Tredu abb Flexpendant käyttöliittymä dokumentaatio

Leevi Pinta

Sisällysluettelo

[Tiivistelmä 2](#_Toc190633164)

[sovellukset 3](#_Toc190633165)

[Visual Studio Code (VS-Code) 3](#_Toc190633166)

[Robotstudio (ABB:n oma) 3](#_Toc190633167)

[Postman 3](#_Toc190633168)

[Ensimmäiset askeleet uutta käyttöliittymää tehdessä 4](#_Toc190633169)

[Robotstudion käynnistäminen 4](#_Toc190633170)

[Robottiin yhdistäminen 4](#_Toc190633171)

[Ensimmäiset askeleet vanhaa käyttöliittymää jatkaessa 5](#_Toc190633172)

[Vanhan koodin haltuunotto 5](#_Toc190633173)

[Robottiin yhdistäminen 5](#_Toc190633174)

[Miten robotin koodi noin niin kuin teoriassa toimii ja mitä missäkin? 6](#_Toc190633175)

[Controllerin ”luuranko” 6](#_Toc190633176)

# Tiivistelmä

Tässä dokumentissa kerrotaan, miten ABB:n flexpendantiin tehdään oma käyttöliittymä ja miten pystyy jatkamaan jo käytössä olevaa käyttöjärjestelmää.

# sovellukset

Jotta robottiin saadaan käyttöliittymä, niin se vaatii tietyt sovellukset. Yleensä näitä sovelluksia ovat koodieditori, sovellus robottia varten, sekä muita vaihtoehtoisia sovelluksia.

## Visual Studio Code (VS-Code)

Tässä projektissa käytin koodieditorina Visual Studio Codea, koska sitä on tullut käytettyä koko ajan ja sen osaa parhaiten. Koodieditori on ilmainen ja helppo oppia käyttämään.

Koodieditorin voi ladata täältä: [VS-Code download](https://code.visualstudio.com/Download)

## Robotstudio (ABB:n oma)

Jotta robottiin saadaan tietoa ja robotista saadaan tietoa, tarvitaan sovellus, joka yhdistyy robottiin. Kun kyse on ABB:n robotