tekoälyn seuraavalla tavalla:

”Tekoäly viittaa järjestelmiin, jotka osoittavat älykästä käyttäytymistä analysoimalla ympäristöään ja suorittamalla toimintoja – jossain määrin itsenäisesti – tiettyjen tavoitteiden saavuttamiseksi.” [3, s. 1]

Tekoäly on käsitteenä hyvin laaja ja sitä voidaan pitää eräänlaisena sateenvarjoterminä. Se kattaa allensa useita eri teknologioita ja käyttökohteita. Yhteistä näille teknologioille ja sovelluksille on niiden tulkinnanvarainen älykkyys. [3, s. 1] Teknologian saralla tekoälyllä tarkoitetaan koneen kykyä käyttää ihmisen älykkyyteen liittyviä taitoja, kuten oppiminen, suunnittelu sekä luominen [4.].

Tekoäly oppii asioita suoraan sille annetusta datasta. Se havaitsee rakenteita, kaavoja, piirteitä ja säännönmukaisuuksia sille annetusta tietomassasta, jonka pohjalta se luokittelee ja ennustaa. Oppiminen pohjautuu sille annettuun tekoälymalliin, joka mukautuu saadessaan lisää dataa. [5.]

Koska tekoäly oppii vain sille annetun datan perusteella, voi se keskittyä vain sille annetun opetusdatan mukaisiin tehtäviin. Tämä tarkoittaa esimerkiksi sitä, että puheentunnistustehtäviin koulutettu tekoäly mitä todennäköisimmin ei kykene ajamaan autoa itsenäisesti. [5.]

2.1.1 Tekoälyn kolme aaltoa

Boucherin mukaan [3.] tekoälyn kehitys voidaan ajatella kolmena aaltona. Ensimmäinen näistä aalloista oli ns. symbolisen tekoälyn kehitys. Siinä ihmiset määrittivät tekoälylle tarkkoihin sääntöihin perustuvat parametrit eli algoritmit. Tekoälyn tehtävänä oli tällöin vastata sille annettuihin kysymyksiin ja ongelmiin algoritmien perusteella. Tekoälyn kehityksessä symbolinen tekoäly oli ensiaskel. Symbolinen tekoäly toimii edelleen tilanteissa, joissa tarvitaan selkeät ja jyrkät säännöt. [3, s. 2-3.]

Symbolisen tekoälyn voidaan ajatella toimivan esimerkiksi seuraavalla tavalla: Tekoäly on opetettu erottelemaan puut karkeasti toisistaan joko lehti-tai havupuiksi. Mikäli puussa kasvaa havunneulasia, luokittelee tekoäly sen havupuuksi. Jos puussa kasvaa sen sijaan lehtiä, luokitellaan se lehtipuuksi. Jos tästä halutaan vielä kompleksisempi, voidaan tekoäly opettaa erottelemaan puulajeja toisistaan. Jos puussa on vaahteran lehtiä, kyseessä on vaahtera. Jos puussa on tammen lehtiä, kyseessä on tammi jne. Symbolinen tekoäly tarvitsee siis selkeät, ihmisen määrittelemät säännöt toimiakseen.

Tekoälyn toinen aalto on datapohjaisempi lähestymistapa symboliseen tekoälyyn verrattuna. Siinä tekoälylle annetaan dataa analysoitavaksi. Tekoäly tekee itsenäisesti päätelmiä datan sisällöstä ja ominaisuuksista, jonka perusteella se tekee päätelmiä sille esitetyistä kysymyksistä ja ongelmista. Tämä datapohjainen lähestymistapa on kehittynyt kiivasta tahtia viime vuosikymmenien aikana. Tällainen tekoäly ei tosin tarkoita sitä, että tekoäly varsinaisesti ymmärtäisi datan, jota se käsittelee, taikka sitä ympäröivää maailmaa. [3, s. 3-13.]

Tekoälyn kolmas aalto koskee tekoälyn tulevaisuutta. Sen ajatuksena on, että tekoäly kykenisi osoittamaan varsinaista älykkyyttä ymmärtämällä sille annetun tiedon ja miten tämä tieto on suhteessa ympäröivään maailmaan. Nykypäivänä kyseessä on vain spekuloiduista tekoälyn mahdollisuuksista, eikä se ole nykyteknologialla mahdollista. [3, s. 13-14.]

2.1.2 Heikko ja vahva tekoäly

Mäntylä sanoo artikkelissaan, että tekoäly voidaan luokitella älykkyytensä perusteella kahteen tyyppiin: heikkoon ja vahvaan tekoälyyn. [6.]

Heikko tekoäly saa pohjansa sille määritellyistä säännöistä eli algoritmeista. Se oppii ja tekee päätelmiä näiden sääntöjen perusteella. Heikko tekoäly kykenee analysoimaan sille annettua dataa löytäen piirteitä ja kaavoja siitä, mutta se ei varsinaisesti kykene ymmärtämään sille annettua dataa, missä viitekehyksessä itse tekoäly toimii tai ovatko sen tekemät päätökset oikeita tai järkeviä. Voidaan siis ajatella, että heikko tekoäly vain käyttäytyy älykkäästi olematta älykäs. [6.]

Vahvan ja heikon tekoälyn erona on se, että vahva tekoäly kykenee itsenäiseen ajatteluun. Tämä tarkoittaa sitä, että vahva tekoäly ymmärtää oppimansa asian sisällön, viitekehyksen missä se toimii sekä ympäröivää maailmaa. Tämänhetkisellä teknologialla ylletään vain heikkoon tekoälyyn. [6.]

2.1.3 Keinotekoiset neuroverkot ja syväoppiminen

Nicholsonin [7] mukaan keinotekoisilla neuroverkoilla (Artificial Neural Network, ANN) tarkoitetaan keinotekoista rakennetta, jonka toiminta perustuu löyhästi ihmisaivojen toimintaan ja rakenteeseen. Pain [8.] mukaan neuroverkot rakentuvat useista kerroksista keinotekoisia neuroneita eli perseptroneja. Keinotekoiset neuroverkot on suunniteltu tunnistamaan kaavoja, ominaisuuksia ja piirteitä niille syötetystä datasta hyödyntäen datan tulkintaan esimerkiksi luokittelua, ryhmittelyä ja regressiota. [8.]

Luokittelussa on tärkeää, että neuroverkolle syötetyssä datassa on leima eli label. Neuroverkko oppii korrelaation luokkien ja datan välillä nimenomaan leiman avulla. [7.]

Toisin kuin luokittelussa, datan ryhmittelyssä neuroverkko ei tarvitse etukäteen määriteltyä leimaa datalle. Ryhmittelyssä neuroverkko etsii itsenäisesti yhtäläisyyksiä, ominaisuuksia ja kaavamaisuuksia sille syötetystä datasta. [7.]

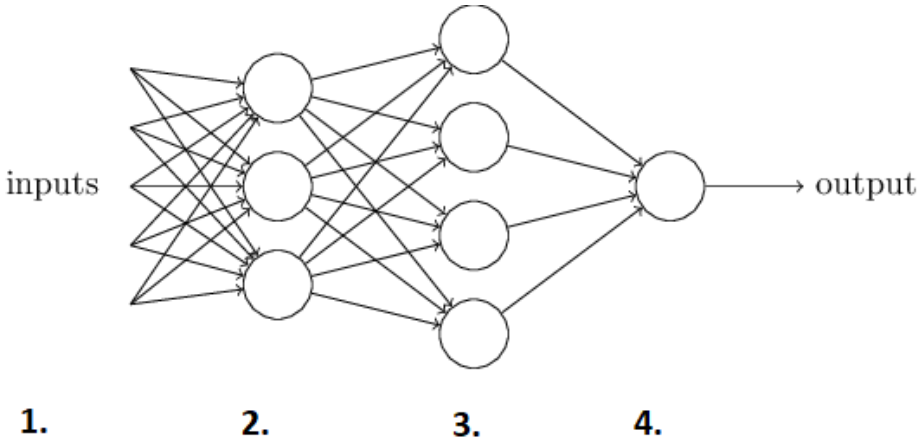
Regressio on menetelmä, jota käytetään, kun halutaan löytää tilastollisia vastaavuuksia menneiden, tämänhetkisten ja tulevaisuuden tapahtumien välillä eli kun halutaan menneisyyden perusteella tehdä ennusteita tulevaisuudesta. Esimerkiksi regression avulla voitaisiin ennustaa, milloin jokin koneen osa tulee hajoamaan samanlaisten koneiden aiemman toiminnan ja kunnon perusteella. [7.]

Nielsenin [9.] mukaan, keinotekoisen neuroverkon rakennuspalikat eli neuronit, toimivat niin, että ne ottavat vastaan useita binaarisia tuloja (x1, x2, x3, … ,xn) ja tuottavat yhden binaarisen lähdön. Lähdön laskemiseksi neuronille tarvitaan paino (w), joka on numeerinen arvo. Paino määrittää lähdön vastaavien tulojen arvot suhteessa lähtöön. Neuronin lähtö on aina joko 1 tai 0 ja se riippuu siitä, onko painotettu summa enemmän vaiko vähemmän kuin jokin määritetty

kynnysarvo. Myös kynnysarvo on numeerinen arvo, joka toimii yhtenä neuronin parametreistä. Edellä kuvattu on ilmaistu matemaattisesti yhtälössä 1. [9.]

$$Output = \begin{cases} 0, \ \sum_j w_j x_j \leq kynnysarvo \\ 1, \ \sum_j w_j x_j \geq kynnysarvo \end{cases} \quad (1)\ [9.]$$

Neuronin voidaan ajatella olevan kone, joka tekee päätöksiä analysoimalla sille annettua ”todistusaineistoa” eli dataa. Päätöksentekomalleihin vaikutetaan muuttamalla neuronille parametreina annettujen painojen ja kynnysarvojen numeerisia arvoja. [9.]



Kuva 1. Neuroverkon rakenne. [9, kuvaa muokattu.]

Kuvassa 1 on esitetty yhdenlainen malli neuroverkon rakenteesta. Ensimmäinen kerros toimii inputkerroksena, jota tietoa syötetään seuraavalle kerrokselle. Toisen kerroksen neuronit tekevät päätöksensä inputin sekä niille määritettyjen parametrien perusteella. Toisen kerroksen päätelmät ovat yksinkertaisia ja ne toimivat tuloina kolmannelle kerrokselle. Kolmas neuronikerros tekee monimutkaisempia päätöksiä toisen kerroksen lähtöjen ja neuroneille annettujen parametrien perusteella. Kolmannen kerroksen lähdöt toimivat tuloina neljännelle, eli viimeiselle, neuronikerrokselle. Viimeinen kerros tekee lopullisen päätöksen kyseiselle neuronille annettujen tulojen ja parametrien perusteella. Viimeinen kerros toimii siis outputkerroksena. Ensimmäisen ja viimeisen neuronikerroksen väliin jäävät neuronikerrokset toimivat piilokerroksina. Edellä kuvattu on esitetty matemaattisesti yhtälössä 2, jossa w ja x ovat vektoreita, joiden komponentteina ovat painot ja tulot. [9.] Käytännössä neuroverkoissa saattaa olla huomattavan paljon enemmän kerroksia, kuin vain kuvassa esitetyt neljä, lisäten neuroverkon kompleksisuutta.

$\sum_j w_j x_j \quad korvataan \quad w \cdot x \equiv \sum_j w_j x_j$ (2) [9.]

Seuraava muutos yhtälöön on siirtää kynnysarvo toiselle puolelle epätasa-arvoa ja korvata se neuronin bias-merkinnällä eli poikkeaman merkinnällä. Asia on esitetty yhtälössä 3.[9.]

$b \equiv -kynnysarvo$ (3) [9.]

Neuronin säännöt voidaan kirjoittaa yhtälössä 4 esitetyllä tavalla.

$$Output = \begin{cases} 0, & w \cdot x + b \leq kynnysarvo \\ 1, & w \cdot x + b \geq kynnysarvo \end{cases} \qquad (4)\ [9.]$$

Neuroverkon koulutuksen päätavoitteena on löytää painot ja bias-arvot, jotka minimoivat kustannusfunktion (quadratic cost function), joka on esitetty yhtälössä 5. [9.]

$C(w,b) \equiv \frac{1}{2n}\sum_x \|y(x) - a\|^2$, jossa $w = kaikki\ neuronin\ painot$ (5) [9.]

$b = kaikki\ neuroverkon\ poikkeamat$

$n = kaikki\ koulutustulot$

$a = vektori\ kaikista\ neuroverkon\ lähdöistä, kun\ tulo\ on\ x$

Kustannusfunktio on algoritmi, jonka avulla löydetään painot ja bias-arvot, joilla neuroverkon lähtö on likimäärin y(x) kaikille harjoitustuloille x. [9.]

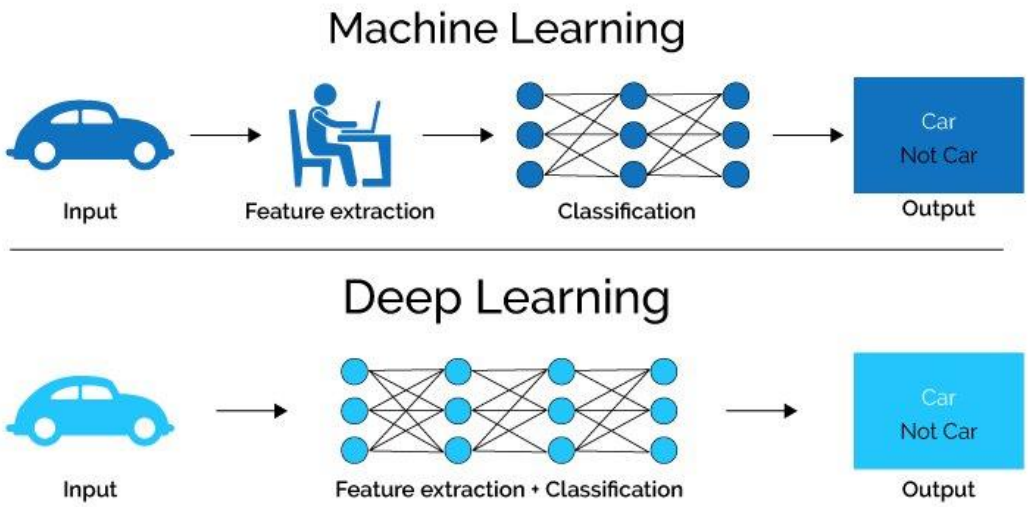
Dongesin [10] mukaan, tärkeä työkalu kustannusfunktion minimointiin on gradient descent optimointifunktio, jonka tarkoitus on löytää minimi differentiaaliyhtälöstä. Se mittaa muutoksen kaikissa neuroverkon painoarvoissa suhteessa bias:n muutokseen. Optimointifunktiota voidaan ajatella funktion kaltevuutena. Mitä jyrkempi funktion kaltevuus on, sitä nopeampaa neuroverkko oppii. Kun taas jos funktion kaltevuus on 0, neuroverkko lakkaa oppimasta. Matemaattisesti ajateltuna gradient descent on osittainen derivaatta suhteessa tuloihin. [10.]

Oppimisnopeus vaikuttaa oleellisesti siihen, kuinka nopeasti liikutaan optimaalisia painoarvoja kohti. Oppimisnopeus ei saa olla liian suuri, sillä optimointifunktion askeleista tulee liian pitkät, eikä tällöin välttämättä koskaan päädytä optimaalisiin arvoihin. Oppimisnopeus ei myöskään saa olla liian pieni. Liian pieni oppimisnopeus ei estä pääsemästä optimaalisiin painoarvoihin, mutta liian pieni oppimisnopeus hidastaa neuroverkon oppimista huomattavasti. [10.]

ANN, RNN ja CNN

Neuroverkkoja on olemassa useita erilaisia, joten tässä opinnäytetyössä keskitytään vain kolmeen yleisimpään neuroverkkotyyppiin: Artificial Neural Network (ANN), Recurrent Neural Network (RNN) sekä Convolutional Neural Network (CNN). Ne ovat syväoppimisen keinoja datan käsittelyyn.

Syväoppimisen ja koneoppimisen ero on esitetty yksinkertaisesti kuvassa 2. Kun koneoppimisessa tyypillisesti ihminen joutuu itse poimimaan piirteet datasta koneoppimismallin koulutusta varten, syväoppimisessa syväoppimismalli poimii piirteet sille syötetystä datasta itsenäisesti ja automaattisesti. [8.]



Kuva 2. Koneoppimisen ja syväoppimisen ero. [8.]

ANN voidaan ajatella olevan keinotekoisen neuroverkon perusmalli. Pain [8] mukaan se on tyypillisesti eteenpäin syöttävä neuroverkko, joka tarkoittaa, että neuroverkossa data liikkuu vain eteenpäin. ANN toimii niin, että sille annetaan aktivointifunktio, joka esittää neuroverkolle epälineaarisia ominaisuuksia. Aktivointifunktion avulla neuroverkko kykenee oppimaan minkä tahansa suhteen tulon ja lähdön välillä. ANN neuroverkoissa käsitellään tyypillisesti taulukkomuotoista, tekstimuotoista sekä kuvien muodossa olevaa informaatiota. Sitä on käytetty yksinkertaiseen luokitteluun, kasvojen tunnistukseen sekä konenäköön ja puheen tunnistukseen. [8; 11.]

RNN neuroverkot ovat takaisinpäin syöttäviä neuroverkkoja, jotka ottavat vastaan jaksollisen informaation sille syötetystä datasta jakaen parametreja eri ajanjaksoilla [7]. Nielsenin [9] RNN perusajatuksena on, että neuronit laukeavat vain tietyn ajanjakson ajan, jonka jälkeen ne siirtyvät lepotilaan. Neuronien laukeaminen voi stimuloida muita neuroneita laukeamaan myöhemmin, jotka vuorostaan voivat stimuloida muita neuroneita laukeamaan. Tästä voi seurata neuronien laukeamisen ryöppy. [9.]

RNN toimii syöttämällä informaatiota takaisinpäin neuroverkossa, jotta lopullinen lähtö voitaisiin ennustaa paremmin. Niissä ensimmäinen kerros on aina eteenpäin syöttävä kerros, piilokerrosten ollessa takaisinpäin syöttäviä kerroksi. Piilokerrokset muistavat muistifunktion avulla osan niille syötetystä informaatiosta. [11.]

Pain mukaan, koska RNN käyttää hyödykseen riippuvuussuhdetta datan jaksollisuudessa, voidaan niitä käyttää aikasarjadatan, tekstimuotoisen datan sekä audio-datan prosessointiin [8]. Takaisinpäin syöttäviä neuroverkkoja käytetään tekstin prosessointisovelluksissa, kuten kielioppivirheiden tarkastus, tekstistä puheeksi käännöksissä, kuvien tunnisteissa, käännösten tekemisessä sekä analysoimaan, onko esimerkiksi sähköpostiviestin sisältö sävyltään positiivinen vaiko negatiivinen. [11.]

CNN, eli konvoluutioneuroverkot, hyödyntävät konvoluutioita etsiessään piirteitä ja kaavoja sille syötetystä, pääasiassa kuvamuotoisesta, datasta. Konvoluutioneuroverkoissa on kolmiulotteinen asetelma neuroneita, joka poikkeaa ANN ja RNN neuroverkkojen kaksiulotteisesta asetelmasta neuroneita. Konvoluutioneuroverkot rakentuvat suodattimista. CNN ensimmäinen kerros on konvoluutiokerros, joka prosessoi informaatiota pieneltä alueelta sille annetusta datasta, esimerkiksi kuvasta. Konvoluutiokerroksessa CNN kerää kuvasta ominaisuuksia, jotka se vie suodattimelle. Yhdellä suodattimella käsitellään koko kuvan eri osa-alueet ominaisuuskartan muodostamista varten. CNN ymmärtää ominaisuuskartan avulla, missä järjestyksessä pikselit ovat suhteessa toisiinsa auttaen sitä tunnistamaan kohteita ja objekteja kuvassa, niiden sijainnit sekä suhteen muihin objekteihin. Ensiksi CNN tunnistaa kuvasta jyrkkiä reunoja ja yksinkertaisia muotoja. CNN täyttää lisää informaatiota tähän jokaisella ennustuskierroksella. Tyypillisesti konvoluutioneuroverkkoja on käytetty kuvien käsittelyyn, konenäkösovelluksiin, puheen tunnistukseen sekä konekäännöksiin. [8; 11.]

2.2 Koneoppiminen

Koneoppiminen on yksi tekoälyn osa-alueista, jonka periaatteena on, että systeemi voi oppia kerätystä datasta, tunnistaa kaavoja ja tehdä päätöksiä niin, että ihminen on panostanut koko prosessiin mahdollisimman vähän. Koneoppiminen poerustuu koneoppimisalgoritmeihin, jotka kouluttavat koneen oppimaan suoraan informaatiosta käyttäen laskennallisia keinoja. [13, s. 5; 12.]

Koneoppimisen päätavoitteena on päästä mahdollisimman lähelle syväoppimista, eli että ohjelmisto osaisi autonomisesti päätyä toivottuun tulokseen sille annetuissa ongelmissa mahdollisimman vähällä ihmisen panostuksella. Jotta päästäisiin tällaisiin tuloksiin, täytyy osata valita oikeanlainen koneoppimismalli, joka sopii koulutusdataa ja oppimistehtävää varten. Koneoppimismalleja on erilaisia ja ne voidaan jakaa tyypeiltään seuraavasti: ohjattu oppiminen, osittain ohjattu oppiminen, vahvistava oppiminen sekä ohjaamaton oppiminen. [13, s. 9; 12.]

Ohjatussa oppimisessa luodaan koneoppimismalli, joka tekee ennusteita olemassa olevan informaation perusteella. Tämä tarkoittaa sitä, että koneoppimisalgoritmille annetaan dataa, joka on jo valmiiksi luokiteltu ihmisen toimesta. Luokitellun datan avulla algoritmi oppii etsimään sille syötetystä datasta algoritmille opetettuja luokkia. Koneoppimisalgoritmi saa siis tiedossa olevan paketin dataa tulona sekä tiedossa olevat vasteet kyseiselle datalle. Algoritmi oppii vertailemaan omia tuloksiaan haluttuihin tuloksiin löytääksensä virheitä toiminnassaan. Ohjatussa oppimisessa käytetään luokittelua, kun halutaan ennustaa erilaisia vastauksia, ja regressiota, kun halutaan ennustaa jatkuvia vasteita. [13, s. 7.]

Osittain ohjattu oppiminen on hyvin samanlainen kuin ohjattu oppiminen ja sitä käytetään samantyyppisissä tapauksissa kuin ohjattua oppimista. Sen sijaan, että kaikki käytössä oleva koulutusdata olisi valmiiksi luokiteltua, osittain ohjatussa oppimisessa algoritmissa hyödynnetään myös luokittelematonta dataa. Tyypillisesti käytössä on pieni määrä luokiteltua dataa ja huomattavasti enemmän luokittelematonta dataa. [12.]

Vahvistava oppiminen toimii niin, että algoritmi etsii parhaita ratkaisuja yrityksen ja erehdyksen kautta. Sen tavoitteena on päätyä haluttuun tulokseen mahdollisimman vähillä yrityksillä. Vahvistavassa oppimisessa on toimija, joka tekee päätöksiä, ympäristö, jossa toimija tekee päätöksensä sekä toiminnot, eli päätökset, joita toimija kykenee tekemään. Toimija tekee siis

päätöksensä jossain tietyssä ennalta määritetyssä ympäristössä, johon on määritelty toiminnot, joita toimija kykenee suorittamaan. Vahvistavaa oppimista voidaan hyödyntää esimerkiksi robotiikkaan, navigaatioon tai peleihin liittyvissä tehtävissä. [12.]

Ohjaamattomassa oppimisessa koneoppimisalgoritmille ei anneta valmiiksi luokiteltua dataa eikä sille kerrota lopputuloksia, mihin sen halutaan päätyvän. Ohjaamattoman oppimisen päätavoitteena onkin tarkkailla dataa ja yrittää löytää siitä kaavamaisuuksia ja luonnollisia rakenteita. Ohjaamattomassa oppimisessa hyödynnetään usein datan ryhmittelyä tai klusterointia, kun halutaan etsiä kaavamaisuuksia, rakenteita tai datan ryhmittymistä. Koneoppimisen oppimismalleista ohjaamaton oppiminen on lähimpänä syväoppimista. [13, s. 8; 12.]

Kun halutaan valita oikea opetusmalli koneoppimisalgoritmin koulutukselle, valintaan ei valitettavasti ole yksiselitteisiä sääntöjä ja sopivan opetusmallin valinta voi vaatia usean yrityksen ennen kuin löydetään käytössä olevaan dataan sopiva malli. Opetusmallin valinnassa onkin oleellista tietää, millaista informaatiota ollaan käsittelemässä, mitä tietoa informaatiosta halutaan johtaa ja miten tätä informaatiota on tarkoitus hyödyntää jatkossa. Koneoppimisalgoritmin valintaa varten on myös tärkeää esikäsitellä käytettävä data, jotta se on sellaisessa muodossa, jota algoritmi kykenee tulkitsemaan. Koska datatyyppejä on monia erilaisia, kuten signaalimuotoinen data, kuvat, teksti ja numeerinen data, vaativat ne myös omanlaisensa esikäsittelyn. [13, s. 9-10.]

Koneoppimismallin koulutukseen liittyy monia eri työvaiheita. Aluksi tarvitaan riittävästi tarkoitukseen soveltuvaa dataa. Käytettävä data on myös esikäsiteltävä sellaiseen muotoon, joka on koneoppimisalgoritmille hyödynnettävässä muodossa. Saadusta datasta on johdettava ominaisuuksia. Esimerkiksi antureista saatavasta signaalimuotoisesta datasta voidaan johtaa ominaisuuksia huippuanalyysin, pulssi- ja siirtymämittausten sekä spektrin mittausten avulla. Koneoppimismalli koulutetaan johdettujen ominaisuuksien pohjalta. Jotta saadaan paras mahdollinen koneoppimismalli, koulutusta tulisi iteroida. Lopuksi parhaiten toimiva malli integroidaan haluttuun järjestelmään. [13, s. 16-29.]

2.3 Konenäkö

Konenäkö on yksi tekoälyn ja koneoppimisen sovelluskeino. Sen avulla tietokoneet ja systeemit voivat etsiä videoiden ja kuvien sisällöstä merkityksellistä informaatiota, kuten mitä objekteja kuvista ja videoista löytyy. Informaation perusteella kone tekee päätöksiä tai suosituksia. Konenäkö tuottaa siis kuviin ja videoihin pohjautuvaa tietoa. [12.]

Konenäön toiminnan kannalta on kaksi tärkeää teknologian osa-aluetta: syväoppiminen ja konvoluutioneuroverkot. Syväoppimista hyödynnetään konenäön koulutuksessa niin, että kokenäkömallille syötetään paljon visuaalista informaatiota sisältäen luokiteltua ja luokittelematonta dataa. Tietokone sitten ”katsoo” dataa oppiakseen erottamaan kuvat toisistaan ja tunnistamaan objekteja kuvista. Syväoppimisessa kone opettaa itseään oppimaan datasta. [12.]

Konvoluutioneuroverkkoja käytetään auttamaan konenäköä katsomaan kuvia. Tämä tapahtuu hajottamalla kuvat pikseleihin, joilla on jokin tag tai label. CNN sitten tekee konvoluutioita ja tarkistaa tulosten oikeellisuuden, kunnes ennustukset alkavat käydä toteen. Myös takaisinpäin syöttäviä neuroverkkoja voidaan käyttää, kun halutaan analysoida videokuvaa, sillä se auttaa konenäköä hahmottamaan, miten videon yksittäiset kuvat ovat liitoksissa toisiinsa. [12.]

3 Industry 4.0 ja ennakoiva vikadiagnostiikka

3.1 Ennakoiva vikadiagnostiikka ja älykäs kunnonvalvonta

Dr Bangert:n mukaan jokaiselle koneelle tai laitteelle käy sen elinkaaren aikana niin, että sen toiminta ei ole enää optimaalista. Tällä ei vielä tarkoiteta tilannetta, jossa koneen tai laitteen toiminta lakkaisi täysin, vaan tilannetta jolloin se ei toimi aivan kuten kuuluisi. Esimerkiksi koneen virrankulutus on voinut kasvaa sen normaaliin virrankulutukseen verrattuna, operointilämpötila ei välttämättä ole niissä rajoissa joissa sen kuuluisi olla tai koneessa voi operoinnin aikana ilmetä epätavallista tärinää. Tällaiset indikaattorit voivat kertoa koneeseen kehittyneestä tai kehittymässä olevasta viasta. Jotta koneiden kuntoa ja toimintaa saataisiin seurattua, kunnonvalvonta ja vikadiagnostiikka ovat oleellisia työkaluja tuotannossa. [2.]

Rhoads:n mukaan, perinteisesti kun on puhuttu kunnossapidosta, on tarkoitettu ehkäisevää huoltoa. Ehkäisevässä huollossa tiedetään tietyistä koneista niiden tyypillinen elinikä, koneen eri osien tyypilliset eliniät, odotettavissa olevat operaatiotunnit, miten koneen eri osat kuluvat ja kuinka usein ne tarvitsevat voitelua tai muita huoltotoimenpiteitä. Näiden tietojen perusteella ehkäisevässä kunnossapidossa on suunniteltu koneen huollot etukäteen riippumatta siitä, onko mitään varsinaista vikaa tai ongelmaa havaittu. [1.]

Ehkäisevällä kunnossapidolla on hyötynsä, kuten se, että koneistot tulee varmasti huollettua, mutta on sillä myös haittansakin. Suunnitelluihin ajantasaishuoltoihin menee tyypillisesti paljon aikaa ja henkilöstöresursseja. Jos koneistossa ei ole mitään varsinaista vikaa, ajantasaishuoltoon panostetut resurssit menevät hukkaan. Ajantasaishuollon aikana ei myöskään tule havaittua vikoja, jotka voivat piileä jossain muualla kuin sillä hetkellä huollettavassa koneen osassa. Perinteisillä kunnossapidon menetelmillä ei välttämättä kerätä anturidataa koneen toiminnasta ennen ja jälkeen huollon, jonka avulla voitaisiin tehdä päätelmiä siitä, kuinka hyödyllinen ja tehokas huolto on ollut. [1, s. 2-3.]

Puustisen [15] mukaan perinteisessä kunnossapidossa varsinaiset koneen toimintaa haittaavat viat on tyypillisesti korjattu vasta siinä vaiheessa, kun vika on jo ilmaantunut ja seisauttanut koko tuotannon. Salminen [16] kertoo artikkelissaan, että ehkäisevässä kunnossapidossa koneista

mahdollisesti kerättyä anturidataa on hyödynnetty vasta siinä vaiheessa, kun on tutkittu jo tapahtuneita asioita eikä ohjaamaan ennakoivaa kunnossapitoa.

Ennakoivassa vikadiagnostiikassa ja teollisuuden älykkäässä kunnonvalvonnassa pyritään siihen, että kone tai laitteisto kykenisi ilmoittamaan itsenäisesti siihen kehittyneestä tai kehittymässä olevasta viasta niin, että järjestelmä voitaisiin huoltaa hyvissä ajoin ennen kuin vika ehtii aiheuttaa vakavia ongelmia tuotannossa [16]. Mäen [17] mukaan tavoitteena on, että viat voitaisiin ennakoida hyvissä ajoin ennen kuin ne aiheuttavat ongelmia tai kalliita seisautuksia tuotannossa huoltotöiden takia. Seisautukset tuotannossa yhtäkkisesti ilmaantuneen vian tai ongelman takia voivat osoittautua pitkäkestoisiksi ja kalliiksi korjata, varsinkin jos vian aiheuttaja ei ole jo valmiiksi tiedossa. [17.]

Ennakoiva vikadiagnostiikka ja teollisuuden älykäs kunnonvalvonta perustuvat ennakoiviin malleihin, jotka voivat arvioida laitteen jäljellä olevaa käyttöikää ja huoltotarvetta havaittujen epänormaalien ilmiöiden perusteella. Näitä epänormaaleja ilmiöitä voivat olla esimerkiksi liian korkeaksi noussut operointilämpötila, epätavallinen tärinä koneistossa ja koneen noussut virrankulutus. [18, s. 2.]

Eräs tapa suorittaa kunnonvalvontaa teollisuudessa on seurata laitteeseen tai koneeseen kytkettyjä sulautettuja yksittäisiä antureita ja niiden tuottamaa dataa, esimerkiksi kosteus- tai lämpötila-anturin tuottamaa dataa. Yksittäisestä anturista kerätään dataa silloin kun kone on käynnissä suorittamassa normaaleja prosessejaan. Prosessin aikana kerätyn datan perusteella asetetaan ylä- ja alarajat sille, minkä ajatellaan olevan normaalia koneen toiminnassa. Tällä menetelmällä oletetaan, että anturien tuottamien signaalien ollessa näiden määriteltyjen raja-arvojen sisällä, kone toimii normaalisti. Mikäli esimerkiksi lämpötila-anturista saadut lukemat ylittävät määritetyt raja-arvot indikoiden liian korkeaa operointilämpötilaa, järjestelmä antaa ilmoituksen epänormaalista toiminnasta. [2.]

Ongelma edellä mainitulla menetelmällä on, että koneen toimiessa normaalisti, signaalit todennäköisesti käyvät hetkellisesti määriteltyjen raja-arvojen ulkopuolella riippuen prosessin vaiheesta, esimerkiksi prosessin käynnistyessä tai loppuessa. Tällaisessa järjestelmässä hetkellinen raja-arvojen ylitys tai alitus, aiheuttaisi väärän hälytyksen. Ongelmana on myös se, että seuraamalla vain yhden ympäristöanturin tuottamia signaaleja ei saada kokonaiskuvaa koko järjestelmän toiminnasta. Esimerkiksi jos seurataan vain lämpötila-anturin tuottamaa dataa, ei

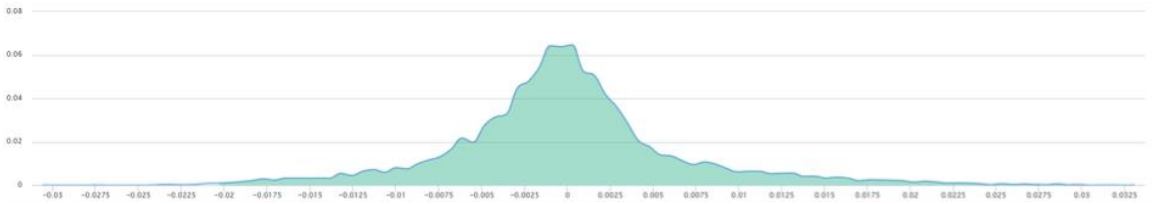
voida mitenkään saada tietoon jos kone alkaa tärisemään epätavallisella tavalla, jos virrankulutus yhtäkkisesti nousee poikkeavan korkeaksi tai jos ilmankosteus nousee epätavallisen korkeaksi. Tällöin, vaikka kone toimisikin lämpötila-anturin tuottamien signaalien mukaan normaalisti, ei välttämättä tule havaittua epätavallista toimintaa, joista muunlaisten anturien tuottamat signaalit voisivat kertoa. Voidaan siis sanoa, että yhden anturin tuottamista signaaleista ei voida saada kokonaiskuvaa koko järjestelmän terveydestä. Tehokkaan ennakoivan vikadiagnostiikan kannalta olisikin oleellista seurata useasta erilaisesta anturista kerättyjä signaaleja, jotta saataisiin parempi ymmärrys ja kokonaiskuva järjestelmän toiminnasta ja terveydestä. [2.]

Ennakoivassa vikadiagnostiikassa ja teollisuuden älykkäässä kunnossapidossa tärkeä työkalu on tekoäly. Tekoäly koulutetaan datalla, jota on kerätty, kun kone tai laite on ollut optimaalisessa toimintakunnossa. Dataa olisi kerättävä mahdollisimman paljon koneen normaalista toiminnasta, jotta tekoäly oppii, miten laitteen prosessit toimivat normaaleissa olosuhteissa. Tämä tarkoittaa sitä, että järjestelmän instrumenttien, toimilaitteiden ja säätimien on oltava optimaalisessa kunnossa silloin kun dataa on kerätty tekoälyn koulutusta varten. Antureilla prosessista kerätty data voi olla esimerkiksi informaatiota tärinästä, virrankulutuksesta, lämpötilasta ja ilmankosteudesta. [15; 16.]

Kunnossapidon näkökulmasta kyse on prosessimittausten ja kunnonvalvonta-antureiden keräämästä datasta, jolla tarkoitetaan myös eri laitteiden ohjaamiseen liittyvää tilatietoa, kuormitustietoa sekä toimintaympäristöön liittyvää tietoa [17]. Saadun datan pohjalta muodostetaan matemaattinen malli siitä, miten järjestelmä toimii optimaalisessa toimintakunnossa. Saatu koneoppimismalli edustaa sille annettujen parametrien suhdetta, jotka liittyvät laitteeseen ja sen toimintaan. [2.]

Saatua normaalin toiminnan koneoppimismallia käytetään suoraan etsimään vikoja laitteen tai koneen toiminnasta seuraamalla järjestelmästä otettavia jatkuvia mittauksia. Mikäli mittauksissa ilmenee normaalista toiminnasta poikkeavia tilanteita tai lukemia, ne indikoivat epänormaalia toimintaa, joka voi olla ensimmäinen merkki viasta, joka on kehittymässä järjestelmään. Laitteen normaalin toiminnan arvoja voidaan verrata jatkuvan mittauksen arvoihin missä tahansa vaiheessa. Koska odotettu anturien tuottama arvo on laskettu koneoppimismallin perusteella, tiedetään ennalta poikkeamien todennäköisyysjakauma, joka on esitetty kuvassa 3, eli kuinka todennäköisesti mittaus on tietyn verran loitolla odotuksista. Näiden odotusten perusteella voidaan päätellä kuinka terve laite tai kone on, tai sitä vastoin hyödyntää luottamusväliä, kun

päätellään onko anturin antama signaali jo liian kaukana normaalista. Mikäli arvo on liian kaukana luottamusvälistä, järjestelmä antaa hälytyksen. Hälytystä voidaan parantaa antamalla lisätietoa siitä, missä vika todennäköisesti on tai kuinka huonossa laite todennäköisesti on helpottaen huoltohenkilökunnan työtä. [2.]



Kuva 3. Poikkeamien todennäköisyysjakauma. [2]

Dr Bangertin mukaan hyödyntämällä älykästä kunnonvalvontaa ja ennakoivaa vikadiagnostiikkaa voidaan vähentää huoltotöihin tarvittavia resursseja. Hyödyntämällä tekoälyä kunnonvalvontajärjestelmässä voidaan vähentää kunnonvalvontajärjestelmän ylläpitoon tarvittavaa ihmisten panostusta jopa puolella. Tämä johtuu siitä, että parhaimmassa tapauksessa kunnonvalvontajärjestelmä oppii itse erilaisten antureiden signaalien perusteella, mikä on sen toiminnalle normaalia eikä normaalin toiminnan rajapintoja tarvitse asettaa manuaalisesti. Näin myös kunnonvalvontajärjestelmän antamien väärien hältysten määrä voi vähentyä huomattavasti, jopa 90%, joka puolestaan vähentää ihmisten tarvetta määrittää laitevikoja yli 60%:lla. [2.]

Kun älykäs kunnonvalvonta ja ennakoiva vikadiagnostiikka otetaan käyttöön tuotantojärjestelmässä, huoltotöistä tulee enemmän ennakoivaa, kun ennen se on ollut enemmän reaktiivista. Ennakoivan vikadiagnostiikan avulla kehittyvät laiteviat voidaan havaita hyvissä ajoin etukäteen, ennen kuin viat kykenevät aiheuttamaan suurempaa vahinkoa koneistossa. Huoltotyöt voidaan suunnitella niin, että ne häiritsevät tuotantoa mahdollisimman vähän. Ennakoivan vikadiagnostiikan ja älykkään kunnonvalvonnan avulla saadaan parannettua tuotannon tehokkuutta sekä laitteiden käytettävyyttä ja vähentää huoltotöihin meneviä kustannuksia. [2.]

3.2 Industry 4.0

Puhuttaessa termeistä Industry 4.0 tai Teollisuus 4.0 tarkoitetaan teollisuuden neljättä vallankumousta. Toisin kuin aiemmissa teollisissa vallankumouksissa, Industry 4.0:ssa keskitytään yhdistettävyyteen, automatioon, koneoppimiseen sekä reaaliaikaiseen dataan. Neljännen teollisen vallankumouksen tarkoitus onkin saada aikaan älykkäitä tehtaita ja älykästä teollisuutta. [19; 20.]

Teollisia vallankumouksia on toistaiseksi ollut neljä. Ensimmäisen teollisen vallankumouksen aikana siirryttiin käyttämään vesi- ja höyryvoimaa hyödyntäviä koneita tehostamaan tuotantoa. Toinen teollinen vallankumous alkoi 1900-luvun alussa, jolloin tehtaissa otettiin käyttöön teräs ja sähkö. Tällöin otettiin käyttöön myös tuotantolinjat ja massatuotanto. Kolmas teollinen vallankumous alkoi 1950-luvulla, jolloin tuotantoon otettiin mukaan olennaisena osana elektroniikka ja tietotekniikka. [19.]

Ominaista neljännelle teolliselle vallankumoukselle on lisääntyvä automaatio sekä älykkäiden tai fiksujen, koneiden ja tehtaiden hyödyntäminen, joka tähtää tuotteiden tuottavampaan ja tehokkaampaan tuotantoon [20].

Älykkääseen tehtaaseen kuuluu ajatus siitä, että tehtaan sisällä kerätään dataa sulautetuista antureista. Antureista kerättyä dataa analysoidaan, jotta voitaisiin tutkia trendejä tuotannossa, seurata tehtaan koneiston kuntoa, tunnistaa kaavamaisuuksia tuotannossa ja tehdä parempia ratkaisuja koko tuotantoprosessissa. Älykkäissä tehtaissa dataa voidaan käyttää myös muissa organisaation osissa, kuten logistiikassa, jotta voidaan saavuttaa suurempi ymmärryksen taso koko tuotantoketjusta. Älykkään tehtaan verkon arkkitehtuuri riippuu suoraan yhdistettävyydestä. Tehtaan koneistosta kerätty reaaliaikainen anturidata on oleellinen osa valvontaa ja laitteiston kunnossapitoa. [20.]

Industry 4.0:lla on kuusi erilaista teknologian osa-aluetta, jotka ajavat sitä eteenpäin. Näitä ovat IIoT, Cloud Computing, AI ja koneoppiminen, Edge Computing, Cybersecurity sekä Digital Twin. Näiden teknologioiden avulla on tarkoitus saada aikaiseksi älykkäitä tehtaita. [20.]

IioT, eli Industrial Internet of Things, on merkittävä osa älykkäitä tehtaita IIoT:tä on esimerkiksi se, että tehtaan koneistoon kytketään älykkäät anturit, joilla kerätään paljon arvokasta dataa

koneiden toiminnasta eri asteissa niiden elinkaarta. Älykkäiden antureiden avulla voidaan seurata esimerkiksi koneiston kuntoa, virrankulutusta ja huollon tarvetta, tehden älykkäästä kunnonvalvonnasta helpompaa. [20.]

Toimitusketju myös on integroitava osaksi älykkäitä tehtaita. Toimiakseen kunnolla toimitusketjun on oltava läpinäkyvää ja tehokasta. Kun tavarantoimittajien kanssa jaetaan jonkin verran tietoa tuotannosta, kuten tuotannossa tapahtuvat viiveet yms., voi tavarantoimittaja suunnitella toimitukset paremmin resurssien säästämiseksi. Lisäksi seuraamalla sääolosuhteita, kuljetuskumppanin ja jälleenmyyjän tietoja, tuottaja voi ennakoida milloin tuote olisi hyvä saada toimitukseen, jotta se olisi kuluttajan tai asiakkaan käytettävissä juuri oikeaan aikaan. [20.]

Älykkään tuotannon toiminta vaatii eri asioiden liitettävyyttä ja integrointia. Näitä ovat esimerkiksi tekniikka, tuotanto, toimitusketju, myynti ja jakelu. Pilvipalvelut ja Cloud Computing auttavat mahdollistamaan näiden asioiden integroinnin osaksi tuotantoa. Hyödyntämällä Cloud Computing-ominaisuuksia eli pilvipalveluissa suoritettavia laskutoimituksia, voidaan tehokkaammin säilöä ja käsitellä tuotannon eri vaiheissa kertyvää dataa. [20.]

Tekoäly ja koneoppiminen mahdollistavat sen, että koko tuotantoketjusta kerätty valtava määrä dataa voidaan hyödyntää mahdollisimman tehokkaasti. Esimerkiksi koneistosta kerätystä datasta voidaan luoda koneoppimismalleja, joiden pohjalta tekoäly voi päätellä, milloin kone tarvitsee huoltoa, milloin se on hajoamassa ja onko tuotettavassa tuotteessa mahdollisesti jotain vikaa. Datan, ja sen pohjalta luotujen koneoppimismallien, perusteella voidaan suorittaa teollisuuden älykästä kunnonvalvontaa ja ennakoivaa vikadiagnostiikkaa, jolloin voidaan pidentää laitteiston ja järjestelmän elinikää sekä vähentää seisauksia tuotannossa yllätyshuoltojen takia. [20.]

Edge Computing eli reunalaskenta, tarkoittaa laskennallisten toimitusten suorittamista itse dataa tuottavan lähteen luona. Esimerkiksi tämä voi tarkoittaa jonkin koneen yhteydessä olevassa mittalaitteessa suoritettua laskentaa, joka on riippumaton siitä, onko mittalaite kytketty verkkoon tai yhteydessä pilvipalveluihin. Reunalaskenta on tärkeää tilanteissa, jolloin tuotannossa täytyy puuttua esimerkiksi turvallisuusriskiin tai mahdolliseen vikaan tuotteessa mahdollisimman pian, ennen kuin kalliita vahinkoja on päässyt tapahtumaan. Reunalaskenta tapahtuu datan tuottavan anturin tai mittalaitteen läheisyydessä vähentäen viivettä tarvittavassa reaktiossa, jos kone täytyy pysäyttää vian korjaamista varten. Data ei tällöin myöskään lähde koneistosta esimerkiksi pilvipalveluun, vähentäen riskiä tiedon vuotamisesta vääriin käsiin. [20.]

Cybersecurity eli tietoturva, on tärkeää älykäissä tehtaissa ja Industry 4.0:ssa. Sama yhdistettävyys, jota älykkäät tehtaat ja niiden koneisto tarvitsee tuotannon optimointia varten, tarjoaa myös uusia tuloreittejä hyökkäyksille ja haittaohjelmille. On siis tärkeää pitää tietoturva osana Industry 4.0 tehtaiden ja sovellusten kehityksessä. [20.]

Digital Twin on Industry 4.0 ominaisuus, joka mahdollistaa sen, että tuottajat voivat tehdä virtuaalisen kopion heidän tehtaistaan, prosesseistaan, tuotantolinjasta ja jopa toimitusketjusta. Digital Twin luodaan tehtaasta ja tuotannosta kerätyn datan perusteella. Se mahdollistaa sen, että tuottajat voivat simuloida esimerkiksi uuden tuotteen lisäämisen tuotantoprosessiin ennen kuin tehdään kalliita muutoksia itse varsinaisen tehtaan koneistossa. Näin voidaan testata, miten uuden tuotteen tuotanto saadaan optimoitua sekä minimoitua viat ja virheet tuotanossa. [20.]

Simuloinnin, uusien teknologioiden ja materiaalien avulla älykkäissä tehtaissa voidaan helposti tuottaa yksilöityjä tuotteita tai tuote-eriä asiakkaiden tarpeiden mukaan [20].

Toisin kuin edellisissä teollisissa vallankumouksissa, Industry 4.0:ssa saadaan aikaiseksi se, että asiakkaiden tarpeet voidaan ottaa paremmin ja yksilöllisemmmin huomioon. Hyödyntäen uusia teknologoioita, kuten esimerkiksi 3D-tulostus, voidaan tehdä pieneriä jotain tuotetta, joka on yksilöity tiettyä asiakasta varten. Verrattuna edellisten teollisten vallankumousten massatuotantoon, Industry 4.0 mahdollistaa massaräätälöinnin asiakkaiden yksilöllisiin tarpeisiin. [20.]

4 Opinnäytetyössä käytettävät ohjelmistot

Tässä kappaleessa esitellään opinnäytetyössä käytettyjä ohjelmistoja. Esiteltävät ohjelmistot liittyvät suoraan käytettävään kehitysalustaan ja ovat täten joko suoraan STMicroelectronics omia ohjelmistoja tai työkaluja tai niihin liittyvien yritysten työkaluja, kuten Cartesiam NanoEdge AI Studio.

4.1 NanoEdge AI Studio by Cartesiam

NanoEdge AI Library, ja sen lisäksi NanoEdge AI Studio, ovat Cartesiam-yrityksen kehittämiä. NanoEdge AI Library on staattinen tekoälykirjasto sulautettuja C-kielen ohjelmia varten, jotka toimivat Arm Cortex M-sarjan mikrokontrollereissa. Se hyödyntää tekoälyä luodakseen datamalleja vastaanottamistaan signaaleista, jotka tulevat suoraan käytettävän mikrokontrollerin tai kehitysalustan sisäänrakennetuista antureista. Antureita voivat olla esimerkiksi kosteus- ja lämpötila-anturit sekä muut ympäristöanturit. NanoEdge AI kirjastojen avulla paikallinen tekoälyn koulutus ja analyysi voidaan integroida C-koodiin. Kirjasto toimii autonomisesti mikrokontrollerissa sekä koulutusvaiheen että jatkuvan analyysin aikana ja se tulee esikäännetyn .a-tiedoston muodossa tarjoten rakennuspalikat älykkäiden ominaisuuksien implementointiin C-kielessä. [21.]

Kaikki NanoEdge AI-kirjastot ovat optimoitu toimimaan millä tahansa ARM Cortex-M sarjan mikrokontrollerilla. Ne ovat erittäin tehokkaita muistin käytön suhteen (1-20 Kb RAM/Flash), niillä on nopea päättelykyky (1-20 ms päättelynopeus, M4 80 MHz) ja ne toimivat suoraan mikrokontrollerin sisällä. Kaikki luodut tekoälykirjastot voidaan integroida suoraan olemassaolevaan koodiin tai laitteistoon ja niillä on hyvin pieni virrankulutus. Kaikki NanoEdge AI kirjastot ovat itsenäisiä pilvipalveluista lisäten niiden tietoturvaa. [21.]

Jotta NanoEdge AI kirjasto saadaan luotua, tarvitaan NanoEdge AI Studio niminen ohjelma. NanoEdge AI Studio toimii hakukoneena tekoälykirjastoille. Sen tarkoitus on löytää paras mahdollinen NanoEdge AI kirjasto haluttua kehitysalustaa varten käyttäen hakukoneen parametreina muunmuassa antureista kerättyjä signaaleja sekä itse kohdealustan ja sen mikrokontrolleria. [21.]

Kun NanoEdge AI Studion hakukoneesta halutaan löytää käyttötarkoitukseen sopiva tekoälykirjasto, tarvitaan hakukoneeseen seuraavat parametrit: mikrokontrollerin tyyppi, muistivaatimukset, käytettävän anturin tyyppi sekä esimerkkejä anturista saaduista signaaleista. Signaalit voivat olla käsittelemättömiä tai esikäsiteltyjä, mutta ne täytyy formatoida muotoon, jota hakukone voi tulkita. Tyypillisesti hakukoneeseen tarvitaan ainakin sellaisia signaaleja, jotka edustavat seurattavan koneen toimintaa silloin kun se on toiminut optimaalisesti. [21.]

NE AI Studiossa on tarjolla erityyppisiä tekoälykirjastoja: Anomaly Detection ja Classification. Anomaly Detection -kirjastoja käytetään epätavallisen tai poikkeavan toiminnan havaitsemiseen koneessa tai prosesseissa. Sitä käytetään kun käytettävällä alustalla, esimerkiksi STEVAL STWINKT1B, on käyty läpi koulutusvaihe, jossa tekoälymallille on opetettu kaikki toiminta, joka on seurattavan koneen käytökselle normaalia. Anomaly Detection -kirjastot ovat optimoituja, mutta ne täytyy kouluttaa sen jälkeen, kun ne on asennettu kohdealustaan. Ne keräävät dataa suoraan kohdealustan mikrokontrollerista ja oppivat käyttökohteessaan dynaamisesti ja autonomisesti. Poikkeavan toiminnan havaitseminen tapahtuu itse mikrokontrollerin sisällä. [21.]

Classification -kirjastoja käytetään erottelemaan ja tunnistamaan erilaisia käyttäytymismalleja datassa riippumatta siitä, ovatko havaitut käyttäytymismallit normaaleja vaiko epänormaaleja seurattavan laitteen toiminnassa sekä luokittelemaan havaitut käyttäytymismallit ennalta määrättyihin kategorioihin. Luokittelukirjasto on etukäteen sekä optimoitu että koulutettu ja sisältää staattisen tietokannan. Itse signaalien luokittelu tapahtuu mikrokontrollerin sisällä. [21.]

NanoEdge AI Studiosta saadun tekoälykirjaston mukana ei tule sellaisenaan käyttökelpoista C-kielen koodia. Kun tekoälykirjasto halutaan integroida haluttuun projektiin, tulee käyttäjän itse kirjoittaa tarvitsemansa C-kielen ohjelma. Koodissa tulisi hyödyntää NanoEdge AI kirjaston smart-funktioita, joita ovat initialize, learn ja detect. Käyttäjän tulee itse kirjoittaa, kääntää ja ajaa ohjelma haluamallensa mikrokontrollerille. [21.]

Kun luotu tekoälykirjasto sulautetaan mikrokontrollerille, antaa se mikrokontrollerille kyvyn ymmärtää antureiden toimintamallit automaattisesti. Näin käyttäjä ei varsinaisesti tarvitse tietotaitoa matematiikasta, koneoppimisesta, tekoälystä taikka datatieteestä. Ainut tarvittu tietotaito on sulautetusta C-kielen ohjelmoinnista. Kirjasto sisältää tekoälymallin, joka on suunniteltu tuomaan koneoppimiskyvyn mille tahansa C-kielen ohjelmalle hyödyntäen yksinkertaisia funktiokutsuja, kuten learn, detect ja classify. [21.]

4.2 STM32 ohjelmistot

4.2.1 STM32CubeProgrammer

STM32CubeProgrammer on ohjelmistotyökalu, joka on tarkoitettu STM32-tuotteiden ohjelmointiin. Se tukee STMicroelectronics tuotteita, jotka pohjautuvat ARM Cortex-M mikrokontrollereihin. STM32CubeProgrammer toimii sekä komentorivin (CLI) että graafisen käyttöliittymän kautta. [22, s. 1.]

Työkalun avulla voidaan pyyhkiä ohjelmia, ohjelmoida, katsoa sekä verifioida mikrokontrollerin flash-muistia. Se tukee debug ja bootloader käyttöliittymiä, kuten ST-LINK debug probe, UART, USB DFU, I2C, SPI sekä CAN bootloader käyttöliittymiä. [22, s. 1.]

STM32CubeProgrammer toimii Linux, MacOS sekä Windows käyttöjärjestelmissä [22, s. 1].

4.2.2 STM32CubeIDE

STM32CubeIDE on C/C++ ohjelmointialusta, joka toimii Windows, Linux sekä MacOS-käyttöjärjestelmissä. Sillä voidaan konfiguroida oheislaitteet, luoda sekä kääntää koodia ja siinä on virheenkorjausominaisuudet. STM32CubeIDE on suunniteltu STM32-tuotteiden mikrokontrollereita ja mikroprosessoreita varten. [23, s. 2.]

STM32CubeIDE pohjautuu Eclipse-ohjelmointialustan viitekehykseen. Se integroi STM32-konfiguraatiot ja projektin luonti ominaisuudet STM32CubeMX-ohjelmasta ajan säästämiseksi kehityksessä ja asennuksessa. [23, s. 2.]

Kun STM32CubeIDE:ssä on valittu joko tyhjä STM32 mikrokontrolleri tai esikonfiguroitu mikrokontrolleri, IDE luo projektin ja siihen tarvittavan alustuskoodin. Halutessaan käyttäjä voi palata alustukseen ja konfigurointiin missä tahansa vaihetta kehitystä generoidakseen uudelleen alustuskoodin ilman, että uudelleen generointi vaikuttaisi käyttäjän itsensä kirjoittamaan koodiin. [23, s. 2.]

IDE:ssä on mahdollista analysoida stack ja build, jolloin käyttäjä saa tietoa projektin statuksesta ja muistivaatimuksista. [23, s. 2.]

4.2.3 STM32 ST-LINK utility

SMT32 ST-LINK utility, STSW-LINK004, on ohjelmistokäyttöliittymä, joka on tarkoitettu STM32-tuotteiden mikrokontrollereiden ohjelmointiin. Se on myöhemmin korvattu STM32CubeProgrammerilla. Sillä voidaan lukea, kirjoittaa sekä varmistaa laitteen muisti. [24, s. 1.]

ST-LINK utility sisältää laajan valikoiman työkaluja. Niiden avulla voidaan ohjelmoida STM32-tuotteiden sisäinen ja ulkoinen muisti, varmistaa ohjelmoinnin sisältö sekä automatisoida STM32-laitteiden ohjelmointia. [24, s. 1.]

ST-LINK utility toimii sekä graafisen käyttöliittymän että komentorivin kautta [24, s. 1].

5 STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B

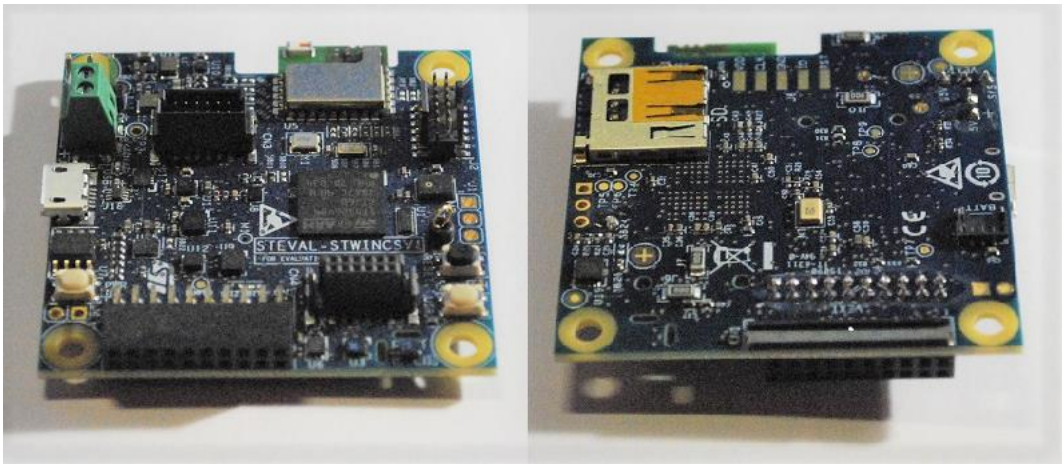
5.1 STEVAL-STWINKT1B -kehitysalusta

STEVAL SensorTile wireless industrial node, tai STEVAL-STWINKT1B, on kehitysalusta, joka on suunniteltu teollisten IoT-sovellusten kehitystä, suunnittelua ja testausta varten. Tällaisia teollisen IoT:n sovelluksia ovat esimerkiksi kunnonvalvonta ja ennakoiva huolto. [25, s. 1.]

Alustassa on sisäänrakennettuna teollisuuteen sopivia antureita, kuten kosteus- ja lämpötila-antureita, tärinäantureita, magnetometrejä ja mikrofoneja. STEVAL-STWINKT1B tukee langattomia BLE-yhteyksiä sisäänrakennetun BLE-ominaisuuden kautta. Langalliset yhteydet toimivat sisäänrakennetun RS485-vastaanottimen kautta. [25, s. 1.]

5.1.1 STEVAL-STWINKT1B -kehitysalustapaketin osat

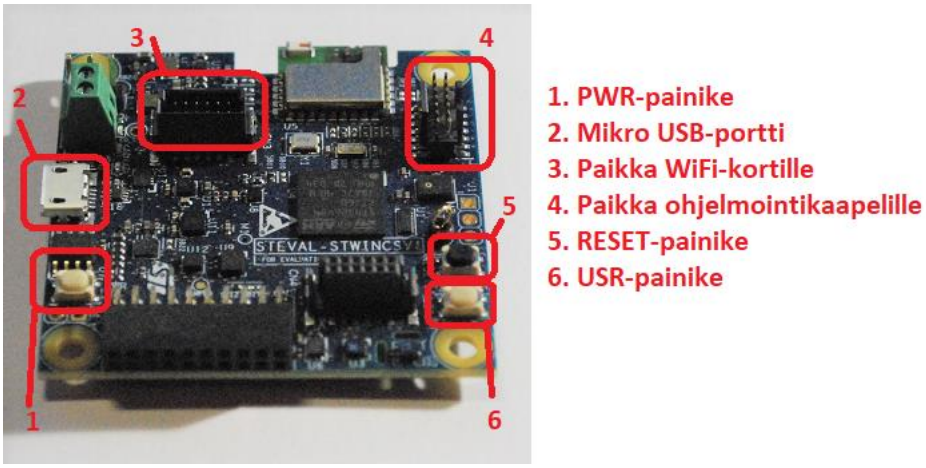
STEVAL-STWINKT1B kehitysalustapakettiin kuuluu itse STEVAL-STWINKT1B-ydinjärjestelmä eli kehitysalusta, suojaava muovikotelo, 480 mAh 3,7 V Li-Po-paristo, STLINK-V3MINI-debugger/programmeri sekä ohjelmointikaapeli. Toimiakseen kehitysalusta tarvitsee myös kaksi mikro USB-kaapelia, yksi kehitysalustaa varten ja toinen STLINK-V3MINI-debuggeria/programmeria varten. Kehitysalusta on esitetty kuvassa 4 ja Li-Po-paristo on esitetty kuvassa 5. [26, s. 2-3.]



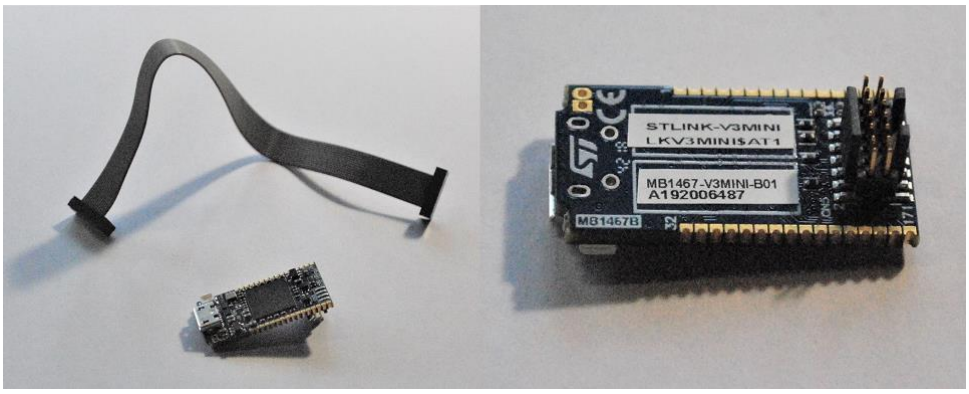
Kuva 4. STEVAL-STWINKT1B-kehitysalusta. Vas. kuvattuna yläpuolelta, oik. kuvattuna alapuolelta.

Kuva 5. 480 mAh, 3,7 V Li-Po-paristo.

Kuvassa 6 on esitetty alustan painikkeet ja liittimet. Kun kortti ei ole kytkettynä USB-kaapelin kautta tietokoneeseen, mutta kehitysalustaan on kytketty mukana tuleva Li-Po-paristo, PWR-painikkeella voidaan kontrolloida laitteen ON/OFF-tilaa. Kohtaan 3 voidaan kytkeä lisälaitteena erikseen hankittava STEVAL-STWINWFV1 WiFi-kortti. Ohjelmointikaapeli, jolla alusta ja STLINK-V3MINI kytketään sille tarkoitettuun paikkaan, on esitetty kohdassa 4. RESET-painikkeella voidaan resetoida alusta. USR-painikkeen avulla voidaan ohjelmoida laiteohjelmisto kehitysalustalle mikäli STLINK-V3MINI ei ole käytettävissä [26, s. 22].

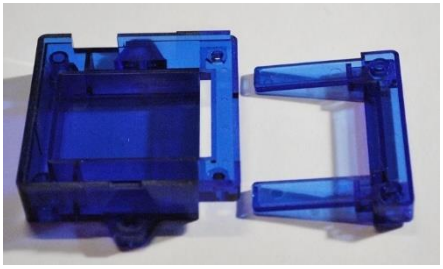


Kuva 6. STEVAL-STWINKT1B-kehitysalustan painikkeet ja liittimet.



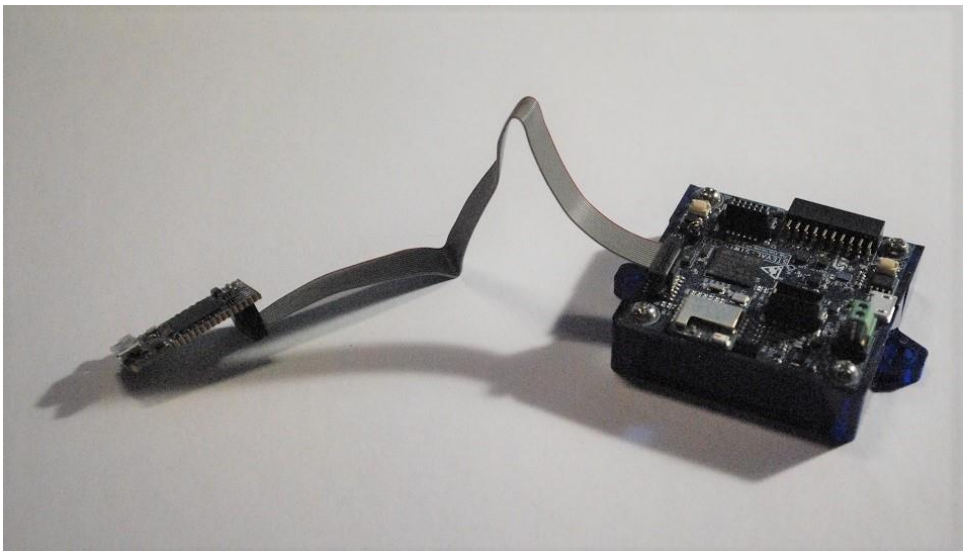
Kuva 7. STLINK-V3MINI sekä ohjelmointikaapeli.

Kuvassa 7 näkyy STLINK-V3MINI sekä ohjelmointikaapeli. STLINK-V3MINI on erillinen debuggeri/programmeri minikoetin, joka on suunniteltu STM32-tuoteperheen mikrokontrollereille. Sen avulla saadaan aikaiseksi Virtual COM Port käyttöliittymä isäntä-pc:lle, jotta isäntä-pc voi kommunikoida laitteen mikrokontrollerin kanssa UART kautta. Jotta STLINK saadaan yhdistettyä alustaan, tarvitaan ohjelmointikaapeli. Sekä STLINK-V3MINI että kehitysalusta täytyy yhdistää tietokoneeseen omilla micro-USB -kaapeleillaan. [26, s. 19.]



Kuva 8. STEVAL-STWINKT1B-kehitysalustan muovinen suojakotelo.

Kuvassa 8 näkyy kehitysalustan mukana tuleva muovinen kehikko. Sitä käytetään pitämään alusta ja paristo yhdessä paketissa niin, että liikkuvia osia on mahdollisimman vähän. Muovikehikon sisällä on tilaa magneeteille, joilla alusta voidaan kiinnittää metallista pintaa vasten, esimerkiksi jonkin mitattavan koneen kylkeen. Muovikehikko kiinnitetään alustaan paketin mukana tulevilla muttereilla. [26, s. 19.]



Kuva 9. STEVAL-STWINKT1B-kehitysalusta kasattuna.

Kuvassa 9 näkyy STEVAL-STWINKT1B-kehitysalusta kasattuna. Li-Po-paristo on kiinnitettynä kehitysalustan alapuolelle ja kehitysalusta on kiinnitetty muovikehikkoon. STLINK-V3MINI on kytketty kehitysalustaan ohjelmointikaapelilla.

5.1.2 Mikrokontrolleri

STEVAL-STWINKT1B-kehitysalustan mikrokontrolleri, STM32L4R9Z1, on matalavirtainen ja sen rakenne perustuu Arm Cortex M4 32-bittiseen RISC-ytimeen, joka toimii 120 MHz:n taajuuteen asti. [26, s. 7.]

Kehitysalustassa on sulautettuna high-speed muisteja, 2 Mb Flash-muistia sekä 640 Kb SRAM, joustava ulkoisen muistin ohjain (FSMC) staattista muistia varten, kaksi OctoSPI Flash-muistin käyttöliittymää sekä laaja valikoima I/O-äärilaitteita yhdistettynä kahteen APB bus, kahteen AHB bus sekä 32-bittiseen multi-AHB bus matriisiin. [26, s. 9-10.]

Kehitysalustassa on myös nopea 12-bittinen ADC, kaksi komparaattoria, operaatiovahvistinta, sekä DAC-kanavaa, sisäisen jännitteen referenssibufferi, kaksi 16-bittistä ajastinta moottorien hallintaan, seitsemän 16-bittistä ajastinta sekä kaksi matalavirtaista 16-bittistä ajastinta. Kehitysalusta tukee neljää digitaalista filtteriä DFSDM:aa varten ja siinä on 24 kapasitiivistä tunnistuskanavaa. [26, s. 9-10.]

Laitteen operointilämpötila on -40°C … +85°C alueella. Käytettäessä laitteen sisäistä LDO-säädintä käyttöjännite on 1,71 V ... 3,6 V alueella ja ulkoista SMPS-jännitelähdettä käytettäessä käyttöjännite on 1,05 V ... 1,32 V alueella. [26, s. 9-10.]

Kehitysalustan mikrokontrollerissa on tarjolla 24 kapasitiivista kanavaa antureita varten ja useita eri viestintäkanavia. Siinä on neljä I2C-väylää, kolme SPI-väylää, kolme USART-, kaksi UART- ja yksi matalavirtainen UART-kanavaa, kaksi SAI-kanavaa, yksi SDMMC, yksi CAN, yksi USB OTG fullspeed-kanava, käyttöliittymä kanavaa varten sekä DMA2D -kontrolleri. [26, s. 9-10.]

Kehitysalustassa on sisäänrakennettuna SPBTLE-1S BLE-systeemi, joka tukee useaa eri BLE-roolia samanaikaisesti. Tämä tarkoittaa sitä, että laite kykenee toimimaan samanaikaisesti Bluetooth Smart Masterina sekä slave roolissa. BLE-moduuli perustuu Blue NRG-1:een ja BLE protokollapino on sulautettu moduuliin. Systeemissä on integroituna radio, sulautettu antenni sekä korkeataajuuksisia oskillaattoreita. Laitteen emittoima RF on +5 dBm ja RF-kanavan keskitaajuus on 2402~2480 MHz. [26, s. 10.]

5.1.3 Virranhallinta

STEVAL-STWINKT1B-kehitysalustassa on useita virranhallintaominaisuuksia, joiden avulla saadaan aikaiseksi hyvän matala virrankulutus lopullisessa käytettävässä sovelluksessa. Alustan pääasiallinen virranlähde on litium-ioni-paristo, 3,5 V ja 480 mAh, sekä integroitu laturi, STBC02, jonka jännitealue on 4,8 V ... 5,5 V. [26, s. 11-14.]

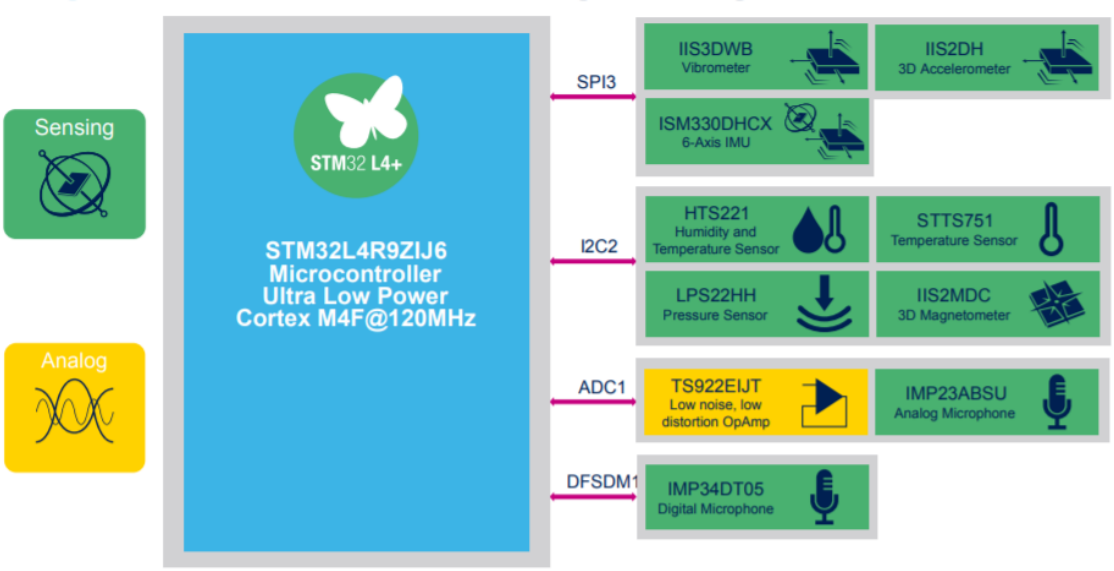
Kehitysalustaan voidaan ottaa virtaa useista eri lähteistä: 5 V jännitelähtö mikro USB-kaapelin kautta, 4,8 V ... 5,5 V J5-yhdistimen kautta, jolloin virta on rajattava 2 ampeeriin, sekä Li-Po-paristo, jonka jännite on 3,7 V ja virta 480 mAh. Paristoa ei ole pakko käyttää, sillä STBC02 -laturi tarkistaa aina saatavilla olevat jännitelähteet ja valitsee niistä yhden virtalähteeksi. Jos paristo on kytketty alustaan yhden muun virtalähteen lisäksi, STBC02 lataa paristoa. Kun paristo on käytössä, laite toimii 35°C operointilämpötilassa, muulloin operointilämpötila on 45°C. [26, s. 11-14.]

Kun Li-Po-paristo ei ole kytketty kehitysalustaan, laitteen ON/OFF-tila riippuu siitä, onko ulkoinen virtalähde kytkettynä vaiko ei. Li-Po-pariston ollessa kytkettynä laitteen ON/OFF-tilaa voidaan kontrolloida kuvassa 5 esitetyn PWR-painikkeen kautta. Painettaessa PWR-painiketta kahden

sekunnin ajan laite kytkeytyy päälle. Kun PWR-painiketta painetaan uudestaan, laite sammuu. [26, s. 11-14.]

5.1.4 Anturit

STEVAL-STWINKT1B-kehitysalustassa on useita sisäänrakennettuja antureita, joiden tarkoitus on tukea ja mahdollistaa Industry 4.0-sovelluksia. Anturit käyttävät kehitysalustan tarjoamia SPI-, I2C- sekä ADC-väyliä. Kuvassa 10 on esitetty kehitysalustan anturit sekä niiden käyttämät tiedonsiirtoväylät. [26, s. 4.]



Kuva 10. STEVAL-STWINKT1B-kehitysalustan anturit ja tunnistuselementit sekä niiden käyttämät tiedonsiirtoväylät. [26, s. 4]

HTS221 on analoginen kosteus- ja lämpötila-anturi, joka mittaa ympäristöstä lämpötilaa ja suhteellista kosteutta. Anturi toimii -40°C … +120°C lämpötila-alueella [26, s. 5; 27].

LPS22HH MEMS on pietsoresistiivinen paineanturi. Se mittaa absoluuttisen paineen ja toimii -40°C … +85°C lämpötila-alueella [26, s. 5; 28].

STTS751 on digitaalinen lämpötila-anturi. Lämpötila-anturin resoluutio voidaan asettaa käyttäjän toimesta. Valittavana oleva resoluutio on 9–12 bitin välillä. 10-bittinen resoluutio on anturin oletusasetus, jolloin anturin erottelukyky on 0,25°C ja nimellinen muuntamisaika on 21 ms. 9-

bittisellä resoluutiolla anturin erottelukyky on 0,5°C ja 12-bittisellä resoluutiolla erottelukyky on 0,0625°C. Anturin käyttämä 2-wire SMBus mahdollista sen, että yksi sovellus voi tarkkailla jopa kahdeksaa eri lämpötilavyöhykettä. [26, s. 6; 29.]

ISM330 DHCX on kolmiulotteinen kiihtyvyysanturi ja gyroskooppi, joka on suunniteltu Industry 4.0 sovelluksia varten. Kiihtyvyysanturin skaala on ±2/±4/±8/±16 g ja gyroskoopin skaala on ±125/±250/±500/±1000/±2000/±4000 astetta/sekunti. Anturi toimii -40°C … +105°C lämpötila-alueella ja sen käyttöjännite on 1,71 V ... 3,6 V alueella. Anturilla on 6-kanavainen synkronoitu lähtö, sisältää koneoppimisytimen ja siinä on sulautetut funktiot kallistuksen tunnistusta, herätystä, 6D/4D orientaatiota, klikkausta ja tuplaklikkausta varten. Anturissa on sulautettu lämpötila-anturi sekä pedometri askelten tunnistusta varten. [26, s. 6; 30.]

IIS3DWB on digitaalinen tärinäanturi. Sillä on ultraleveä kaistanleveys, se on matalakohinainen ja 3-akselinen. Anturin skaala on ±2/±4/±8/±16 g ja sillä on erittäin laaja ja tasainen taajuusvaste, 6 kHz. Anturi toimii -40°C … +105°C lämpötila-alueella ja sen käyttöjännite on 2,1 V ... 3,6 V alueella. Anturi soveltuu tärinän seurantaan teollisessa ympäristössä. [26, s. 6; 31.]

IIS2DH on 3-akselinen kiihtyvyysanturi, jolla on täysimittainen asteikko. Täysimittaisen asteikon skaala on ±2/±4/±8/±16 g. Anturi kykenee mittaamaan kiihtyvyyksiä lähtödatanopeuksilla, jotka ovat välillä 1 Hz … 5,3 kHz. Kiihtyvyysanturi voidaan konfiguroida tuottamaan keskeytyssignaaleja riippuen kahdesta itsenäisestä herätys/vapaapudotus-tapahtumasta sekä itse laitteen sijainnista. Anturi kykenee toimimaan -40°C … +85°C lämpötila-alueella. Anturin käytöjännite on 1,71 V … 3,6 V alueella ja sen virrankulutus on hyvin alhainen, 2 µA. Anturilla on useita käyttötiloja: low power, normal, korkearesoluutioinen suuntautumisen tunnistus joka auttaa laitetta tunnistamaan orientaatiotaan, liikkeen sekä vapaan pudotuksen tunnistus. [26, s. 6; 32.]

IIS2MDC on 3-akselinen magnetometri, jolla on kolme magneettikentän kanavaa sekä ±50 gaussin dynaaminen alue ja 16-bittinen ulostulo. Anturin käyttöjännite on 1,71 V … 3,6 V alueella ja toimii -40°C … +85°C lämpötila-alueella. Yksittäisen mittaustilan taajuus on enimmillään 150 Hz. Anturissa on ohjelmoitava interrupt-generaattori sekä sulautettu selftest ja lämpötila-anturi. Laite voidaan konfiguroida tuottamaan keskeytyssignaalin, mikäli magnetometri havaitsee magneettikentän. [26, s. 7; 33.]

IMP23ABSU on analoginen MEMS-mikrofoni. Se on matalavirtainen ja rakennettu kapasitiivisella tunnistavalla elemetillä, joka kykenee tunnistamaan akustisia aaltoja. Mikrofonilla on hyvin

korkea akustinen ylikuormituspiste, 130 dBSPL, sekä tyypillinen 64 dB:n signaalikohinasuhde. Mikrofoni toimii -40°C … +85°C lämpötila-alueella ja sillä on suuntaamaton herkkyys, korkea signaalikohinasuhde sekä erittäin tasainen taajuusvaste. Mikrofonin virrankulutus on maksimissaan 150 µA. [26, s. 7; 34.]

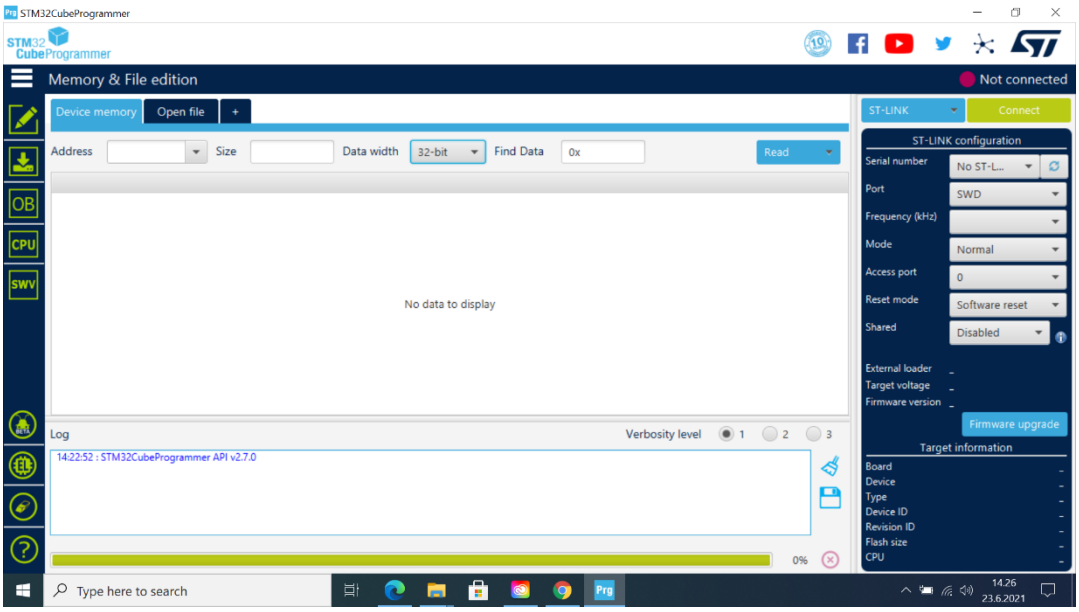
TS922-operaatiovahvistin on integroitu kehitysalustaan analogista mikrofonia varten. Operaatiovahvistin on matalakohinainen, 9 nV/√Hz, siinä on vähäinen vääristymä, matala poikkeama sekä korkea lähtövirta, 80 mA. Operaatiovahvistin toimii hyvin korkealaatuisissa, matalajännitteisissä tai paristokäyttöisissä äänijärjestelmissä ja toimii vakaasti jopa 500 pF:n kapasitiivisessa kuormassa. Operaatiovahvistimen käyttöjännite on 2,7 V ... 12 V alueella. [26, s. 6; 35.]

IMP34DTOS on digitaalinen MEMS-mikrofoni, joka on matalavirtainen ja suuntaamaton. Sillä on kapasitiivinen tunnistuselementti sekä IC-käyttöliittymä. Mikrofonissa on 64 dB:n signaalikohinasuhde sekä -26 dBFS ±3 dB herkkyys. Mikrofonin akustinen ylikuormituspiste on 122,5 dBSPL. [26, s. 7; 36.]

5.2 STEVAL-STWINKT1B -laiteohjelmistopaketti

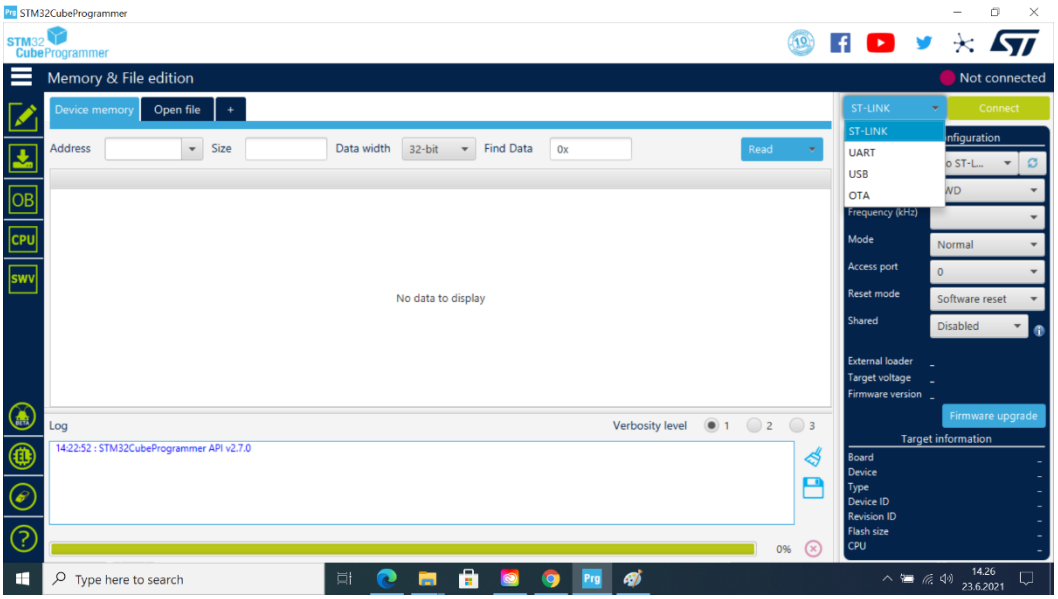
STEVAL-STWINKT1B-kehitysalustalle on ladattavissa ilmaiseksi firmware-paketti, eli laiteohjelmisto, joka sisältää esimerkkisovelluksia. Esimerkkisovellukset perustuvat STM32Cube-ohjelmistoon ja niissä tulee mukana tarvittavat ajurit, joilla voidaan hallita kaikkia kehitysalustan antureita ja ominaisuuksia. Tässä kappaleessa perehdytään kahteen paketissa tulevaan esimerkkiprojektiin: Serial_DataLog sekä BLE_SampleApp. [37, s. 1, 6-7.]

Esimerkkiprojektien suoritukseen tarvitaan tietokone Windows 10 -käyttöjärjestelmällä, kaksi micro-USB -kaapelia, STEVAL-STWINKT1B-kehitysalusta, STINK-V3MINI sekä 14-pinninen ohjelmointikaapeli. Esimerkkiprojektit on suunniteltu testattavaksi STM32CubeProgrammer ohjelmalla, jonka lisäksi tarvitaan TeraTerm. STEVAL-STWINKT1B ja STLINK-V3MINI yhdistetään toisiinsa kuvan 9 esittämällä tavalla, jonka jälkeen ne yhdistetään tietokoneeseen micro-USB -kaapeleilla. Kun koko paketti on yhdistetty tietokoneeseen, voidaan avata STM32CubeProgrammer. [37, s. 6.]



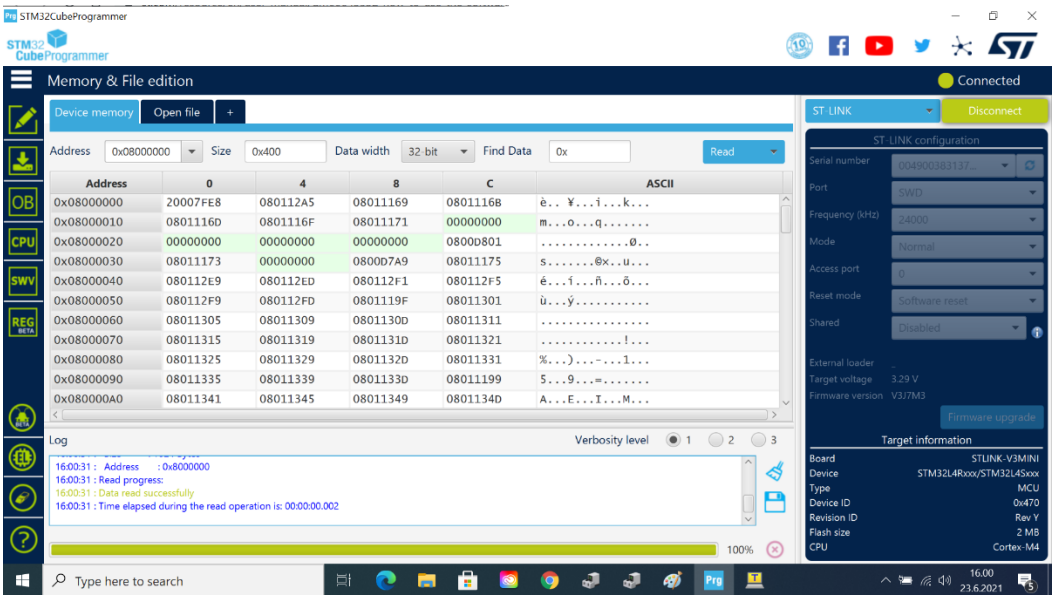
Kuva 11. STM32CubeProgrammer, käyttöliittymä.

Kuvassa 11 näkyy STM32CubeProgrammer käyttöliittymä. Yhteysvaihtoehdot ovat näkymän oikeassa reunassa. Sinisestä alasvetovalikosta valitaan, mitä kautta halutaan luoda yhteys kehitysalustaan. Vihreästä connect-painikkeesta muodostetaan yhteys.



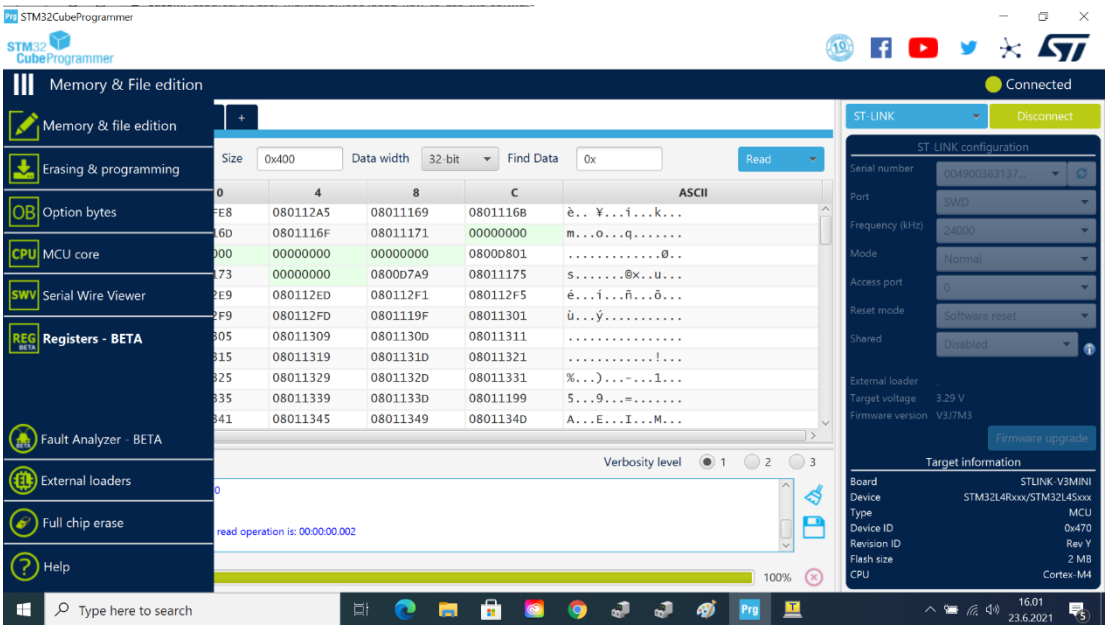
Kuva 12. STM32CubeProgrammer, alasvetovalikko.

Kuten kuvassa 12 näkyy, alasvetovalikossa on neljä eri vaihtoehtoa yhteyden muodostamiselle: ST-LINK, UART, USB sekä OTA. Esimerkkiprojekteja varten valitaan ST-LINK yhteysmuodoksi, jonka jälkeen painetaan connect-painiketta.



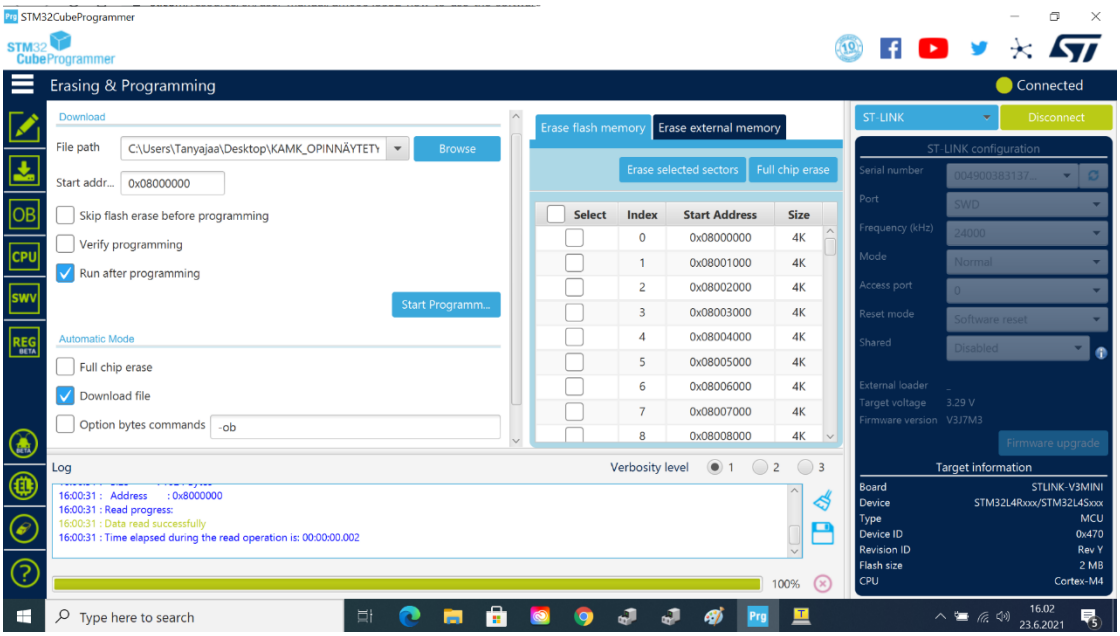
Kuva 13. Onnistunut yhteys.

Kuvassa 13 on käyttöliittymän näkymä, kun yhteys on muodostunut onnistuneesti. Tästä siirrytään kuvassa 14 näkyvään valikkoon, joka on näkymän vasemmassa reunassa.



Kuva 14. STM32CubeProgrammer, vasemman reunan valikko.

Valikossa on useita eri vaihtoehtoja. Firmware-esimerkkiprojekteissa tarvitaan kahta ensimmäistä: Memory & File Edition sekä Erasing & Programming. Kuvassa 14 näkyy Memory & File Edition näkymä. Erasing & Programming -näkymä on nähtävissä kuvassa 15.



Kuva 15. Erasing & Programming -näkymä.

Erasing & Programming -näkymässä tarvitaan Browse- ja Start Programming- toimintoja. Browse-toiminnon kautta päästään hakemaan projektiin tarvittava binaaritiedosto. Start Programming nimensä mukaisesti aloittaa ohjelmoinnin.

5.2.1 Serial_DataLog -esimerkkiprojekti

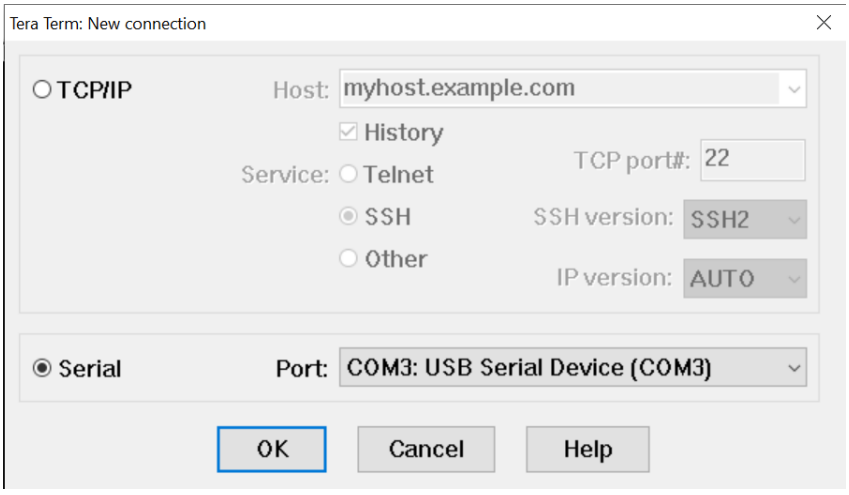
Serial_DataLog -esimerkkiprojekti on esimerkkisovellus siitä, miten STEVAL-STWINKT1B-kehitysalustan antureiden tuottamaa dataa voidaan striimata USB-kaapelin kautta tietokoneelle hyödyntäen laitteen Virtual COM Port luokkaa. Jotta laitteen striimaamaa dataa voidaan seurata, tarvitaan serial terminal ohjelmisto, kuten TeraTerm tai puTTY. [37, s. 6.]

Serial_DataLog-sovellus toimii niin, että kehitysalustan resetoinnin jälkeen asennettu laiteohjelmisto konfiguroi HAL:n, laitteen kellot, LED1:sen sekä LED2:sen. Se myös alustaa USB-

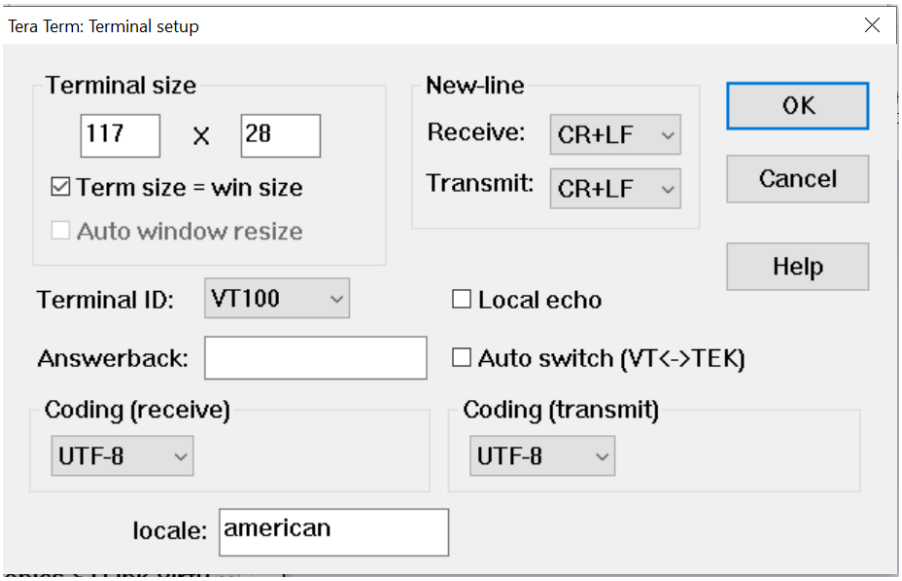
oheislaitteen, luo säikeet GetData_Thread ja WriteData_Thread sekä aktivoi FreeRTOS-ajastimen. [37, s. 6.]

Esimerkkiprojektin ajamista varten kehitysalusta yhdistetään yhdessä STLINK-V3MINI:n kanssa tietokoneeseen omilla micro-USB -kaapeleillaan. Kehitysalustaan muodostetaan yhteys, kuten kuvissa 11-12 on esitetty. STM32CubeProgrammerissa siirrytään Erasing & Programming -valikkoon. Browse-toiminnon avulla haetaan ladattu laiteohjelmisto, josta etsitään esimerkkiprojektit kansiosta Serial_DataLog-kansio. Serial_DataLog-kansiosta valitaan haluttu binaaritiedosto, joka on tässä tapauksessa USB_DataLog.bin. Erasing & Programming -valikossa laitetaan raksi ruutuun, jossa lukee Run after programming, jonka jälkeen painetaan painiketta Start Programming. Tällä tavoin esimerkkiprojekti on saatu asennettua kehitysalustaan.

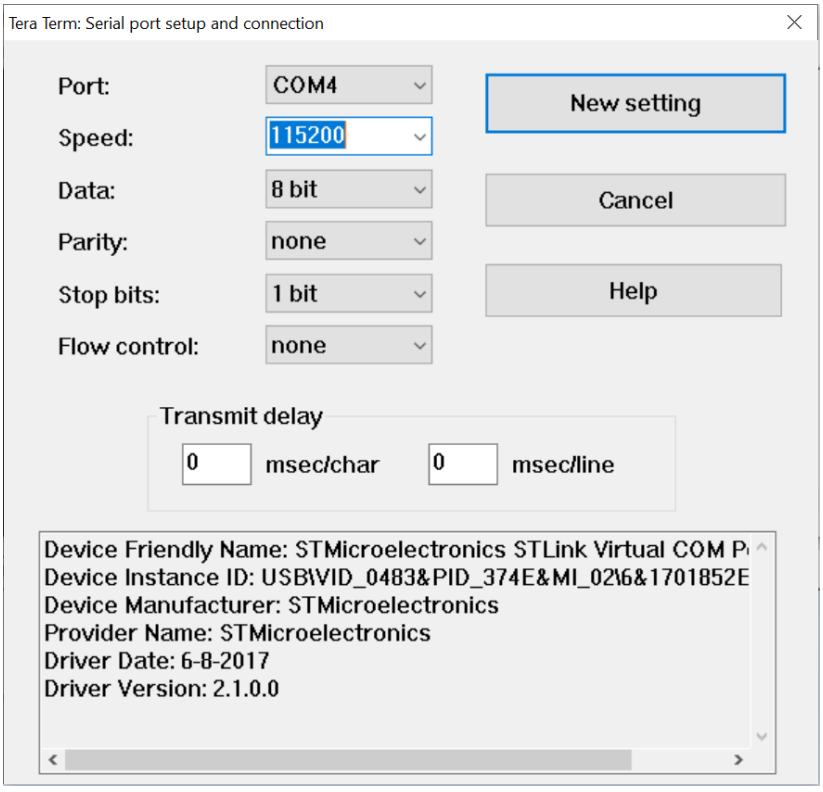
Kun laiteohjelmisto on asennettu kehitysalustaan, tietokone tunnistaa kehitysalustan Virtual COM Port -luokkana, tässä tapauksessa COM 3. Datan striimaamista varten avataan TeraTerm. Kuvissa 16-18 on esitetty asetukset jotka ovat tarpeelliset serial-yhteyden muodostamista varten TeraTerm -ohjelmassa.



Kuva 16. Uuden yhteyden muodostaminen TeraTerm -ohjelmassa.

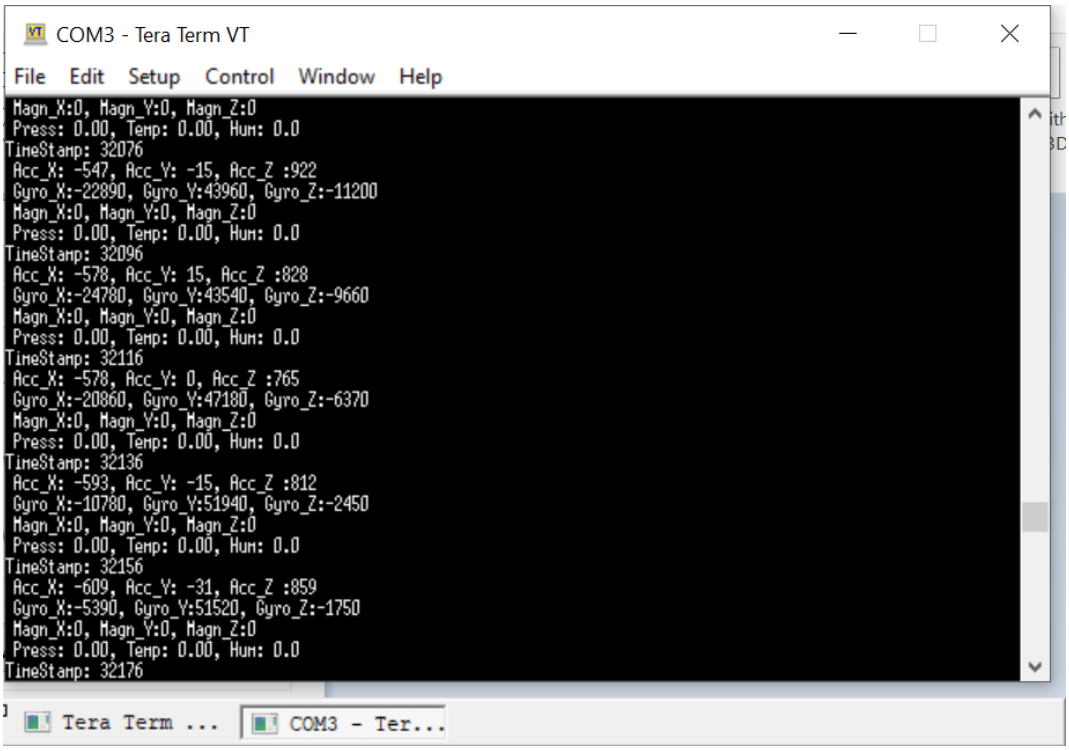


Kuva 17. TeraTerm terminaalin asetukset.



Kuva 18. TeraTerm serial port sekä yhteyden asetukset.

Kun yhteys on onnistunut, datan striimauksen tulisi näyttää samantapaiselta, kuin kuvassa 19.



Kuva 19. Onnistuneen yhteyden striimattu data TeraTerm terminaalissa.

Kuten kuvassa 19 näkyy, STEVAL-STWINKT1B striimaa dataa sarjaterminaaliin. Striimatussa datassa näkyy timestamp, kiihtyvyysanturin tuottama data, gyroskoopin tuottama data, korkeustiedot, paineanturin tuottama data ilmanpaineesta, lämpötila-anturin tuottama data sekä kosteusanturin tuottama data suhteellisesta ilmankosteudesta.

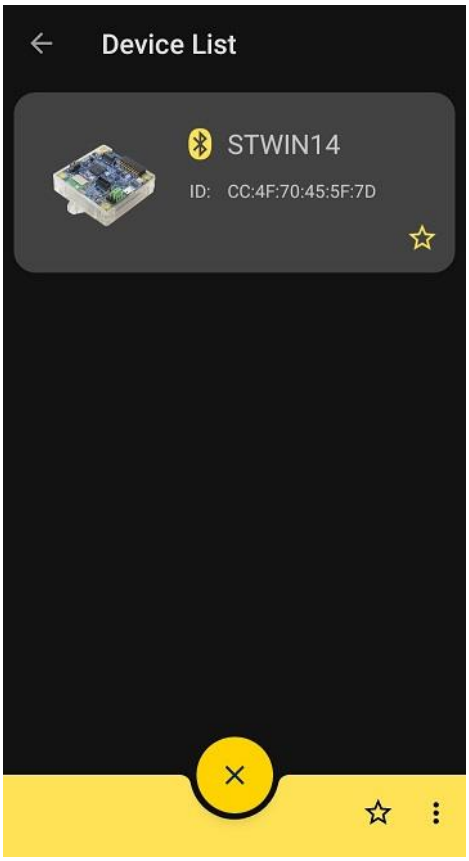
5.2.2 BLE_SampleApp -esimerkkiprojekti

STEVAL-STWINKT1B firmware-pakettiin kuuluva BLE_SampleApp -esimerkkiprojekti tarjoaa kehitysalustalle Bluetooth Low Energy konfiguraation, jonka avulla kehitysalusta voi striimata ympäristöantureiden tuottamaa dataa BLE-yhteyden välityksellä mobiililaitteelle. Esimerkkiprojektia varten tarvitaan sekä Android- että iOS-mobiililaitteille suunniteltu ST BLE Sensor-niminen ilmainen mobiilisovellus. [37, s. 9.]

Kun kehitysalustalle on suoritettu firmwaren asennus ja reset, firmware suorittaa seuraavat aloitustehtävät:

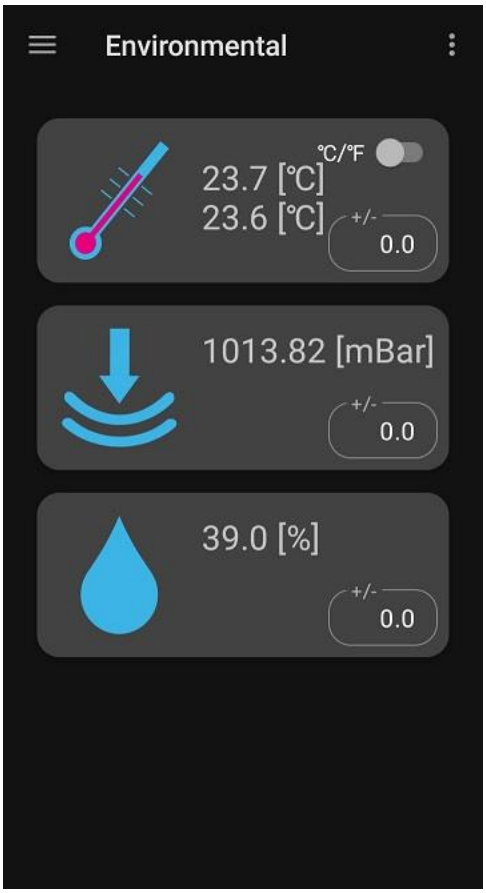
- HAL:n konfigurointi
- akun laturin konfigurointi
- kohdealustan alustukset
    - USB-oheislaite debuggausta varten
        - LED1 ja LED2
        - ympäristöanturit
        - BLE stack alustus
    - BLE-palveluiden alustus
    - ajastimen alustus
    - main loop aloitus
        - LED hallinta
        - BLE event hallinta
        - ympäristöantureiden datan hallinta [37, s. 9.]

Kehitysalusta on yhdistettävä BLE_SampleApp-esimerkkisovellusta varten STLINK-V3MINI:n kanssa tietokoneeseen samalla tavalla kuin Serial_DataLog-esimerkkisovelluksessa. Yhteys kehitysalustaan muodostetaan samalla tavalla kuin Serial_DataLog-sovelluksessa STM32CubeProgrammerissa. Yhteyden muodostamisen jälkeen Erasing & Programming -valikosta etsitään Browse-toiminnon kautta projektia varten tarvittava binaaritiedosto, BLE_SampleApp.bin. Tarkistetaan, että run after programming on valittuna ja painetaan painiketta start programming. STM32CubeProgrammer ilmoittaa, että yhteys on katkaistu, joka on normaalia sovelluksen toiminnalle.



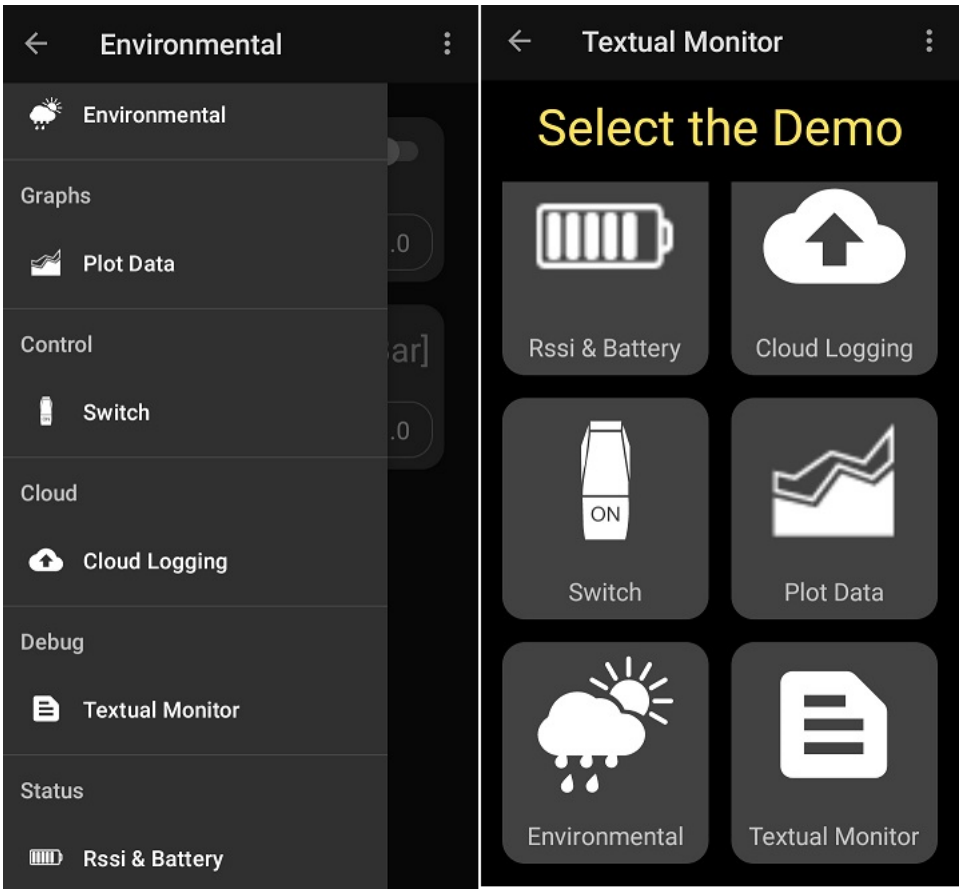
Kuva 20. ST BLE Sensor, vapaat BLE yhteydet.

Avataan ST BLE Sensor-sovellus mobiililaitteella ja etsitään oikea laite, johon halutaan yhdistää. Kuvassa 20 on sovelluksen näkymä, jossa on listattuna STEVAL-STWINKT1B-kehitysalusta vapaana yhteytenä. Klikkaamalla STWIN14 painiketta mobiililaite yrittää muodostaa yhteyden kehitysalustaan.



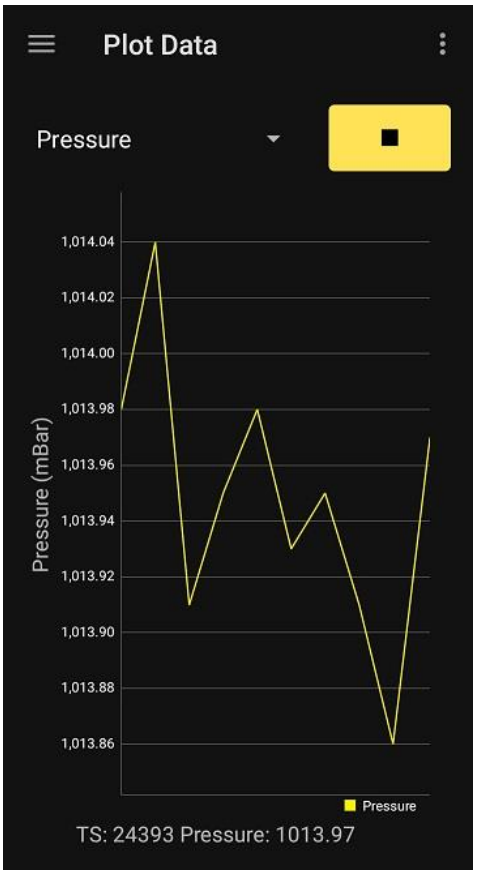
Kuva 21. ST BLE Sensor, ympäristöanturit.

Kuvassa 21 on mobiililaitteen näkymä, kun BLE-yhteys on muodostettu kehitysalustaan. Näkymässä on kahden eri lämpötila-anturin, analoginen ja digitaalinen lämpötila-anturi, tuottama data, paineanturin tuottama data sekä kosteusanturin tuottama data suhteellisesta kosteudesta. Klikkaamalla jokaista anturia erikseen voidaan saada lisää tietoa anturin tuottamasta datasta. Näkymän vasemmassa reunassa on painike valikolle.



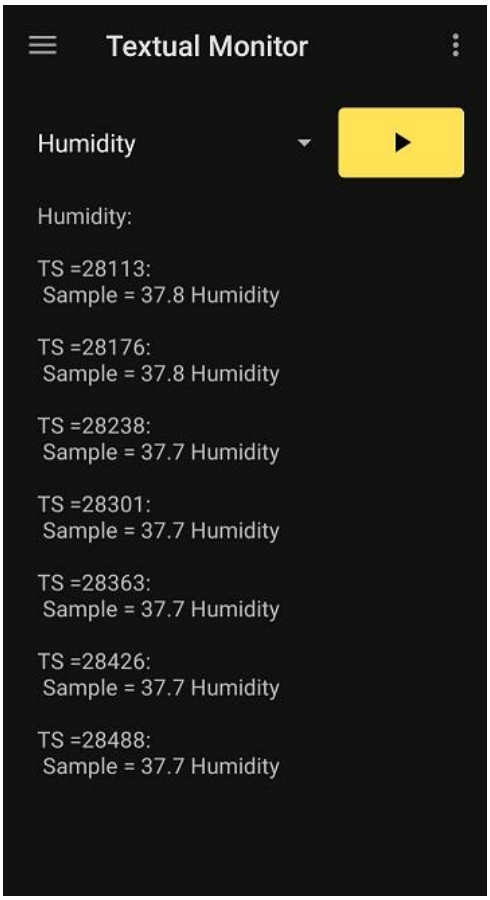
Kuva 22. ST BLE Sensor -sovelluksen valikko.

Kuvassa 22 STE BLE Sensor -sovelluksen valikko, josta voidaan valita haluttu demo testattavaksi. Kuvassa vasemmalla on näkymän vasemman reunan valikko. Kuvassa oikealla on valittavina samat demot, mutta kuvakkeina. Environmental valikosta päästään samaan näkymään kuin kuvassa 21.



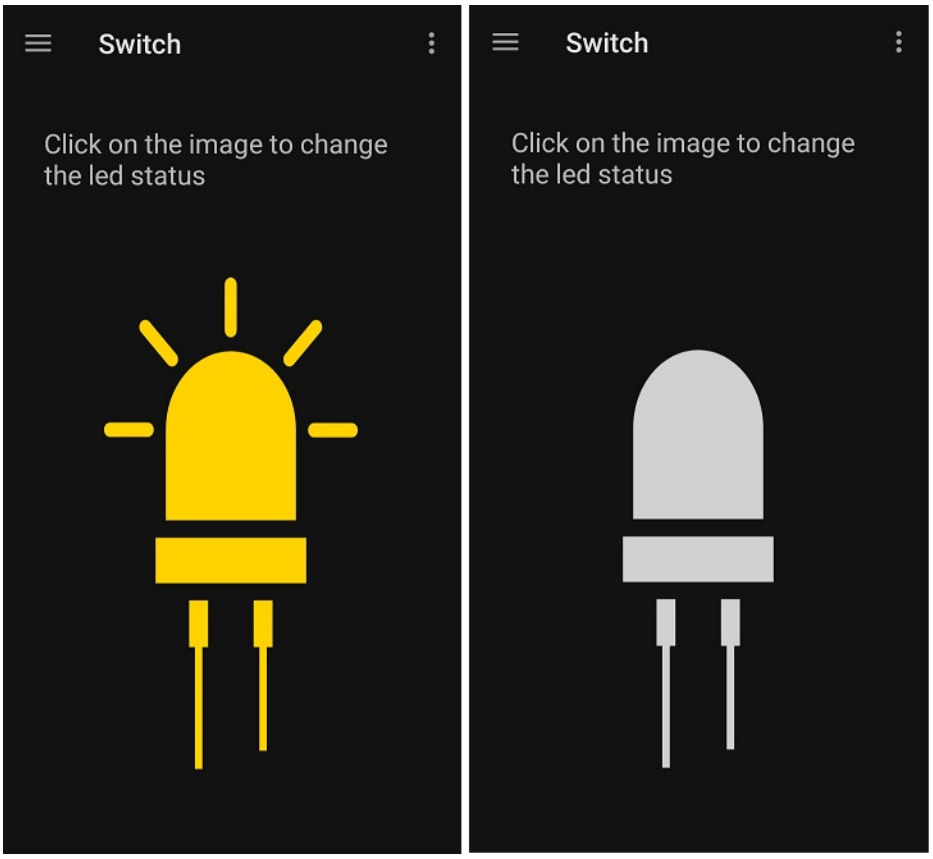
Kuva 23. ST BLE Sensor, Plot Data näkymä.

Mikäli ympäristöantureiden tuottamaa dataa halutaan tallentaa kaavioihin, se onnistuu kuvassa 23 näkyvän Plot Data demon kautta. Plot Data -demossa voidaan valita, mitä ympäristöanturia halutaan seurata ja tallentaa saman anturin data kaavioon. Jos sama data halutaan tallentaa mieluummin tekstimuodossa, onnistuu se Textual Monitor -demon kautta, joka näkyy kuvassa 24. Demossa voidaan alasvetovalikon kautta valita, minkä anturin tuottamaa dataa halutaan tallentaa tekstimuodossa.



Kuva 24. ST BLE Sensor, Textual Monitor näkymä.

Switch-demon kautta voidaan kontrolloida kehitysalustan sisäänrakennettua vihreää LED-valoa. Switch-demo on esitetty kuvassa 25. Painamalla LED-valon kuvaketta voidaan sammuttaa ja laittaa päälle kehitysalustan LED-valo.



Kuva 25. ST BLE Sensor, Switch toiminto.

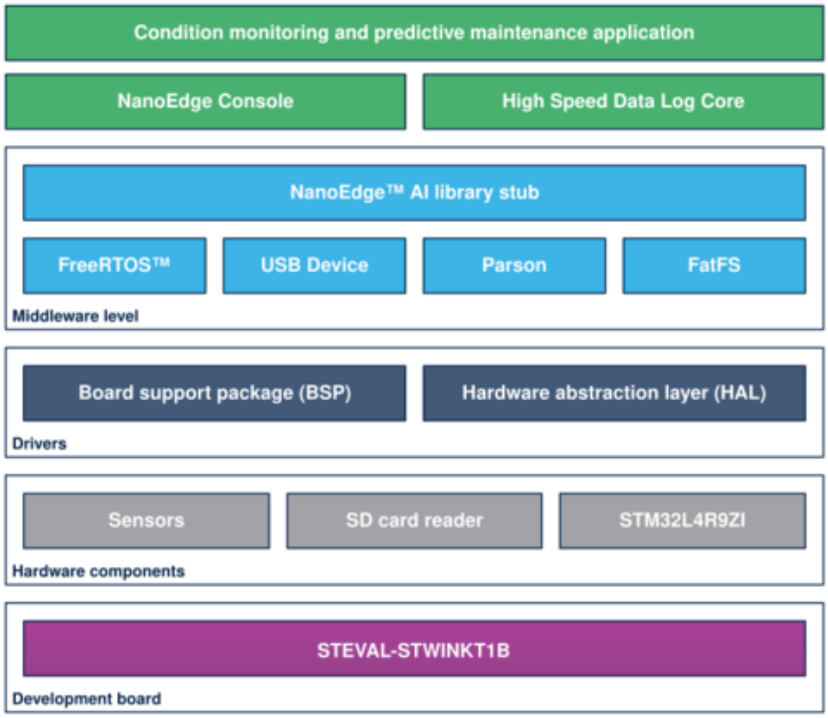
Cloud Loggin demolla voidaan tallentaa kerättyä anturidataa suoraan pilvipalveluun. Rssi & Battery -demon kautta voidaan säädää kehitysalustan virrankäyttötoimintoja, kuten laitteen power ON/OFF-rutiinit.

6 STEVAL SensorTile Wireless Industrial Node ja FP-AI-NANOEDG1

STEVAL-STWINKT1B Sensor Node kehitysalustan koneoppimisytimen tekoälyominaisuuksien testaamiseksi päätettiin hyödyntää ST Microelectronics -yrityksen kehittämää FP-AI-NANOEDG1-V2.0.0 -kunnonvalvontafunktiopakettia. Tähän päädyttiin, sillä FP-AI-NANOEDG1-V2.0.0 -kunnonvalvontafunktiopaketti oli opinnäytetyön tekohetkellä ajankohtaisin esimerkkiprojekti, joka ST Microelectronics -yrityksellä oli tarjolla kohdealustan koneoppimisytimen testausta varten.

FP-AI-NANOEDG1-V2.0.0 on STEVAL-STWINKT1B-kehitysalustan tärinäanturin tuottamaan dataan perustuva kunnonvalvontafunktiopaketti. Se on suunniteltu toimimaan STM32-tuoteperheen mikrokontrollereilla. Funktiopaketin tarkoitus on auttaa pistämään alulle NanoEdge AI Studiolla luotujen tekoälykirjastojen kehitys ja ottamaan ne käyttöön kohdealustassa. [38; 39.]

FP-AI-NANOEDG1:ssa on interaktiivinen CLI noden määritystä, datan tallennusta sekä oppimis- ja tunnistamisvaiheiden hallintaan. CLI:n avulla käyttäjä voi konfiguroida ja kontrolloida kehitysalustaa, logittaa tärinäanturin tuottamaa dataa, jota käytetään NanoEdge AI-kirjastojen generointiin sekä suorittamaan koulutus- ja tunnistusoperaatioita AI-kirjastoista laitteen sisäisesti. FP-AI-NANOEDG1-V2.0.0 -kunnonvalvontafunktiopaketin arkkitehtuuri on esitetty kuvassa 26. [39.]



Kuva 26. FP-AI-NANOEDG1-V2.0.0 -kunnonvalvontafunktiopaketin rakenne. [39]

Funktiopaketissa on mukana laajennettu autonominen käyttömuoto sekä laajennettu painikekäyttöinen tila. Nämä käyttömuodot ovat olemassa sellaisia tilanteita varten, jolloin CLI:n käyttäminen ei ole mahdollista, esimerkiksi kun kehitysalustaa halutaan käyttää paristokäyttöisenä sellaiseen koneeseen tai laitteeseen kytkettynä, jossa halutaan suorittaa kunnonvalvontaa. Kaikki perusarvot eri toimintojen toteuttamista varten, kuten datalog, learn ja detect, on laiteohjelmistossa mukana. [39.]

Painikekäyttöisessä tilassa käyttäjä voi käynnistää tai pysäyttää eri toimintoja tai prosesseja kehitysalustassa painamalla kehitysalustan USR-painiketta. Tarjolla olevat toiminnot on esitetty taulukossa 1.

Button Press	Kuvaus	Toiminto
SHORT_PRESS	Painiketta painetaan alle 200 ms ajan ja päästetään irti	Detect
LONG_PRESS	Painiketta painetaan päälle 200 ms ajan ja päästetään irti	Learn
DOUBLE_PRESS	Kaksi peräkkäistä, lyhyttä, painallusta alle 500 ms sisällä	Datalog
ANY_PRESS	Painiketta painetaan ja päästetään irti (toimii päällekkäin muiden tilojen kanssa)	Toiminto pysäytetään

Taulukko 1. FP-AI-NANOEDG1-funktiopaketin toiminnot USR-painikkeelle. [39]

Kullakin hetkellä käynnissä oleva toiminto voidaan nähdä kehitysalustan sisäänrakennetuista LED-valoista, joita on kaksi: yksi vihreä LED-valo ja yksi oranssi LED-valo. Taulukossa 2on esitetty, mitä LED-valojen vilkkuminen merkitsee. [39.]

Pattern	Vihreä	Oranssi
OFF	Power OFF	Power OFF
ON	idle	system error
BLINK	Data logging	Micro SD card missing
BLINK_SHORT	Detecting, no anomaly	Anomaly detected
BLINK_LONG	Learning, status OK	Learning, status FAILED

Taulukko 2. FP-AI-NANOEDG1-funktiopaketin LED-valojen indikoinnit. [38]

Funktiopaketin autonominen käyttömuoto mahdollistaa sen, että se voidaan ottaa käyttöön automaattisesti, kun kehitysalustaan kytketään virrat päälle. Käyttäjä voi valita jopa 10 käskyä, jotka kehitysalusta suorittaa ketjutettuna itsenäisesti. Automodea kontrolloidaan mikro-SD-kortin root-hakemistoon sijoitettavalla Execution_config.json-tiedostolla, jossa määritetään läpikäytävät toiminnot. Root-hakemistoon voidaan sijoittaa myös Device_Config.json-tiedosto, jolla voidaan ohittaa peruskonfiguraatioasetukset. [39.]

6.1 Työssä käytetyt laitteistot ja ohjelmistot

FP-AI-NANOEDG1-V2.0.0 -kunnonvalontafunktiopaketin testauksessa käytettiin seuraavia laitteita:

- STEVAL-STWINKT1B -kehitysalusta
- STLINK-V3MINI

- Ohjelmointikaapeli ja kaksi micro-USB -kaapelia
- FAT32-formatoitu micro-SD -kortti
- Windows 10 -tietokone

FP-AI-NANOEDG1-V2.0.0 -kunnonvalvontafunktiopaketin testaamisessa käytettiin seuraavia ohjelmistoja ja työkaluja:

- STM32 CubeIDE, versio 1.5.0
- STM32 STLINK Utility, ST SW_LINK 004
- TeraTerm
- Anaconda 3
- NanoEdge AI Studio

Jotta FP-AI-NANOEDG1-V2.0.0 -kunnonvalontafunktiopakettia voidaan käyttää tarvitaan seuraavat laitteistot:

- STEVAL-STWINKT1B-kehitysalusta
- STLINK-V3MINI
- Ohjelmointikaapeli
- Kaksi micro-USB -kaapelia
- FAT32-formatoitu micro-SD -kortti
    - 16 GB on riittävän kokoinen
- Windows 10 pöytäkone tai laptop [39.].

6.2 Työvaiheet

FP-AI-NANOEDG1-V2.0.0-kunnonvalvontafunktiopaketin käyttö on monivaiheinen prosessi. Työvaiheita ovat SD-kortin formatointi, anturidatan keräys ja käsittely, tekoälykirjaston etsiminen NanoEdge Ai Studiolla, projektin uudelleenkääntäminen ja ajaminen kohdealustalle sekä tekoälyn koulutus kohdealustassa.

FP-AI-NANOEDG1-V2.0.0-kunnonvalvontafunktiopaketin testauksessa hyödynnettiin sille tarkoitettuja käyttöohjeita: FP-AI-NANOEDG1 V2.0 Getting Started sekä FP-AI-NANOEDG1 V2.0 User Manual.

6.2.1 Micro-SD -kortin formatointi

Micro-SD -kortti formatointi tehtiin syöttämällä 16 GB:n micro-SD -kortti Windows 10 -käyttöjärjestelmäisen kannettavan tietokoneen SD-kortti slottiin. Tämän jälkeen avattiin tietokoneen file explorer josta etsittiin SD-kortin kuvake. Kuvaketta klikattiin hiiren oikealla painikkeella, jonka jälkeen valittiin ”format”-toiminto.

Format-ikkunassa formatoinnin asetuksiksi asetettiin seuraavaa:

- Capasity = 14,8 GB
- File System = FAT32
- Allocation unit size = 64 kilobytes

Kun formatoinnin asetukset oli määritetty, painettiin Start-painiketta. Ruudulle ilmestyi popup-varoitus, joka kertoi formatoinnin aiheuttavan SD-kortilla olemassa olevan datan poistamisesta. Koska SD-kortti oli valmiiksi tyhjä, varoituksessa painettiin OK-painiketta.

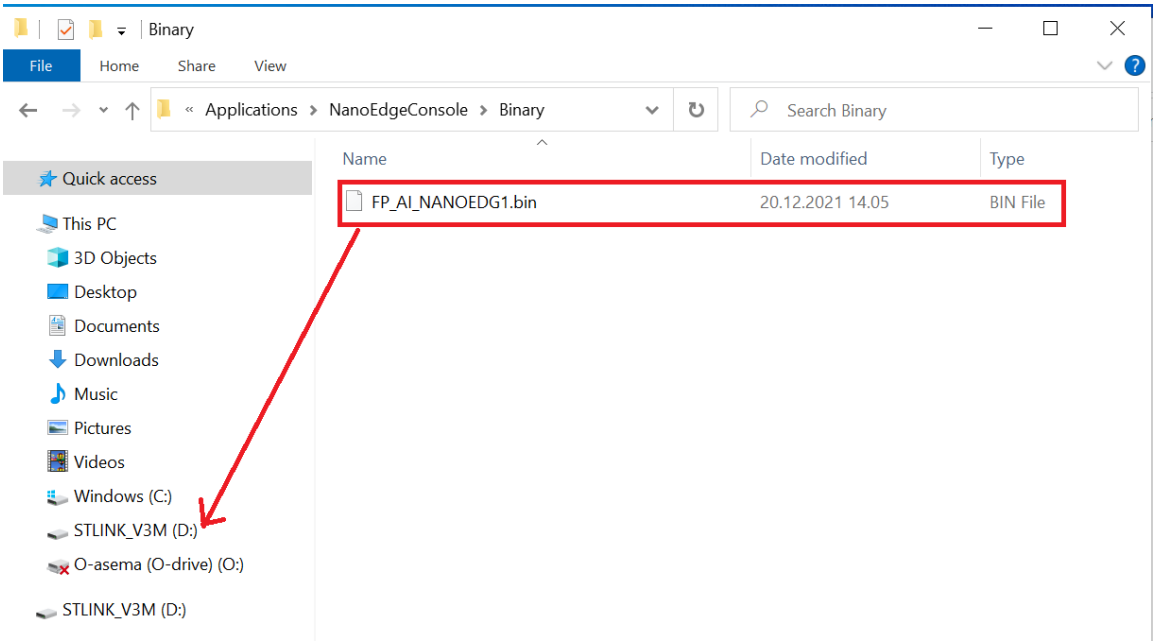
Format Complete ikkunan ilmestyttyä tietokoneen ruudulle SD-kortin formatointi oli valmis. SD-kortti poistettiin tietokoneesta ja syötettiin paikoilleen STEVAL-STWINKT1B-kehitysalustan omaan micro-SD -kortti slottiin.

6.2.2 Anturidatan keräys ja käsittely

FP-AI-NANOEDG1-V2.0.0 ladattiin ST Microelectronics verkkosivuilta ja purettiin haluttuun hakemistoon. Funktiopaketin hakemistosta etsittiin projektin binaaritiedostot, jotka löytyivät seuraavan kansiopolun takaa:

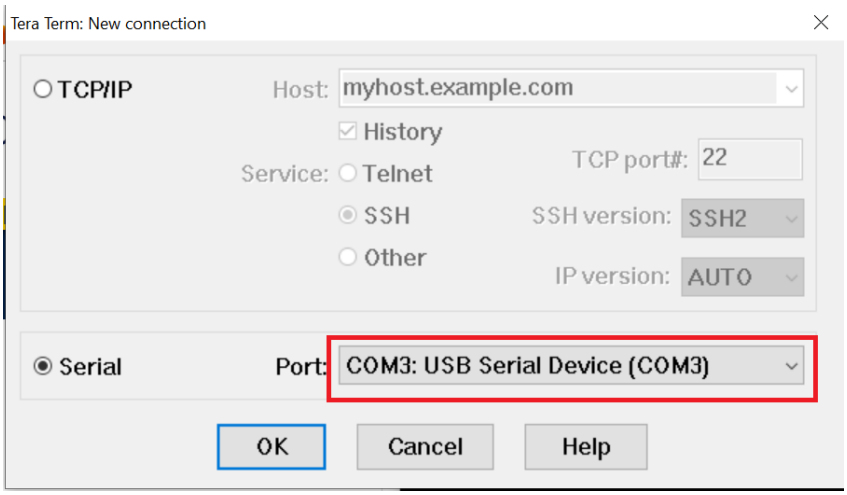
...\FP-AI-NANOEDG1\en.FP-AI-NANOEDG1_V2.0.0_RC7_v2.0.0\FP-AI-NANOEDG1_V2.0.0\Projects\STM32L4R9ZI-STWIN\Applications\NanoEdgeConsole\Binary

STEVAL-STWINKT-1B-kehitysalusta yhdistettiin STLINK-V3MINI -debuggerin kanssa tietokoneeseen. File Explorerin kautta etsittiin STLINK-V3MINI, johon kopioitiin funktiopaketin binaaritiedosto, kuten kuvassa 27 on esitetty. Tällä tavoin saatiin projektin binaaritiedostot ohjelmoitua kohdealustaan.

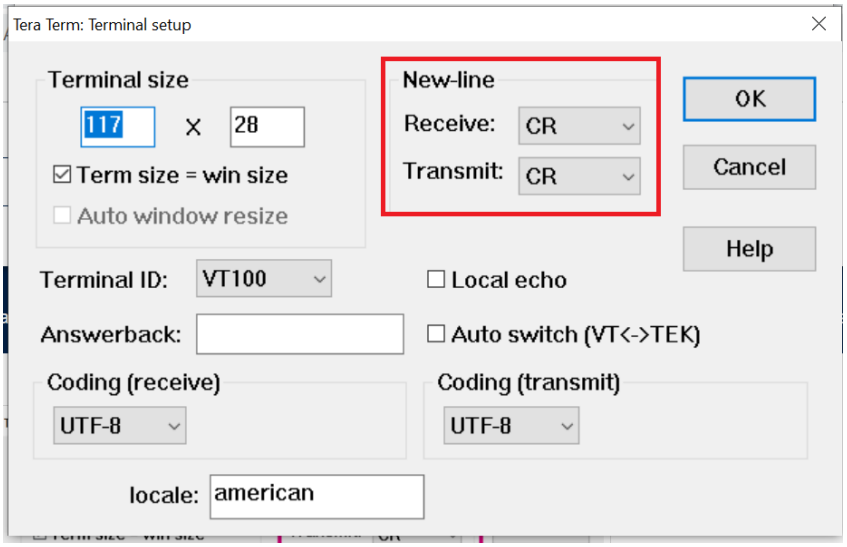


Kuva 27. Binaaritiedostojen ohjelmointi.

Kun binaaritiedostot saatiin ohjelmoitua onnistuneesti STLINK-V3MINI -debuggeriin, muodostettiin kohdealustaan yhteys TeraTerm-ohjelmalla. TeraTerm-sarjayhteyden asetukset on esitetty kuvissa 28 ja 29.

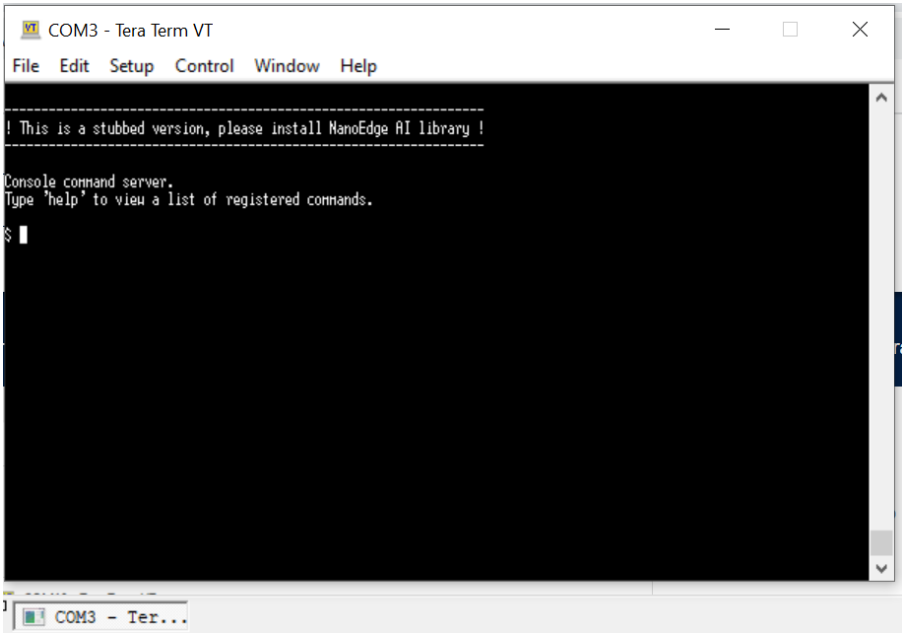


Kuva 28. Sarjayhteyden asetukset, COM-portin valinta.



Kuva 29. Sarjayhteyden asetukset, Receive ja Transmit asetukset.

Kun sarjayhteys saatiin muodostettua, painettiin kehitysalustan RESET-painiketta, joka on esitetty kuvassa 5. TeraTerm-ikkunaan ilmestyi kuvan 30 mukainen näkymä, joka merkitsi sitä, että yhteys oli muodostunut onnistuneesti.



Kuva 30. FP-AI-NANOEDG1-V2.0.0, onnistunut yhteys.

Kun kehitysalustaan on muodostettu sarjayhteys onnistuneesti, voidaan CLI kautta määrittää kehitysalustalle, mitä antureita halutaan käyttää ja kerätä dataa näistä valituista antureista. Esimerkiksi ISM330DHCX-anturissa on sisäänrakennettuna aliantureita, joita voidaan tarvittaessa konfiguroida yksitellen terminaalin kautta. Terminaalin kautta voidaan aktivoida tai sammuttaa anturi/alianturi, asettaa näille output data rate (odr) valittavissa olevista vaihtoehdoista sekä asettaa fullscale alue valittavissa olevista vaihtoehdoista [37.]

Käytettävissä olevista antureista haluttiin lisää informaatiota, joten terminaaliin kirjoitettiin seuraavia komentoja:

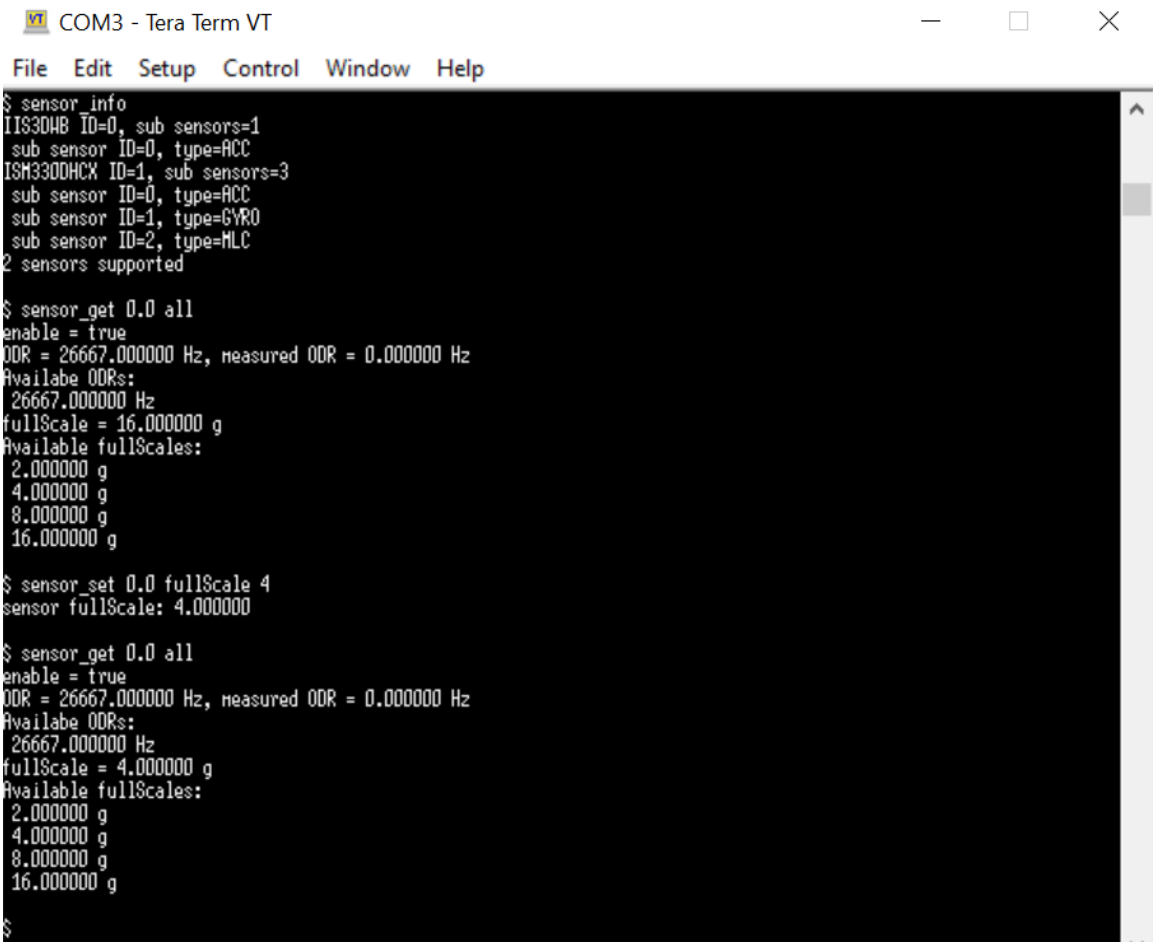
$ sensor_info

$ sensor_get 0.0 all

Näillä komennoilla saatiin oleellista tietoa kehitysalustan kiihtyvyysanturista ja sen mahdollisista asetuksista. Mittausskaalan asettamiseksi kirjoitettiin seuraava komento:

$ sensor_set 0.0 fullScale 4

Kuvassa 31 on esitetty kaikki ajetut komennot ennen datalog-prosessin aloitusta sekä kaikki informaatio, jota saatiin komennoista.



Kuva 31. Antureiden asetusvaihtoehdot.

Datalog-prosessi aloitettiin kirjoittamalla seuraava komento terminaaliin:

$ start datalog

Datalog-prosessin aikana kehitysalustassa vilkkui vihreä LED-valo taulukon 2 osoittamalla tavalla, indikoiden sitä, että datalog-prosessi oli käynnissä. Kun datalog-prosessi haluttiin lopettaa, painettiin tietokoneen näppäimistöstä ESC-painiketta. STEVAL-STWINKT1B-kehitysalustassa olevaan SD-korttiin muodostuu uusi datalog-kansioon jokaisen datalog-prosessin päätyttyä.

Datalog-prosessia testattiin kaikenkaikkiaan neljä kertaa ennen kuin kerättiin varsinaiset datalog-tiedostot kehitysalustan koneoppimisytimen testausta varten. Testausta varten datalogeja kerättiin kaksi, ensimmäinen edustaen normaaleja signaaleja ja toinen edustaen epänormaaleja signaaleja: STM32_DL_005 ja STM32_DL_006.

Datalogeja tarvittiin kaksi, koska FP-AI-NANOEDG1-V2.0.0-kunnonvalvontafunktiopaketin toiminnalle on tärkeää saada kaksi erilaista datalogia: yksi edustaen seurattavaa konetta tai laitetta silloin kun se toimii normaalisti ja toinen, edustaen tilannetta kun seurattava kone tai laite toimii epänormaalisti tai viallisesti.

Funktiopaketin testausta varten tehdyt datalogit tehtiin kahdesta eri tilanteesta. Ensimmäinen datalog kerättiin 15 minuutin ajalalta, kun kohdealusta, eli STEVAL-STWINKT1B Sensor Node, oli paikoillaan häiritsemättä pöydällä. Toinen datalog kerättiin myös 15 minuutin ajalta, mutta tällä kertaa kohdealustaan kohdistettiin mekaanista tärinää koko datalogin keräyksen ajan, edustaen ”vikatilaa” kerätyissä signaaleissa. Kuten aiemmin, datalog prosessit lopetettiin painamalla näppäimistön ESC-painiketta.

Kun normaalin ja epänormaalin tilan datalog-prosessit oli käyty läpi, luotu sarjayhteys katkaistiin kehitysalustaan ja kehitysalustasta irroitettiin SD-kortti. SD-kortti siirrettiin takaisin tietokoneen micro-SD -kortti slottiin, jonka jälkeen kortilta kopioitiin kansiot STM32_DL_005 ja STM32_DL_006 FP-AI-NANOEDG1-V2.0.0-projektin Sample-DataLogs-kansioon, joka oli seuraavan kansiopolun takana:

...\FP-AI-NANOEDG1\en.FP-AI-NANOEDG1_V2.0.0_RC7_v2.0.0\FP-AI-NANOEDG1_V2.0.0\Utilities\AI_resources\DataLog

Tämän jälkeen micro-SD -kortti siirrettiin takaisin kehitysalustaan.

Koska kerätyt datalogit eivät ole sellaisenaan ihmissilmin luettavassa muodossa, taikka NanoEdge AI Studiolle sopivassa muodossa, on datalogit esikäsiteltävä. Datalogien esikäsittelyä varten FP-AI-NANOEDG1 -kunnonvalvontafunktiopaketissa tulee mukana Python Jupyter Notebook-tiedosto, Parser_for_HS_Logged_Data-ipynb, jossa on kirjalliset ohjeet ja komennot datan käsittelyä varten.

Jotta data saatiin esikäsiteltyä sopivaan muotoon, avattiin aiemmin esitetyssä kansiossa Anaconda-terminaali. Anaconda terminaalissa ajettiin seuraavat Python-scriptit:

> python STM_DataParser.py ./Sample-DataLogs/STM32_DL_005

> python STM_DataParser.py ./Sample-DataLogs/STM32_DL_006

Näiden funktiokutsujen tehtävä on kääntää datalogit luettavissa olevaan .csv-tiedostomuotoon. Kun binaarimuotoinen data saatiin käännettyä .csv-tiedostoiksi, täytyi vielä ajaa seuraava Python-script Anaconda-terminaalissa:

> python PrepareNEAIData.py ./Sample-DataLogs/STM32_DL_005/ ./Sample-DataLogs/STM32_DL_006/ -seqLength 0124 -sensorName IIS3DWB_ACC

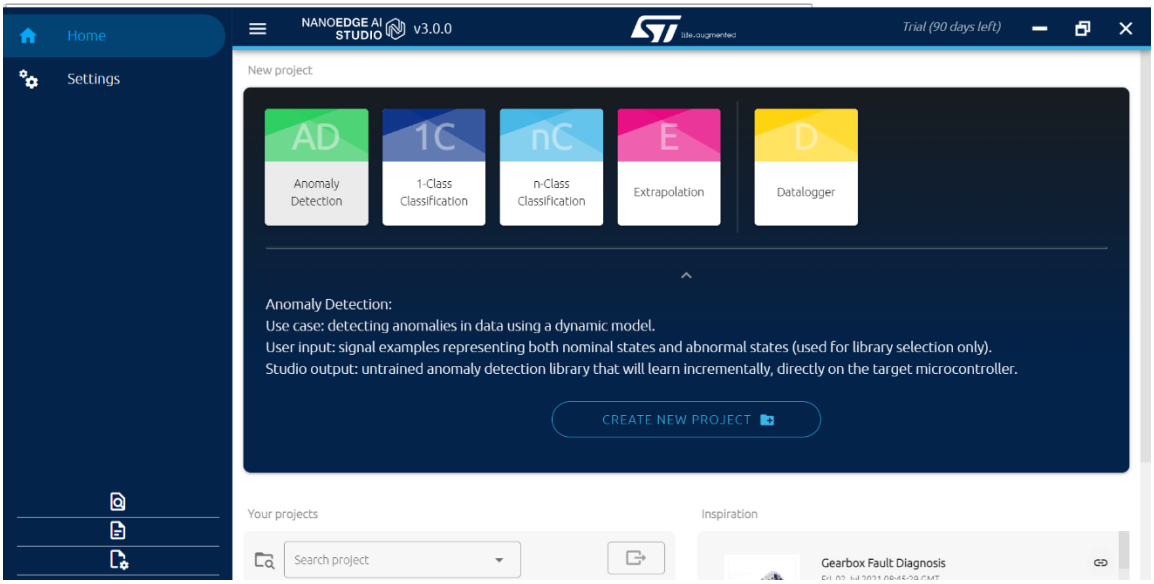
Funktiokutsu loi kaksi .csv-tiedostoa: normalSegments.csv ja abnormalSegments.csv. Nämä tiedostot ovat tärkeitä tekoälykirjaston etsimisessä NanoEdge AI Studiossa, sillä ne toimivat kontekstuaalisena datana hakukoneessa. Luotu normalSegments.csv edustaa laitteen tai koneen signaaleja sen toimiessa normaalisti ja abnormalSegments.csv edustaa laitteen tai koneen signaaleja sen toimiessa epänormaalisti tai viallisesti. Kun signaalitiedostot on luotu onnistuneesti, Anaconda terminaaliin pitäisi ilmestyä kuvan 32 mukainen ilmoitus, joka kertoo, että tiedostot on käsitelty ja kuinka monta hyvää ja huonoa näytettä on löytynyt.

```
Files have been prepared with following details.
Number of good samples :  23200
Number of bad samples :  23526
```

Kuva 32. Anaconda-terminaali, kun signaalien käsittely on onnistunut.

6.2.3 Tekoälykirjaston hakeminen

Kun Anacondalla saatiin käännettyä kerätyistä datalogeista esikäsitellyt signaalitiedostot, normalSegments.csv ja abnormalSegments.csv, oli aika aloittaa sopivan tekoälykirjaston etsiminen NanoEdge AI Studiolla.



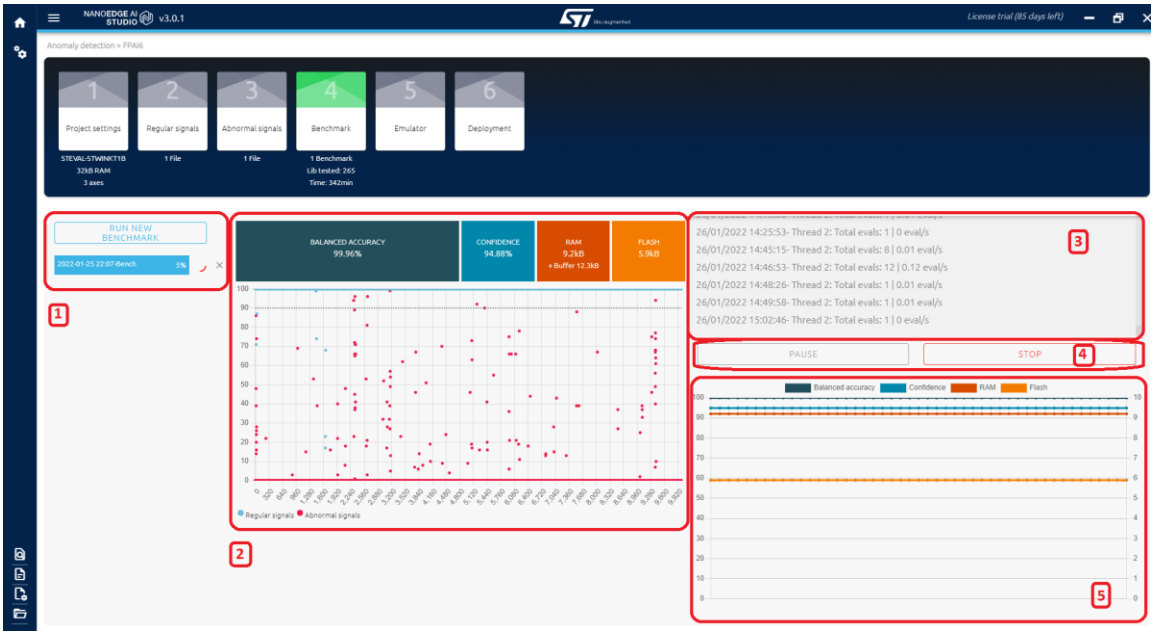
Kuva 33. NanoEdge AI Studio, päävalikko.

NanoEdge AI Studion päävalikko on esitetty kuvassa 33. Koska käytettävä funktiopaketti on tarkoitettu havaitsemaan epätavallisia signaaleja käytettävällä kohdealustalla, valittiin projektiksi Anomaly Detection. Anomaly Detection projektin asetuksiksi laitettiin nimen ja kuvauksen lisäksi seuraavaa:

- Max RAM = 32 kB
- No Flash limit
- Target = STEVAL-STWINKT1B
- Sensor type = Accelerometer 3 axes

Kun projektin asetukset saatiin tallennettua, siirryttiin Regular signals valikkoon. Regular signals-valikossa ADD SIGNALS painikkeen kautta etsittiin normalSegments.csv-tiedosto, joka tuotiin projektiin import-toiminnon kautta edustamaan projektin normaaleja signaaleja. NanoEdge AI Studio käsittelee signaalit ja luo niistä kaaviokuvat: jokaista signaalia kohden tulee kaavio signaalista ja lisäksi kyseisen signaalin FFT. Koska signaalit koskevat kolmiakselista tärinäanturia, on käytettäviä signaalejakin kolme kappaletta. Kun normalSegments.csv saatiin tuotua projektiin, sama prosessi tehtiin Abnormal signals -valikossa, jossa etsittiin abnormalSignals.csv-tiedosto tuotavaksi projektiin. Kumpiakaan signaalijoukkoja ei käsitelty suodattimella.

Kun signaalit saatiin tuotua projektiin, käynnistettiin Benchmarking-prosessi. Benchmarking-prosessi etsii projektiin sopivan tekoälykirjaston käyttäen signaalitiedostoja sekä muita aiemmin määritettyjä asetuksia parametreinaan. Benchmark-prosessi tulisi mielellään antaa ajaa 100%:n asti.



Kuva 34. Benchmarking-näkymä.

Kuvassa 34 on esitetty Benchmarking-näkymä ja sen eri osiot. Kohdassa 1 näkyy luodut, tai meneillään olevat, Benchmarkit. Ikkunassa 2 on Benchmark-prosessin tulokset NanoEdge AI Studion löytämään, parhaaseen mahdolliseen tekoälykirjastoon. Ikkunassa 3 näkyy prosessin säikeet ja informaatiota säikeistä, kuten kuinka monta arviota säie tekee sekunnissa. Ikkunaan tulee ilmoitus joka kerta, kun NanoEdge AI Studio löytää uuden parhaiten sopivan tekoälykirjaston. Ikkunassa 4 näkyy PAUSE ja STOP painikkeet. PAUSE-painikkeella voidaan keskeyttää prosessi tarpeen mukaan ja STOP-painikkeella pysäytetään prosessi kokonaan, mikäli ollaan päästy toivottuun tarkkuuteen saakka. Ikkunassa 5 on graafisesti esitettynä tekoälymallin hakuprosessin kehitys.

Ikkunassa 2 näkyvät Benchmarking-prosessin tulokset. Tuloksiin vaikuttaa neljä eri kriteeriä: Balanced Accuracy, Confidence, RAM sekä Flash. Balanced accuracy kertoo tekoälykirjaston kyvykkyydestä tunnistaa sille annetut signaalit oikein. Mitä korkeampi prosenttiluku balanced

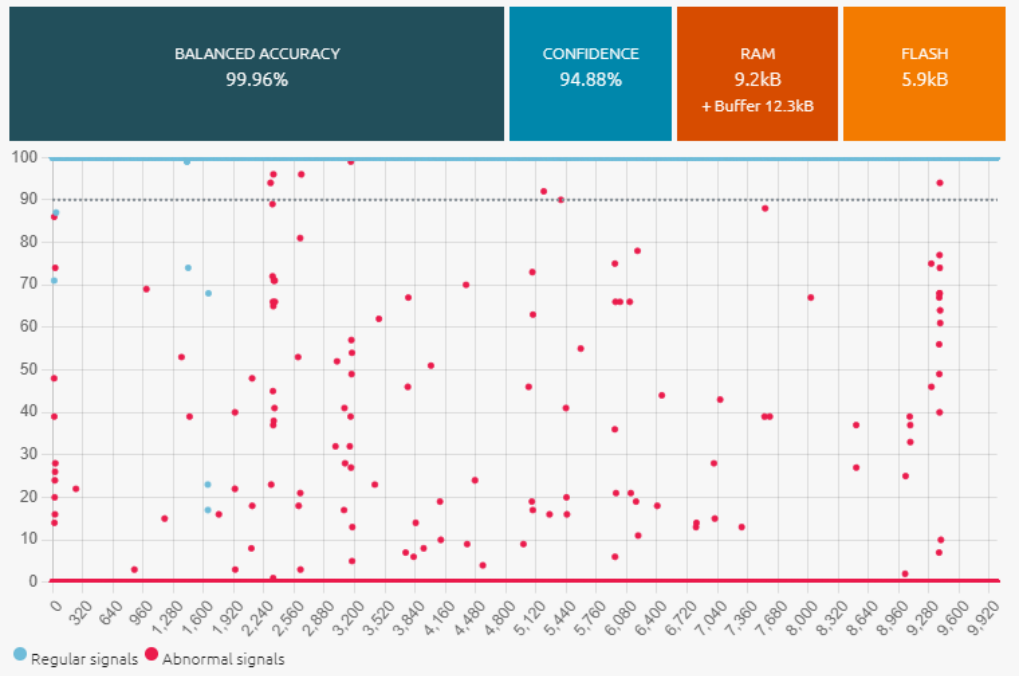
accuracy on, sitä paremmin tekoälykirjasto toimii. Balanced accuracy on tärkein tekoälykirjaston toiminnasta kertovista indikaattoreista. Sen painoarvo parhaan mahdollisen tekoälykirjaston toiminnan indikoinnissa on 90 %. [21.]

Confidence-indikaattori kertoo tekoälykirjaston luottamusvälistä eli sen kyvystä erotella signaalit toisistaan ilman päällekkäisyyksiä. Mitä korkeampi luottamusvälin prosenttiluku on, sitä paremmin se erottelee signaalit toisistaan ilman päällekkäisyyksiä ja epäselvyyksiä. Luottamusvälin painoarvo parhaan mahdollisen tekoälykirjaston toiminnan indikoinnissa on 9 %. [21.]

RAM indikoi maksimi muistin määrää, jota tekoälykirjasto käyttää kun tekoälykirjastoon pohjautuva tekoälymalli on integroituna kohdealustaan. Flash indikoi kirjaston käyttämää flahs-muistin määrää sen ollessa integroituna kohdealustaan. Yhdessä RAM ja Flash painoarvo tekoälykirjaston toiminnassa on 1 %. [21.]

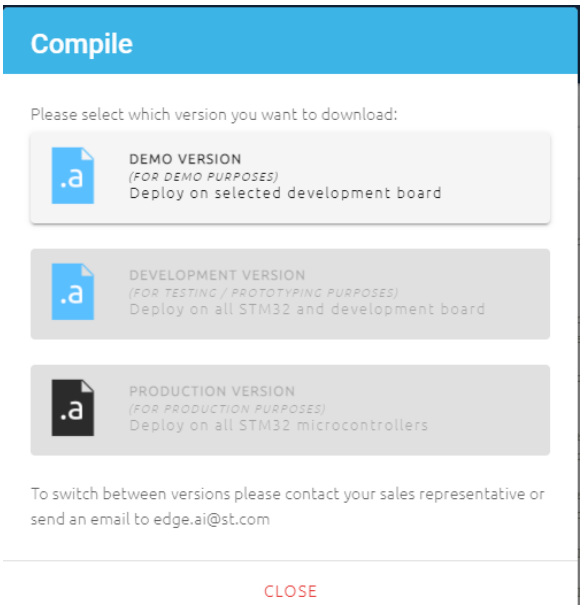
Parhaan tuloksen saavuttamiseksi Benchmarkin tulisi siis antaa prosessoida, kunnes kuvassa 33 esitetty ikkuna 1 on päässyt 100 %:n asti. Tässä prosessissa voi kestää puolesta tunnista useaan tuntiin, riippuen käytettävän tietokoneen laskennallisesta tehosta.

Benchmark-prosesseja tehtiin kaksi. Ensimmäinen prosesseista epäonnistui, koska verkkoyhteys katkesi ja tehty benchmark hävisi. Tässä benchmark-prosessissa oli kestänyt kolme viikkoa. Käytettävässä tietokoneessa oli Intel i5 prosessori, joka on saattanut vaikuttaa huomattavasti prosessin hitauteen. Aikarajoituksista johtuen toista benchmark-prosessia ei ehditty käymään 100%:n asti, vaan prosessi täytyi pysäyttää kesken. Benchmarkin aikana NE AI Studio testasi lähes 300 tekoälykirjastoa. Prosessin aikana NE AI Studio löysi käytettyä dataa varten 15 eri tekoälykirjastoa. Paras löydetty tekoälykirjasto saavutti 99,96 %:n Balanced accuracyn ja 94,88 %:n Confidencen. Tekoälykirjasto vaatii 9,2 kB RAM ja lisäksi 12,3 kB bufferointia varten sekä 5,9 kB FLASH-muistia. Tulokset on esitetty myös kuvassa 35.



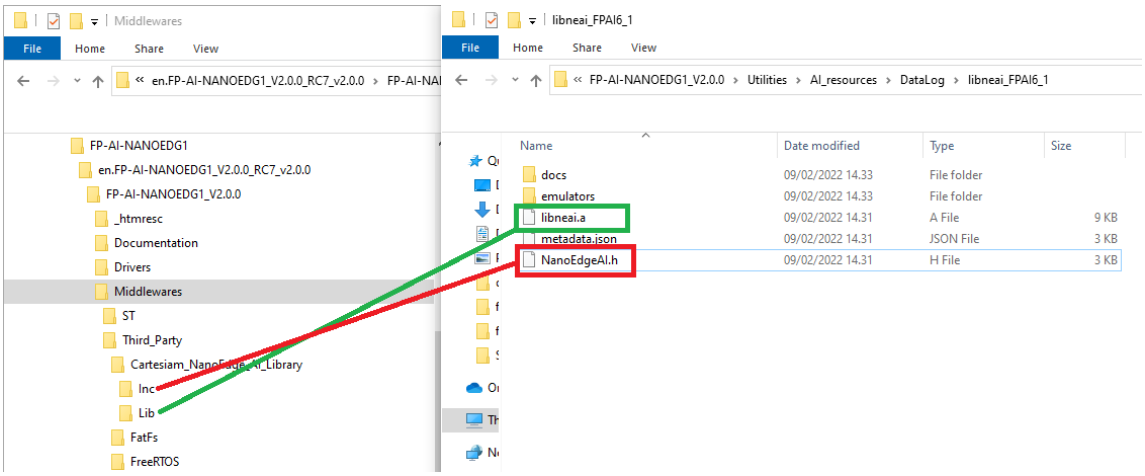
Kuva 35. Benchmark-prosessin tulokset.

NanoEdge AI Studiossa seuraavaksi siirryttiin Deployment osioon, jonka tehtävä on rakentaa paras löydetty tekoälykirjasto paketiksi, joka voidaan integroida haluttuun projektiin. NanoEdge AI Studio valitsee rakennettavaksi aina parhaan löytämänsä tekoälykirjaston ja luo helloworld-pohjan helpottaakseen ohjelmoinnin aloittamista. Deployment-ikkunassa otsikon Compilation Flags alla laitettiin raksi kohtaan ”Float abi” projektin käyttöohjeiden mukaan. Tämän jälkeen painettiin Compile Library painiketta, josta tuli kuvan 36 popup.



Kuva 36. Compilation vaihtoehdot.

Koska NanoEdge AI Studiosta oli käytössä demo-versio, vain ensimmäinen painikkeista oli käytettävissä. Painamalla DEMO VERSION-paniketta NanoEdge AI Studio loi zip-paketin tekoälykirjastosta projektin hakemistoon. Generoitu zip-paketti purettiin hakemistoonsa. Tekoälykirjaston kansiosta kopioitiin tiedostot libneai.a ja NanoEdgeAI.h kansioihin Inc ja Lib kuvan 37 osoittamalla tavalla.



Kuva 37. Kirjastojen sijainnit.

6.2.4 Projektin kääntäminen

Projekti käännettiin STEVAL-STWINKT1B-kehitysalustalle STM32CubeIDE:llä. Käännöstä varten täytyi ensiksi luoda yhteys kehitysalustaan TeraTerm-ohjelmalla, kuten kuvissa 28 ja 29 on esitetty. Tämän jälkeen .project-niminen STM32CubeIDE-tiedosto avattiin FP-AI-NANOEDG1-V2.0.0-kunnonvalvontafunktiopaketin kansiorakenteesta seuraavan kansiopolun takaa:

/FP_AI_NANOEDG1_V2.0.0/Projects/STM32L4R9ZI-STWIN/Applications/NanoEdgeConcole/STM32CubeIDE/

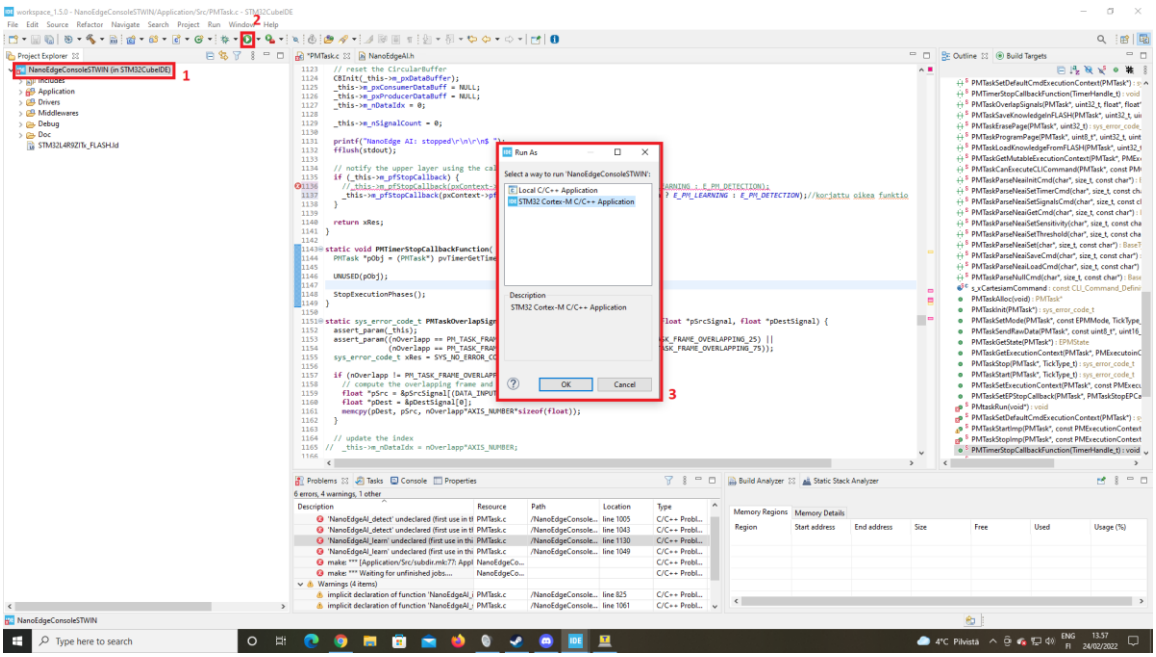
Jotta luotu tekoälykirjasto saatiin ajettua kehitysalustalle, täytyi tehdä muutoksia STM32CubeIDE -projektiin kuuluvaan PMTask.c-tiedostoon, sillä siinä olevat NanoEdge AI-funktiot eivät olleet yhteensopivia NanoEdgeAI.h-kirjastotiedostossa olevien NanoEdge AI-funktioiden kanssa. NanoEdgeAI.h-kirjastotiedosto on sama, mikä kopioitiin edellisen kappaleen kuvassa 37. Tehdyt muutokset on esitetty taulukossa 3.

Rivi	PMTask.c	Korjaus
825	NanoEdgeAI.initialize();	neai_anomalydetection_init();
945	NanoEdgeAI.initialize();	neai_anomalydetection_init();
1007	if (xContext.pfNeaiMode == NanoEdgeAI_detect;	if(xContext.pfNeaiMode == neai_anomalydetection_detect){
1046	_this->m_xDtectContext.pfNeaiMode = NanoEdgeAI_detect;	_this->m_xDetectContext.pfNeaiMode = neai_anomalydetection_detect;
1054	-this->m_xLearnContext.pfNeaiMode = NanoEdgeAI_learn;	-this->m_LearnContext.pfNeaiMode = neai_anomalydetection_learn;
1067	NanoEdgeAI_set_sensitivity(pxContext->xContext.fSensitivity);	neai_anomalydetection_set_sensitivity (pxContext->xContext.fSensitivity);
1137	_this->m_pfStopCallback(pxContext-> pfNeaiMode == NanoEdgeAI_learn ? E_PM_LEARNING : E_PM_DETECTION);	_this->m_pfStopCallback(pxContext-> pfNeaiMode == neai_anomalydetection_learn ? E_PM_LEARNING : E_PM_DETECTION);

Taulukko 3. PMTask.c-tiedostoon tehdyt muutokset.

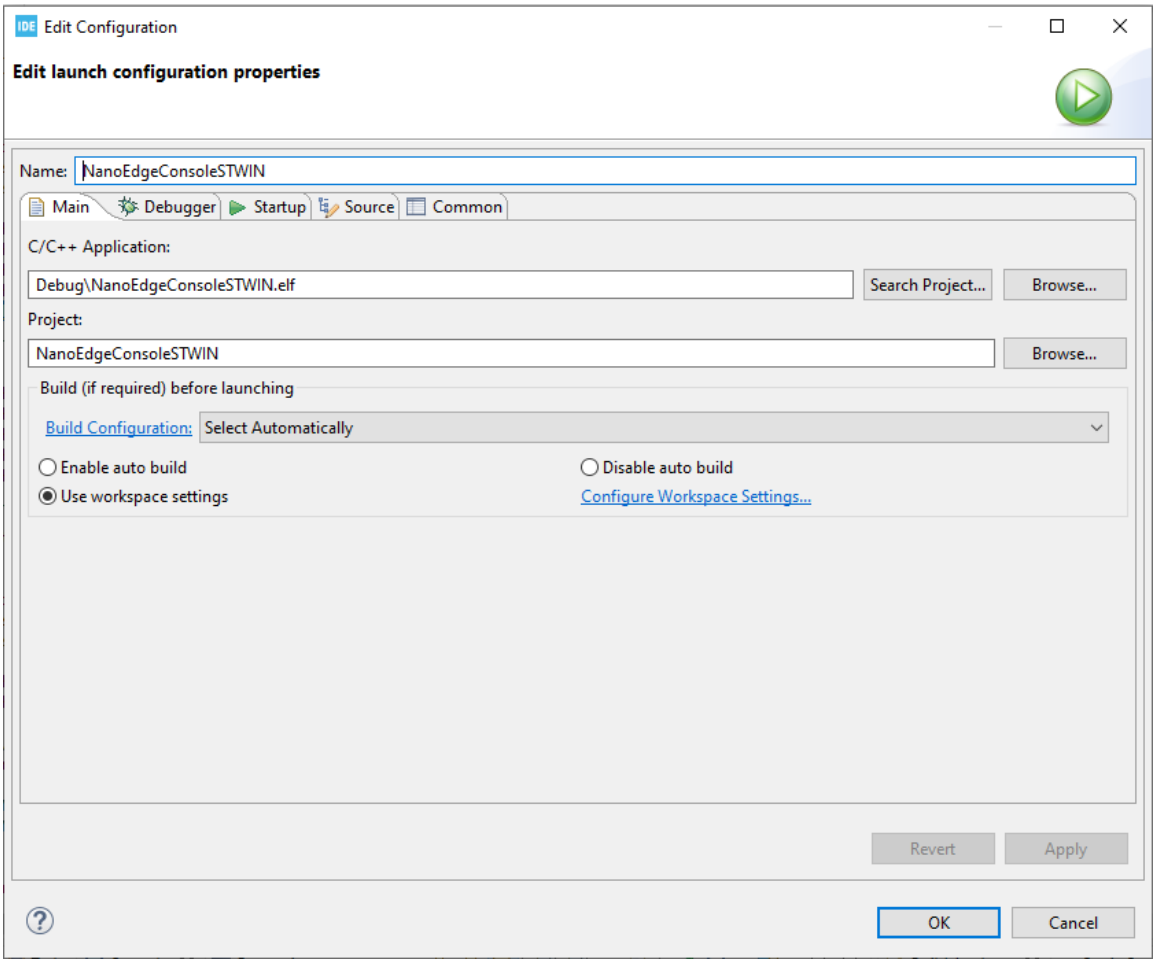
Kun muutokset PMTask.c-tiedoston koodiin oli tehty, ajettiin tekoälykirjasto kehitysalustalle kuvan 38 osoittamalla tavalla. Ensiksi varmistettiin, että NanoEdgeConsoleSTWIN(inSTM32CubeIDE) (kohta 1), oli valittuna. Sitten painettiin RUN-

painiketta (kohta 2). Tämän jälkeen popup-ikkunasta valittiin STM32 Cortex-M C/C++ Application (kohta 3) ja painettiin OK.



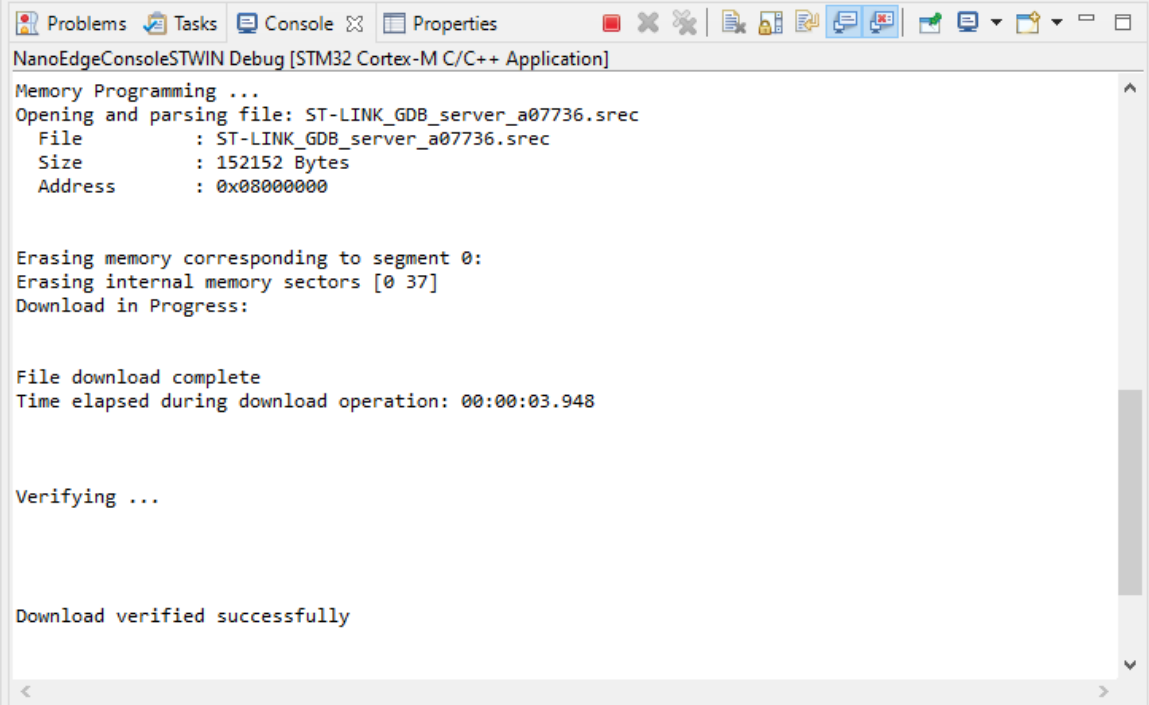
Kuva 38. Tekoälykirjastojen ajaminen kehitysalustalle, osa 1.

Kuvassa 39 on esitetty Edit Configuration-ikkuna, joka tuli esille kun OK-painiketta oli painettu. Kuvassa on projektin asetukset. Käännös aloitettiin painamalla OK.



Kuva 39. Edit Configuration -ikkuna.

Käännösprosessia voidaan seurata STM32CubeIDE:n alareunasta löytyvästä Console-ikkunasta. Konsoliin ilmaantuu viestejä käännösprosessista, kuten esimerkiksi jos tulee vastaan virheitä tai varoituksia sekä käännöksen läpikäydyt vaiheet. Kun käännös oli valmis ja ajettu kehitysalustaan, konsoliin ilmaantui kuvan 40 ”Download verified successfully”-viesti, joka kertoi siitä, että tekoälykirjastot oli käytettävissä STEVAL-STWINKT1B-kehitysalustassa.



Kuva 40. Onnistunut käännös.

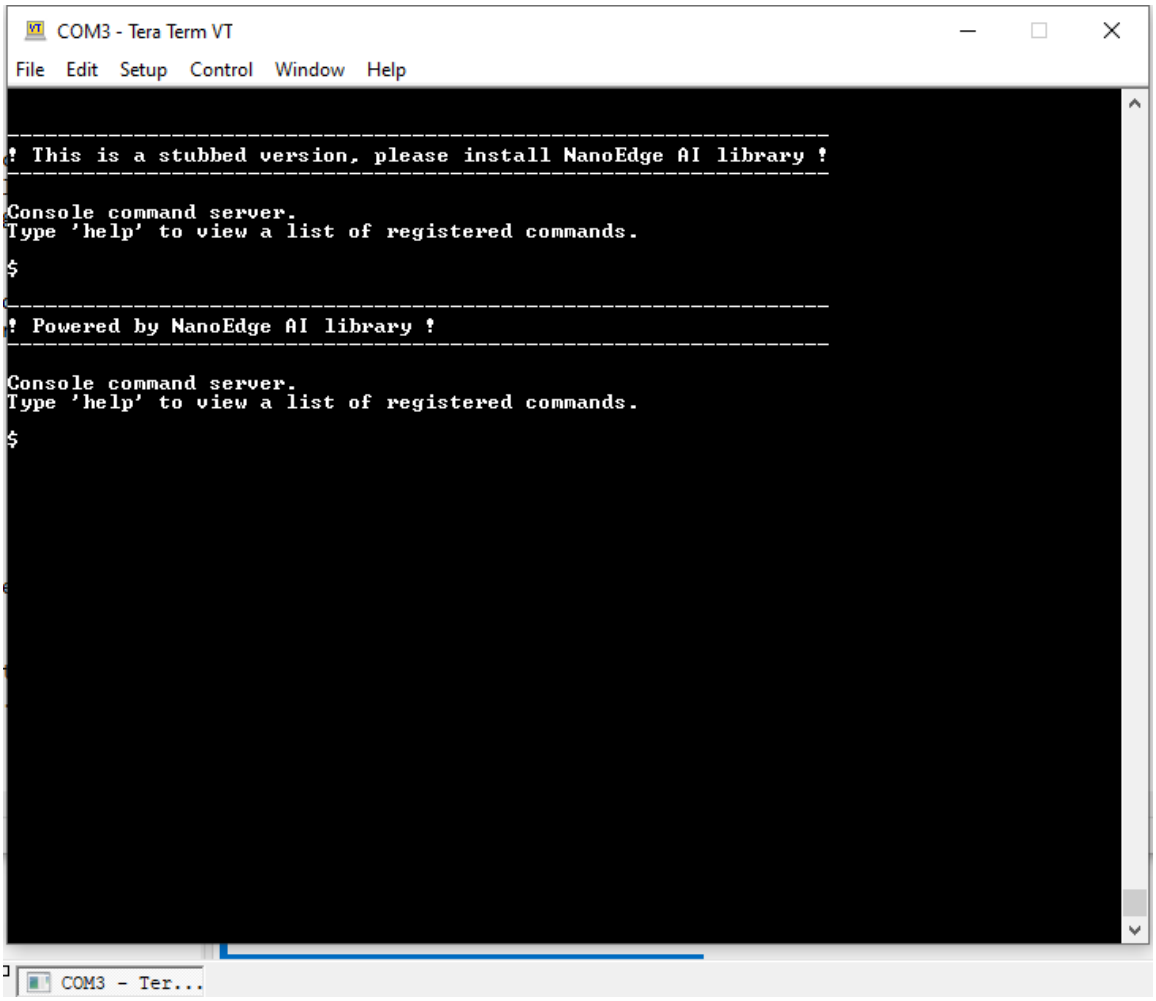
Kuvassa 41 on esitetty TeraTerm-konsoliin ilmaantunut viesti käännöksen onnistumisesta. Ennen oman tekoälykirjaston asentamista TeraTerm-konsolissa lukee:

! This is a stubbed version, please install NanoEdge AI library !

Kun NanoEdge AI-kirjasto on asennettu onnistuneesti kehitysalustaan, tulee TeraTerm-konsoliin ilmaantua viesti:

! Powered by NanoEdge AI library !

Tämän jälkeen tekoälykirjasto on käytettävissä STEVAL-STWINKT1B-kehitysalustassa koulutusta ja vikojen havaitsemista varten.

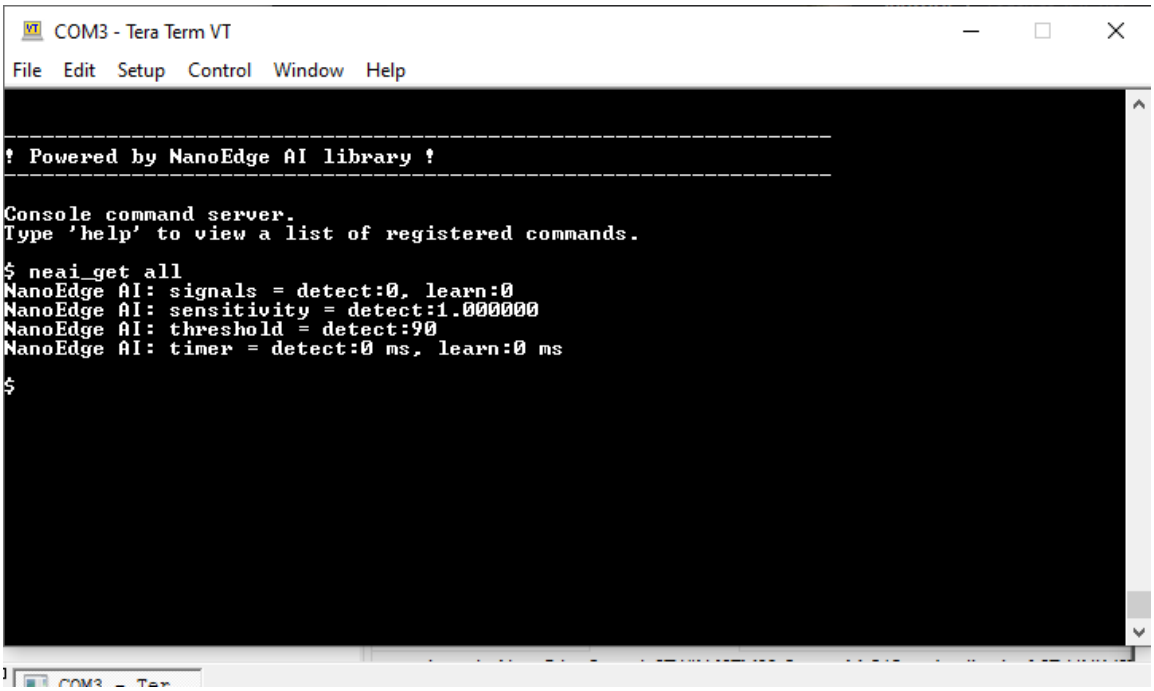


Kuva 41. TeraTerm-ikkuna, kun tekoälykirjastot on ajettu kehitysalustalle.

6.2.5 Tekoälyn koulutus ja testaus

Ennen kuin NanoEdge tekoälykirjastoa voidaan hyödyntää vikojen ja epänormaalien signaalien havaitsemiseen, tulee STEVAL-STWINKT1B-kehitysalustassa kouluttaa tekoäly. Tekoälyä kouluttaessa on oleellista käyttää signaaleja, jotka ovat mahdollisimman samanlaisia kuin datalog-prosessissa kerätyt normaaleja signaaleja edustavat signaalit. NanoEdge AI Studiolla haettu tekoälykirjasto perustuu nimenomaan datalog-prosessissa kerättyihin normaaleja tiloja edustaviin signaaleihin, joten tämän takia on oleellista, että NanoEdge AI kirjastoon perustuvassa tekoälysovelluksessa käytetään signaaleja, jotka ovat mahdollisimman lähellä NanoEdge AI Studion hakukoneeseen syötettyjä normaaleja tiloja edustavia signaaleja. [21; 39.]

Ennen tekoälyn kouluttamista TeraTerm -ohjelmassa voidaan säästää NanoEdge AI kirjaston hyperparametreja [39]. Käytettävät hyperparametrit saadaan näkyviin kirjoittamalla TeraTerm-terminaaliin komento neai_get all. Komennolla saadut hyperparametrit on esitetty kuvassa 42.



Kuva 42. NanoEdge AI käytettävät hyperparametrit.

Säädettäviä hyperparametreja learn ja detect prosesseille ovat:

- Signals, sekä detect että learn prosesseille
- Sensitivity, vain detect prosessille
- Threshold, vain detect prosessille
- Timer, sekä detect että learn prosesseille

Signals ja timer hyperparametreilla voidaan säätää erikseen koulutus- ja vian havaitsemiseprosessien kestot [39]. Hyperparametrit voidaan asettaa komennolla neai_set. Esimerkiksi, jos halutaan koulutusprosessin käyvän läpi 100 signaalia, terminaaliin kirjoitetaan seuraava komento:

$ neai_set signals 100

Tällöin sekä learn että detect prosessien aikana käydään läpi 100 signaalia, jonka jälkeen käynnistetty prosessi päättyy. Se, kuinka monta signaalia koulutusvaiheessa tulisi vähintään käydä läpi riippuu siitä, kuinka monta koulutusiteraatiota NanoEdge AI Studiossa saadussa benchmarkissa suositeltiin minimikoulutusiteraatioiksi yhden signaalin oppimista varten. Koulutusvaiheessa käytettyjä signaaleja tulisi olla 3-10-kertainen määrä verrattuna benchmarkista saattuu minimikoulutusiteraatiomäärään. Esimerkiksi, jos koulutusiteraatioita signaalia varten saatiin benchmark-prosessin aikana 20, tulisi signals-hyperparametriksi minimissään asettaa 60 signaalia, mutta mielellään lähemmäs 200 signaalia. [39.]

Mikäli tekoäly halutaan mieluummin kouluttaa ajan perusteella, voidaan kesto learn ja detect-prosesseille määrittää timer-hyperparametrin kautta. Timer-hyperparametrilla säädetään prosessin kesto millisekunteina, joten mikä tahansa kesto täytyy muuttaa millisekunneiksi ennen kuin timer-hyperparametri säädetään. [39.]

Esimerkiksi jos halutaan learn prosessin kestävän 10 minuuttia, timer hyperparametri säädettäisiin seuraavalla tavalla:

$ neai_set timer 600000

Threshold-hyperparametri on kynnysarvo, jota käytetään detect-prosessissa epänormaalien signaalien raportoimiseen. Jos havaitun signaalin samankaltaisuus kynnysarvoon on alle asetetun arvon, esimerkiksi 90, signaali tulee ilmoitetuksi epänormaalina signaalina. Esimerikisi kynnysarvon ollessa 90, jos tekoäly havaitsee signaalin jonka samankaltaisuuden arvo on 80, havaittu signaali tulee ilmoitetuksi epänormaalina signaalina. Jos signaalin samankaltaisuus on 90 tai päälle 90, ilmoitusta ei tule. [39.]

Threshold arvoa voidaan muuttaa esimerkiksi arvoon 95 seuraavalla komennolla:

$ neai_set threshold 95

Sensitivity-hyperparametria käytetään painoarvona detect-prosessissa. Perusarvo tälle herkkyydelle on 1. Mikäli arvoa korotetaan, signaalien sovitus tehdään tarkemmin. Mikäli arvoa lasketaan, signaalien sovitusta ei tehdä yhtä tarkasti jolloin samankaltaisia signaaleja löydetään todennäköisesti enemmän. [39.]

Mikäli hyperparametreja ei haluta säätää, voidaan tekoälyn koulutus aloittaa kirjottamalla TeraTerm-konsoliin seuraava käsky:

$ start neai_learn

Prosessi voidaan lopettaa esimerkiksi painamalla näppäimistön ESC-painiketta tai kirjoittamalla konsoliin stop ja painamalla ENTER-painiketta. Jos signals tai timer hyperparametreja on säädetty, koulutusprosessi on käynnissä määritetyn keston ajan jonka jälkeen se pysähtyy. [39.]

Kun tekoäly on oppinut jonkin signaalin, terminaaliin tulostuu viesti:

{"signal": <arvo>, "status": succes}

Kun learn prosessi on käyty läypi, tekoälyn toimivuutta voidaan kehitysalustalla testata tekoälyn toimintaa. Detect-prosessin hyperparametreja voidaan säätää vielä tässä vaiheessa. [39.]

Prosessi voidaan aloittaa kirjoittamalla seuraava komento TeraTerm-konsoliin:

$ start neai_detect

Jos prosessille on säädetty kesto, se on käynnissä joko siihen asti että käyttäjä pysäyttää prosessin tai kunnes prosessi on ollut käynnissä määritetyn keston ajan. Mikäli kestoa ei ole ennalta määritetty, prosessi on käynnissä kunnes käyttäjä sen pysäyttää, joko painamalla ESC-painiketta tai kirjoittamalla terminaaliin stop ja painamalla ENTER-painiketta. [39.]

Detect-prosessin aikana terminaaliin tulee ilmoitus joka kerta kun tekoäly havaitsee epänormaalin signaalin tai vian. Tekoäly tällöin ilmoittaa kuinka mones signaali on kyseessä, kuinka samankaltainen se on normaalien signaalien kanssa sekä sen, että kyseessä on epänormaali signaali. [39.]

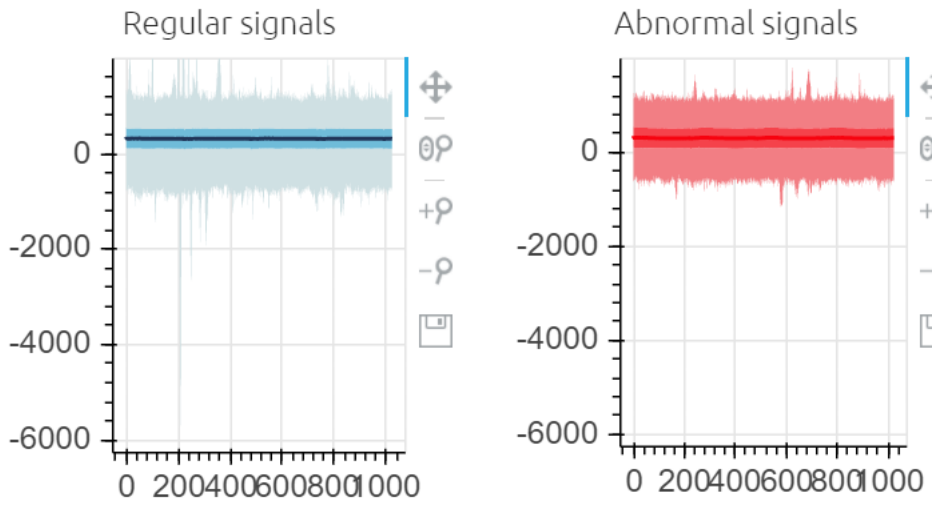
On oleellista tietää, että joka kerta kun STEVAL-STWINKT1B -kehitysalusta resetoidaan, johtui tämä RESET-painikkeen painamisesta, yhteyden menetyksestä tai virran katkaisusta, kaikki tekoälykirjaston oppima data häviää. Tämä tarkoittaa sitä, että jokaisen käynnistyksen ja yhteyden luonnin yhteydessä tekoäly on aina koulutettava uudelleen, mikäli sitä halutaan käyttää vikojen havaitsemiseen. [39.]

6.3 Tulosten käsittely

STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B -kehitysalustan koneoppimisydintä testattiin kappaleessa 6.2 kuvatuin menetelmin ja keinoin. Projekti aloitettiin SD-kortin formatoinnilla, jonka onnistuttua siirryttiin kappaleessa 6.2.2 kuvattuun anturidatan keräykseen datalog-prosessilla.

Datalog-prosessin aikana kerättiin kaikenkaikkiaan kuusi erilaista datalog-tiedostoa. Neljä ensimämistä dataloia kerättiin pesukoneen Whirlpool 6th Sense käytöstä. Datalogit DL_001 ja DL_003 edustivat pesukonetta sen toimiessa ”normaalisti”. Nämä datalogit kerättiin niin, että kehitysalusta oli asetettuna pesukoneen päälle ja pesukoneessa käytettiin datalog-prosessin aikana 15 minuutin ohjelmaa minimi rummun kierrosnopeudella (0 rpm). Datalogit DL_002 ja DL_004 edustivat pesukonetta sen toimiessa ”epänormaalisti”. Nämä datalogit kerättiin muuten samoin menetelmin kuin DL_001 ja DL_003, mutta rummun kierrosnopeudeksi asetettiin pesukoneen maksimi (1600 rpm).

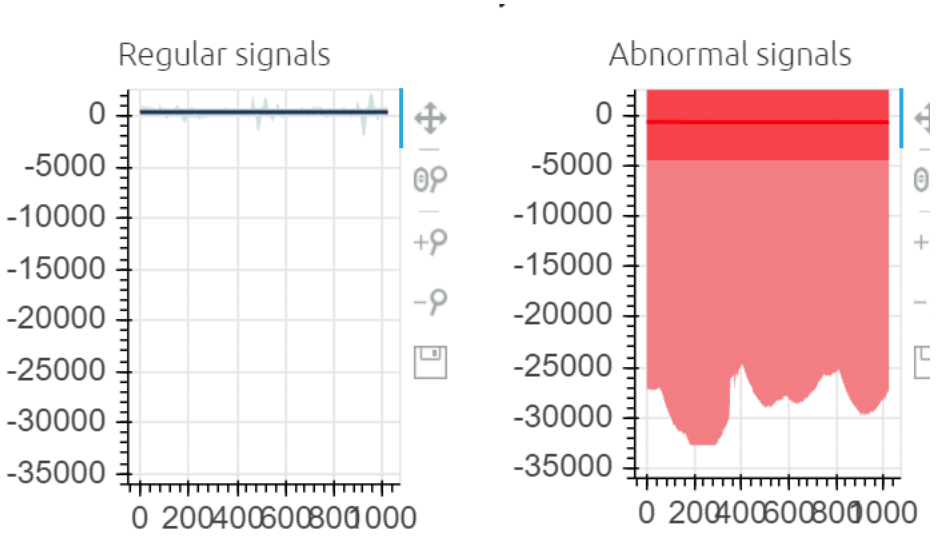
Saadut datalogit muunnettiin ohjeiden mukaisesti normalSegments.csv- ja abnormalSegments.csv-tiedostoiksi. Nämä tiedostot sitten syötettiin kappaleessa 6.2.3 esitetyllä tavalla NanoEdge AI Studion hakukoneeseen. Ennen benchmark-prosessin aloitusta havaittiin, että normalSegments.csv- ja abnormalSegments.csv-tiedostoista saadut signaalit olivat hyvin samankaltaisia, kuten kuvassa 43 on esitetty.



Kuva 43. Ensimmäisten datalogien signaalien vertailu.

Jos NanoEdge AI Studioon syötetyt normaalia tilaa ja epänormaalia tilaa edustavat signaalit ovat liian samankaltaisia, on riski että NanoEdge AI Studio ei välttämättä löydä tekoälykirjastoa, joka kykenisi tarpeeksi hyvin erottelemaan normaalit ja epänormaalit signaalit toisistaan kohdealustassa sen jälkeen, kun koulutusprosessi on käyty läpi itse kohdealustan sisällä.

Koska neljä ensimmäistä kerättyä datalogia eivät välttämättä olisi soveltuneet STEVAL-STWINKT1B-kehitysalustan koneoppimisytimen testausta varten, päädyttiin keräämään kaksi uutta datalogia: DL_005 ja DL_006. DL_005 kerättiin kehitysalustan ollessa häiritsemättä paikoillaan pöydän päällä 15 minuuttia, edustaen näin ”normaalin” tilan signaaleja. DL_006 kerättiin kehitysalustan ollessa 15 minuutin ajan mekaanisen tärinän vaikutuksen alla, edustaen näin ”epänormaalin” tilan signaaleja. Näistä datalogeista generoitiin normalSegments.csv- ja abnormalSegments.csv-tiedostot, jotka sitten syötettiin NanoEdge AI Studion hakukoneeseen. Ennen benchmark-prosessia saadut signaaliesikatselut on esitetty kuvassa 44. Koska signaalit olivat riittävän poikkeavat toisistaan, päätettiin signaaleja käyttää kontekstuaalisena datana NanoEdge AI Studion hakukoneessa eli benchmark-prosessissa.



Kuva 44. DL_005 (vasemmalla) ja DL_006 (oikealla) signaalien vertailu.

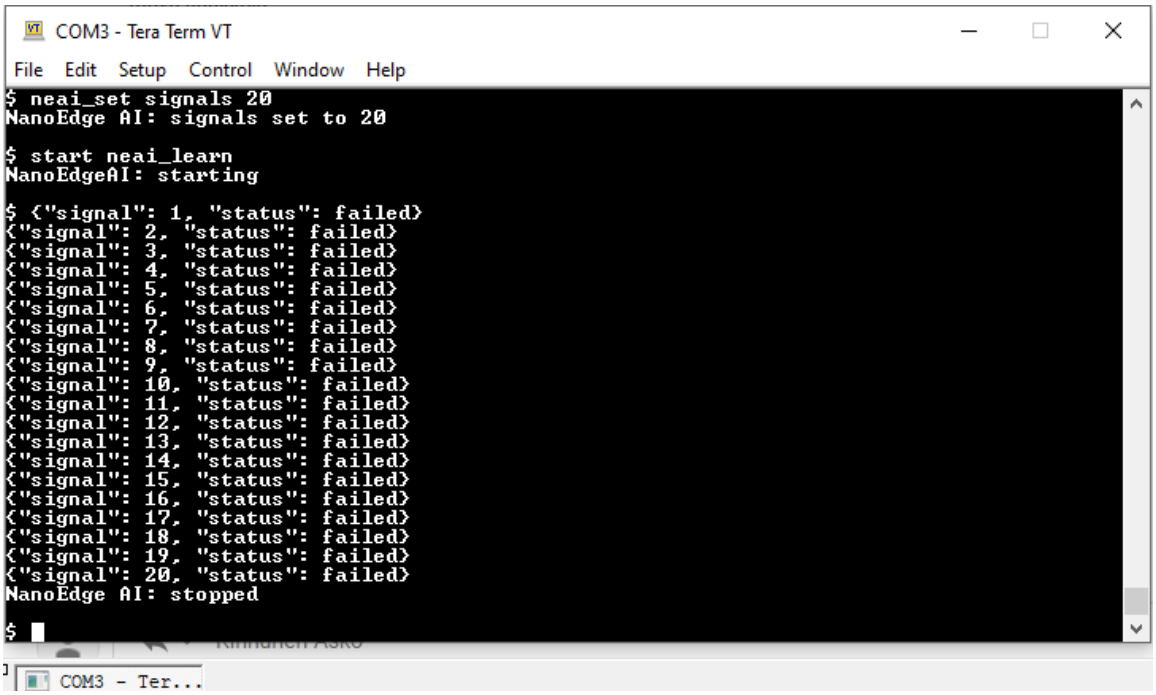
FP-AI-NANOEDG1-V2.0.0 käyttöohjeissa sekä NanoEdge AI Studion käyttöohjeissa ilmaistiin, että benchmark prosessi saattaisi kestää puolesta tunnista muutamaan tuntiin, jonka aikana prosessin kuuluisi käydä 0%:sta 100%:iin. Benchmark-prosesseja tehtiin kaksi, sillä ensimmäinen epäonnistui. Ensimmäinen benchmark-prosessi oli käynnissä noin kolmen viikon ajan, jolloin se oli päässyt 99%:iin asti ennen kuin prosessi epäonnistui. Epäonnistuminen johtui katkoksesta

verkkoyhteydessä prosessin päätösvaiheen aikana, josta johtuen benchmark-prosessista ei saatu käyttökelpoista dataa. On siis erittäin tärkeää, että benchmark-prosessin aikana on käytössä toimiva verkkoyhteys. Tarvittaessa prosessin voi pistää tauolle, mutta kun prosessi halutaan pysäyttää, tarvitaan toimiva verkkoyhteys.

Aikarajoituksista johtuen toiselle benchmark-prosessille ei voitu antaa yhtä paljon aikaa käydä 100%:iin asti. Toinen yritys suoritettiin myös toisella tietokoneella, jota ei kyetty pitämään vain ja ainoastaan benchmark-prosessin käytössä useita päiviä putkeen vuorokauden ympäri. NanoEdge AI Studio ja sen benchmark-prosessi kuormittavat tietokoneen prosessoria merkittävästi pitäen sen koko prosessin ajan päälle 85%:n kuormituksella tehden merkittävän usein 100%:n kuormituspiikkejä. Prosessi pidettiin käynnissä siihen asti, että päästiin kuvassa 35 esitettyihin tuloksiin. Kuvassa 35 esitetty erottelukyky saadulle tekoälykirjastolle vaikutti riittävältä. On huomattavaa tosin, että benchmark-prosessi ei ollut käynnissä 100%:n asti, aiheuttaen ongelmia projektissa myöhemmin.

FP-AI-NANOEDG1-V2.0.0-funktiopaketti on suunniteltu toimimaan sellaisenaan. Tällä tarkoitetaan sitä, että funktiopaketin käyttöohjeissa annetaan ymmärtää, että projektin tulisi kääntyä sellaisenaan STM32CubeIDE:llä kohdealustalle ilman muutoksia. Näin ei kuitenkaan ollut projektia käännettäessä, vaan projektin PMTask.c-koodiin täytyi tehdä taulukossa 3 esitetyt muutokset. Tämä ei sinänsä ole ongelma, se on vain pidettävä mielessä tulevaisuudessa, kun tekoälykirjasto halutaan ajaa kohdealustalle onnistuneesti. Kuvassa 41 on edelleen esitettynä TeraTerm CLI:n näkymä, kun tekoälykirjasto on ajettu onnistuneesti kohdealustaan.

Seuraava vaihe oli kouluttaa tekoäly kohdealustassa kappaleen 6.2.5 mukaisesti. Tämä ei valitettavasti onnistunut suunnitelmien mukaisesti. Ajettaessa koulutusscriptiä, TeraTerm-konsoliin ilmeistyi kuvan 45 mukainen viesti.



Kuva 45. Epäonnistunut start neai_learn script.

Syy tälle epäonnistumiselle ei ole varma. Ottaen huomioon, että FP-AI-NANOEDG1-V2.0.0-kunnonvalvontafunktiopaketin testauksessa lähestulkoon ainoa ongelmia aiheuttanut vaihe oli tekoälykirjaston etsiminen NanoEdge AI Studiolla, joka jäi aikarajoitusten takia keskeneräiseksi, voidaan vahvasti epäillä epäonnistumisen johtuneen tästä. Itse projektin tiedostoista ei löytynyt muita konflikteja, kuin ne mitkä korjattiin taulukon 3 mukaisesti, joten epäonnistuminen tuskin johtuu niistä. Aikarajoituksista johtuen tätä ei valitettavasti voida testata uudelleen taikka varmistaa, että vika johtui nimenomaan NanoEdge AI Studion benchmark-prosessista, joten ongelman aiheuttaja jää spekulaatioksi.

Koska tekoälyn koulutus STEVAL-STWINKT1B-kehitysalustassa ei onnistunut, ei tekoälyn testaaminenkaan luonnollisesti onnistunut kehitysalustalla, sillä sillä tekoälyn testaaminen vaatisi kehitysalustassa koulutetun tekoälyn.

7 Pohdinta

Opinnäytetyön yhtenä tavoitteena oli kerryttää kattava teoriapohja opinnäytetyölle keskeisistä aihepiireistä. Näitä aiheita olivat tekoäly, koneoppiminen, konenäkö, ennakoiva vikadiagnostiikka, teollisuuden älykäs kunnonvalvonta sekä Industry 4.0. Teoriapohjan kasauksessa tähdättiin siihen, että kaikista aihepiireistä olisi selostettuna peruskäsitteet sekä annettu havainnollistavia esimerkkejä, mitä selostuksella ollaan tarkoitettu. Teoriapohjan tulisi olla sellaisessa muodossa, että sitä voitaisiin tulevaisuudessa käyttää kurssimateriaalina.

Opinnäytetyön toisena tavoitteena oli perehtyä STEVAL SensorTile Wireless Industrial Node STEVAL-SWINKT1B -kehitysalustan ominaisuuksiin, antureihin, toimintaan ja sen koeoppimisytimeen. Insinöörityön tavoitteena oli laatia materiaalia kehitysalustasta, jossa olisi esiteltynä sen ominaisuudet, sulautetut anturit, käyttöohjeet eräille sen perussovelluksista sekä syvempi perehdytys kehitysalustan koneoppimisytimeen sekä sen koulutukseen ja käyttöön esimerkin avulla. Tarkoitus oli siis testata STEVAL-STWINKT1B:n koneoppimisytimen käyttöä käytännön olosuhteissa.

Opinnäytetyön tuloksena vaaditusta teoriaosuudesta saatiin laajahko kokonaisuus, jossa työlle keskeisten aihepiirien peruskäsitteet esiteltiin ja selostettiin helposti ymmärrettävällä tavalla. Tekoälystä, koneoppimisesta sekä konenäöstä kerrottiin aihepiireille keskeinen terminologia, miten ne liittyvät toisiinsa ja millaisia menetelmiä niissä käytetään. Esimerkiksi tekoälystä ja sille tyypillisimmistä neuroverkkotyypeistä annettiin käytännön esimerkkejä: mitä ne ovat, miten ne toimivat ja mihin niitä tyypillisesti käytetään. Ennakoiva vikadiagnostiikka sekä teollisuuden älykäs kunnonvalvonta käytiin läpi kokonaisuutena, jossa esiteltiin mitä termeillä tarkoitetaan, miten vikadiagnostiikka ja kunnonvalvonta on ennen toiminut teollisessa ympäristössä, miksi ne olisivat tärkeitä teollisessa tuotannossa tässä kehittyvässä maailmassa, miten niillä voitaisiin tehostaa tuotantoa ja vähentää kustannuksia ja mitä mahdollisuuksia ne voisivat tuoda tulevaisuudessa. Opinnäytetyössä esiteltiin myös käsite Industry 4.0, mitä sillä tarkoitetaan, mikä on sen merkitys ja mihin sillä tähdätään sekä minkälaisia mahdollisuuksia Industry 4.0 sekä sen älykäs teollisuus voivat tuoda mukanaan.

Itse insinöörityönä esiteltiin STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B -kehitysalusta. Työssä esiteltiin kehitysalustan sulautetut anturit sekä muut ominaisuudet ja

kehitysalustapaketin osat. Kahden laiteohjelmistoesimerkin, Serial_DataLog- sekä BLE_SampleApp-esimerkkiohjelmien, avulla esiteltiin miten laitteeseen muodostetaan yhteys, kuinka laitteen keräämään ympäristöantureiden muodostamaan dataan päästään käsiksi sekä CLI:n että mobiilisovelluksen katta. Tuloksena saatiin esittely kehitysalustan ominaisuuksista, sulautetuista antureista ja muutamasta perussovelluksesta, jolla voidaan testata ylipäätään laitteen toimintaa.

Kehitysalustan koneoppimisytimen toimintaan työssä perehdyttiin STMicroelectronics FP-AI-NANOEDG1-V2.0.0-kunnonvalvontafunktiopaketin avulla. Työssä kerrottiin funktiopaketin ominaisuuksista, miten se toimii ja mitä sillä voidaan tehdä sekä käytiin vaihe vaiheelta läpi, kuinka funktiopaketin avulla STEVAL-STWINKT1B-kehitysalustassa voidaan kerätä dataa tekoälyn koulutusta varten, etsiä sopiva tekoälykirjasto joka sopisi käytettyyn dataan, miten tekoälykirjasto saadaan ajettua kehitysalustalle sekä miten tekoäly sitten koulutetaan kohdealustassa ja testataan sen kunnonvalvontaominaisuuksia. Kunnonvalvontafunktiopaketin käytölle laadittiin siis ohjeet ja pyrittiin testaamaan ja havainnollistamaan, kuinka funktiopaketti toimii käytännössä. Tuloksena saatiin käyttöohjeet sekä esimerkki yhdestä tavasta hyödyntää koneoppimisydintä.

Mielestäni opinnäytetyön teoriaosuudesta saatiin tarpeeksi kattava, jotta sitä voidaan hyödyntää kurssimateriaalina. Aihepiirien peruskäsitteet on selitetty läpi selkeästi esimerkkien avulla.

Opinnäytetyössä mielestäni onnistuttiin myös esittelemään STEVAL-STWINKT1B-kehitysalustapaketti, sen osat, sulautetut anturit sekä miten se toimii selkeästi ja havainnollisesti. Laitteen käyttöönotto esitettiin havainnollistamalla kuvamateriaalin avulla sellaisia asioita kuten yhteyden muodostus, STM32CubeProgrammer-ohjelman käyttö sekä miten Serial_DataLog- ja BLE_SampleApp-laiteohjelmistoja käytetään, miten ne toimivat ja minkälaista tietoa niiden kautta voidaan saada kehitysalustan ympäristöantureista, pilvipalveluista sekä virransäästöominaisuuksista. Laiteohjelmistojen käyttöönotto lienee hyvä tapa testata, onko käytössä oleva kehitysalustayksilö toimintakunnossa ja antaa käsityksen siitä, minkälaista dataa laitteelta voidaan esimerkiksi saada.

Opinnäytetyössä haastavimmaksi osuudeksi osoittautui FP-AI-NANOEDG1-kunnonvalvontafunktiopaketin testaus. Alkuvaiheet funktiopaketin testauksessa onnistuivat hyvin muutaman yrityksen jälkeen, eli yhteyden muodostus, datalog-prosessien läpikäynti sekä

saatujen datalogien muuttaminen NanoEdge AI Studion kanssa yhteensopivaan muotoon, eli normalSegments.csv- ja abnormalSegments.csv-tiedostoiksi. Ongelmallisin osuus funktiopaketin testauksessa oli NanoEdge AI Studion käyttö. NanoEdge AI Studiolla tehtävä, tekoälykirjaston löytämiselle kriittinen, benchmark-prosessi kesti opinnäytetyötä tehtäessä huomattavasti pidempään kuin NanoEdge AI Studion käyttöoppaassa annettiin ymmärtää. Koska prosessissa kesti pitkään, eikä käytettävällä laitteistolla ollut realistista suorittaa benchmark-prosessia loppuun asti, sillä siinä olisi kestänyt mahdollisesti viikkoja, on vaikea arvioida tuliko NanoEdge AI Studiosta käyttökelpoista hyödynnettävää materiaalia. Benchmark-prosessin aikana havaittiin myös, että se kuormittaa käytettävän tietokoneen prosessoria huomattavan paljon. Rajoittavana tekijänä prosessin onnistumiselle on siis voinut olla käytössä ollut tietokone, jonka prosessori ei välttämättä ollut tarpeeksi tehokas NanoEdge AI Studion käyttöä varten. Cartesiam taikka STMicroelectronics eivät ole ilmaisseet järjestelmävaatimuksia NanoEdge AI Studion tehokasta käyttöä varten, joten on vaikea kertoa, kuinka tehokkaan koneen tehokasta käyttöä varten mahdollisesti tarvitsisi. Itse projektin kääntäminen kohdealustalle STM32CubeIDE:llä onnistui, kunhan ensiksi varmistettiin projektin rakenteesta, että kooditiedostoissa, pääasiassa PMTask.c- ja NanoEdgeAI.h- tiedostojen käytettävät funktiot ovat yhteensopivia toistensa kanssa taulukon 3 osoittamalla tavalla.

Kaikenkaikkiaan funktiopaketin käytöstä saatiin laadittua selkeät ohjeet, jossa käytiin vaihe vaiheelta läpi, kuinka funktiopakettia käytetään. FP-AI-NANOEDG1-kunnonvalvontafunktiopaketin ja STEVAL-STWINKT1B-kehitysalustan koneoppimisytimen täyttä toiminnallisuutta ei valitettavasti päästy testaamaan täysin. Funktiopaketin tekoälyn koulutusominaisuutta ei päästy testaamaan käytännön olosuhteissa. Tämä johtui siitä, että CLI:ssä ajettu neai_learn script epäonnistui, kuten kuvassa 45 on näytetty. Kuten kappaleessa 6.3 käsiteltiin, asian epäillään johtuneen keskeneräiseksi jääneestä NanoEdge AI Studion benchmark-prosessista. Tätä ei tosin ehditty vahvistamaan taikka korjaamaan opinnäytetyön aikana.

Työtä rajoittavana tekijänä on voinut vaikuttaa työssä käytetty koneisto, eli kannettava Windows 10 -tietokone sekä Windows 10 -pöytäkone, joissa molemmissa oli käytössä Intel:n i5-kantaiset prosessorit. Benchmark-prosessin aikana seurattiin tietokoneiden kuormitusta. Prosessorin kuormitus oli koko prosessin ajan minimissään 85% yltäen silti usein 100% kuormituspiikkeihin. NanoEdge AI Studion benchmark-prosessi voi siis mahdollisesti tarvita tehokkaamman tietokoneen, jotta sen suoritus olisi tehokkaampi.

Koska tekoälyn koulutus ei onnistunut kohdealustalla, eikä myöskään luonnollisesti sen testaus kunnonvalvontatarkoituksessa, on vaikea arvioida suoritetun työn perusteella kuinka hyvin STEVAL-STWINKT1B ja FP-AI-NANOEDG1 soveltuisivat älykkäisiin kunnonvalvontasovelluksiin. On myös huomattava, että opinnäytetyöprosessin kesken NanoEdge AI Studioon tuli ohjelmistopäivitys, ja samalla myös ilmestyi uusi funktiopaketti, FP-AI-MONITOR1, joka on korvaaja FP-AI-NANOEDG1-funktiopaketille. Tämän johdosta STMicroelectronics ei enää suosittele, että FP-AI-NANOEDG1-funktiopakettia käytettäisiin pohjana uusien kunnonvalvontasovellusten kehityksessä. On epäselvää, onko tämä vaikuttanut negatiivisesti FP-AI-NANOEDG1-funktiopaketin tekoälyominaisuuksien testaukseen.

Sellaisenaan opinnäytetyötä voidaan kuitenkin mielestäni käyttää teoriaosan sekä STEVAL SensorTile Wireless Industrial Node STEVAL-STWINKT1B -kehitysalustan esittelyn sekä perusominaisuuksien käytön osalta koulutuksessa kurssimateriaalina, niin kuin opinnäytetyöllä oli tarkoituskin. Ennen kuin FP-AI-NANOEDG1:stä koskevaa osuutta voidaan hyödyntää täysin materiaalina, olisi suositeltavaa testata esimerkiksi Kajaanin Ammattikorkeakoulun supertietokoneella. Mikäli FP-AI-NANOEDG1-funktipaketti saataisiin toimimaan järkevästi, mielestäni sitä voitaisiin edelleen hyödyntää koulutuksessa.

Jatkokehittämisen kannalta voisi myös olla hyvä ajatus tutustua uuteen FP-AI-MONITOR1-funktiopakettiin, sillä siinä voidaan hyödyntää STEVAL-STWINKT1B-kehitysalustan sulautettuja antureita monipuolisemmin, kuin mitä FP-AI-NANOEDG1-funktiopaketissa voidaan. Pääpiirteittäin FP-AI-MONITOR1:n käyttö on hyvin samankaltainen, se on vaan päivitetympi versio aiemmasta. Koska teknologia ja teollisuus ovat alati kehityksen alla, on muutenkin hyvä pysyä ajan hermolla ja testata uusia STMicroelectronics:n julkaisemia funktiopaketteja.

Mikäli opinnäytetyötä halutaan ajatella kurssimateriaalina, niin yksi esimerkki kehitysalustalla tehtävästä kurssityöstä voisi olla seuraavanlainen:

Ota tutkimuksen alle jokin arkinen laite tai kone, jonka kuntoa haluat seurata. Tämä voi olla esimerkiksi tiskikone, jääkaappi, pyykinpesukone jne. Mieti, minkälaista dataa siitä voidaan mahdollisesti kerätä. Suunnittele ja toteuta FP-AI-NANOEDG1/FP-AI-MONITOR1:stä hyödyntäen kunnonvalvontasovellus ja testaa sen toimintaa. Raportoi tulokset.

Tällaisella tehtävänannolla voitaisiin testata opiskelijan ymmärrystä tärkeistä aihepiireistä, kuten tekoäly, koneoppiminen, ennakoiva vikadiagnostiikka ja teollisuuden älykäs kunnonvalvonta.

Tehtävänannon kautta opiskelija oppisi myös tuntemaan kehitysalustan sulautetut anturit, miten se toimii, miten siihen luodaan yhteys ja kuinka näillä keinoin saataisiin koulutettua tekoäly kunnonvalvontatarkoituksiin. Raportoinnin kautta opiskelija saisi myös vahvistettua oleellista taitoa kertoa tekemästä työstään selostaen mikä meni oikein, mikä meni pieleen, minkälaisia tuloksia työstä saatiin ja mitä ne loppujen lopuksi tarkoittavat.

Lähteet

1 Rhoards, J. Best Practices Guide: Predictive Maintenance Using Automatic Fault Detection and Diagnostics. ISC, International Institute for Sustainable Laboratories. [Internet] PDF-dokumentti. 2020. S. 1-6. [viitattu 10.9.2021] Saatavissa: https://www.analytika.com/best-practices-guide-predictive-maintenance-using-automatic-fault-detection-and-diagnostics/

2 Dr Bangert, P. Smart Condition Monitoring Using Machine Learning. Algorithmica Technologies. GmbH. [Internet] Raportti. 2021. [viitattu 5.9.2021] Saatavissa: http://www.algorithmica-technologies.com/en/case_studies/smart-condition-monitoring-using-machine-learning

3 Boucher, P. Artificial intelligence: How does it work, why does it matter and what can we do about it? European Parliamentary Research Service. Scientific Foresight Unit. [Internet] Tutkimus. 2020. S. 1-14. [viitattu 30.8.2021] Saatavissa: https://www.europarl.europa.eu/thinktank/en/document/EPRS_STU(2020)641547

4 Euroopan Parlamentti. Mitä tekoäly on ja mihin sitä käytetään? Euroopan Parlamentti. Yhteiskunta. [Internet] Artikkeli. 2020. [viitattu 30.8.2021] Saatavissa: https://www.europarl.europa.eu/news/fi/headlines/society/20200827STO85804/mita-tekoaly-on-ja-mihin-sita-kaytetaan

5 SAS. Mitä on tekoäly (AI) ja miksi se on tärkeää? Olemus ja merkitys. SAS Institute Inc. [Internet] Artikkeli. 2021. [viitattu 30.8.2021] Saatavissa: https://www.sas.com/fi_fi/insights/analytics/what-is-artificial-intelligence.html

6 Mäntylä, J. Mitä tekoäly on? Skycode Oy. [Internet] Artikkeli. [viitattu 30.8.2021] Saatavissa: https://xn--tekoly-eua.info/mita_tekoaly_on/

7 Pai, A. CNN vs RNN vs ANN. Analyzing 3 Types of Neural Networks in Deep Learning. Analytics Vidhya. [Internet] Artikkeli. 2020. [viitattu 14.10.2021] Saatavissa: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/

8 Nicholson, C. Beginner's Guide to Neural Networks and Deep Learning. Pathmind A.I. Wiki. [Internet] Artikkeli. 2020. [viitattu 14.10.2021] Saatavissa: https://wiki.pathmind.com/neural-network

9 Nielsen, M. Neural Networks and Deep Learning. [Internet] Verkkokirja. 2019. [viitattu 14.10.2021] Saatavissa: http://neuralnetworksanddeeplearning.com/

10 Donges, N. Gradient Descent: An Introduction to 1 of Machine Learning's Most Popular Algorithms. Built In. [Internet] Artikkeli. 2021. [viitattu 14.10.2021] Saatavissa: https://builtin.com/data-science/gradient-descent

11 Great Learning Team. Types of Neural Networks and Definition of Neural Network. Great Learning. [Internet] Artikkeli. 2020. [viitattu 14.10.2021] Saatavissa: https://www.mygreatlearning.com/blog/types-of-neural-networks/

12 IBM. What is computer vision? IBM. [Internet] Artikkeli. 2021. [viitattu 1.9.2021] Saatavissa: https://www.ibm.com/topics/computer-vision

13 MathWorks. Machine Learning with MATLAB. The MathWorks Inc. [Internet] Verkkokirja. 2020. s. 5-9, 16-29.[viitattu 2.9.2021] Saatavissa: https://se.mathworks.com/campaigns/offers/machine-learning-with-matlab.html

14 SAS. Machine Learning – What is it and why it matters. SAS Institute Inc. [Internet] Artikkeli. 2021. [viitattu 1.9.2021] Saatavissa: https://www.sas.com/en_ae/insights/analytics/machine-learning.html

15 Puustinen, M. Älykäs kunnossapito tehostaa tuotantoa. Teknologiainfo, Mediaplanet. Kunnossapito. [Internet] Artikkeli. 2019. [viitattu 5.9.2021] Saatavissa: https://www.teknologiainfo.com/teollisuus/alykas-kunnossapito-tehostaa-tuotantoa/

16 Salminen, J. Tekoäly mahdollistaa älykkään kunnossapidon. ProMaint, kunnossapidon ja tuotannon erikoislehti. [Internet] Näkökulma. Artikkeli. 2019. [viitattu 5.9.2021] Saatavissa: https://promaintlehti.fi/Nakokulma/Tekoaly-mahdollistaa-alykkaan-kunnossapidon

17 Mäki, K. Älykäs kunnossapito. ProMaint, Kunnossapidon ja tuotannon erikoislehti. Kunnonvalvonta ja käyttövarmuus. [Internet] Artikkeli. 2019. [viitattu 5.9.2021] Saatavissa: https://promaintlehti.fi/Kunnonvalvonta-ja-kayttovarmuus/Alykas-kunnossapito

18 STMicroelectronics. ST's Condition Monitoring Solutions. STMicroelectronics. Esite. [Internet] 2021. s. 2. [viitattu 5.9.2021] Saatavissa: https://www.st.com/content/st_com/en/campaigns/condition-monitoring-solutions-brochure.html?ecmp=tt22584_gl_ps_aug2021&aw_kw=condition%20monitoring%20systems&aw_m=p&aw_c=14234832466&aw_tg=aud-1026006975487:kwd-2840498222&aw_gclid=EAIaIQobChMIlp-QhZ_M8gIVLgd7Ch0qDgwuEAAYASAAEgI5kfD_BwE&gclid=EAIaIQobChMIlp-QhZ_M8gIVLgd7Ch0qDgwuEAAYASAAEgI5kfD_BwE

19 Epicor. Mikä on Teollisuus 4.0 – Teollinen esineiden Internet (IIoT, Industrial Internet of Things)? Epicor Software Corporation. [Internet] Artikkeli. 2021. [viitattu 25.8.2021] Saatavissa: https://www.epicor.com/fi-fi/resource-center/articles/what-is-industry-4-0/

20 IBM. What is Industry 4.0? IBM. Artikkeli. 2020. www-dokumentti. Saatavissa: https://www.ibm.com/topics/industry-4-0 [viitattu 25.8.2021]

21 Cartesiam. NanoEdge AI Studio Documentation – What is NanoEdge AI Library?. Cartesiam. [Internet] 2020. [viitattu 23.8.2021] Saatavissa: https://cartesiam-neai-docs.readthedocs-hosted.com/studio/studio.html#i-what-is-nanoedge-ai-library

22 STMicroelectronics. STM32CubeProg STM32CubeProgrammer all-in-one software tool – DB3420 Rev 4. STMicroelectronics. Esite. [Internet] 2019. s. 1. [viitattu 16.11.2021] Saatavissa: https://www.st.com/en/development-tools/stm32cubeprog.html

23 STMicroelectronics. STM32CubeIDE Integrated development environment for STM32 products - DB3871 Rev 6. STMicroelectronics. Esite. [Internet] 2021. s. 2. [viitattu 16.11.2021] Saatavissa: https://www.st.com/en/development-tools/stm32cubeide.html

24 STMicroelectronics. STSW-LINK004 STM32 ST-LINK Utility for STM32 MCUs - DocID029852 Rev 1. STMicroelectronics. Esite. [Internet] 2016. s. 1. [viitattu 16.11.2021] Saatavissa: https://www.st.com/en/development-tools/stsw-link004.html/

25 STMicroelectronics. STSW-STWINKT01 Firmware examples for STEVAL-STWINKT1B evaluation kit for Industry 4.0 - DB3968 Rev 7. STMicroelectronics. [Internet] 2020. s. 1. [viitattu 22.11.2021] Saatavissa: https://www.st.com/en/embedded-software/stsw-stwinkt01.html#documentation

26 STMicroelectronics. UM2777: How to use the STEVAL-STWINKT1B SensorTile Wireless Industrial Node for condition monitoring and predictive maintenance applications. STMicroelectronics. Käyttöohje. [Internet] 2021. s. 2-9, 19, 22. [viitattu 10.11.2021] Saatavissa: https://www.st.com/content/st_com/en/search.html#q=STEVAL-STWINKT1B%20firmware-t=resources-page=1

27 STMicroelectronics. HTS221 - Capacitive digital sensor for relative humidity and temperature. STMicroelectronics. [Internet] 2021. [viitattu 10.11.2021] Saatavissa: https://www.st.com/en/mems-and-sensors/hts221.html

28 STMicroelectronics. LPS22HH - High-performance MEMS nano pressure sensor: 260-1260 hPa absolute digital output barometer. STMicroelectronics. [Internet] 2021. [viitattu 10.11.2021] Saatavissa: https://www.st.com/en/mems-and-sensors/lps22hh.html

29 STMicroelectronics. STTS751 - 2.25 V low-voltage local digital temperature sensor. STMicroelectronics. [Internet] 2021. [viitattu 10.11.2021] Saatavissa: https://www.st.com/en/mems-and-sensors/stts751.html

30 STMicroelectronics. ISM330DHCX - iNEMO inertial module with Machine Learning Core, Finite State Machine with digital output for industrial applications. STMicroelectronics. [Internet] 2021. [viitattu 10.11.2021] Saatavissa: https://www.st.com/en/mems-and-sensors/ism330dhcx.html

31 STMicroelectronics. IIS3DWB - Ultra-wide bandwidth, low-noise, 3-axis digital vibration sensor. STMicroeletronics. [Internet] 2021. [viitattu 10.11.2021] Saatavissa: https://www.st.com/en/mems-and-sensors/iis3dwb.html

32 STMicroelectronics. IIS2DH - Ultra low-power high performance 3-axes accelerometer with digital output for industrial applications. STMicroelectronics. [internet] 2021. [viitattu 10.11.2021] Saatavissa: https://www.st.com/en/mems-and-sensors/iis2dh.html

33 STMicroelectronics. IIS2MDC - High accuracy, ultra-low-power ,3-axis digital output magnetometer. STMicroelectronics. [Internet] 2021. [viitattu 10.11.2021] Saatavissa: https://www.st.com/en/mems-and-sensors/iis2mdc.html

34 STMicroelectronics. IMP23ABSU - Analog bottom port microphone with frequency response up to 80kHz for Ultrasound analysis and Predictive Maintenance applications. STMicroelectronics. 2021. www-dokumentti. Saatavissa: https://www.st.com/en/mems-and-sensors/imp23absu.html [viitattu 10.11.2021]

35 STMicroelectronics. TS922 - Excellent audio performance / low distortion (0.005%). STMicroelectronics. [Internet] 2021. [viitattu 10.11.2021] Saatavissa: https://www.st.com/en/amplifiers-and-comparators/ts922.html

36 STMicroelectronics. IMP34DT05 - MEMS audio sensor omnidirectional digital microphone for industrial applications. STMicroelectronics. [interntet] 2021. [viitattu 10.11.2021] Saatavissa: https://www.st.com/en/mems-and-sensors/imp34dt05.html

37 STMicroelectronics. UM2621: How to use the software package for the SensorTile Wireless Industrial Node based on STM32Cube. STMicroelectronics. Käyttöohje. [Internet] 2020. [viitattu 10.11.2021] Saatavissa: https://www.st.com/en/embedded-software/stsw-stwinkt01.html#documentation

38 STMicroelectronics. FP-AI-NANOEDG1 V1.0 getting started. STMicroelectronics. Artikkeli. [Internet] 2021. [viitattu 15.11.2021] Saatavissa: https://wiki.st.com/stm32mcu/wiki/AI:FP-AI-NANOEDG1_V1.0_getting_started

39 STMicroelectronics. FP-AI-NANOEDG1 V 2.0 getting started. STMicroelectronics. Artikkeli. [Internet] 2021. [viitattu 3.11.2021] Saatavissa: https://wiki.st.com/stm32mcu/wiki/AI:FP-AI-NANOEDG1_V2.0_getting_started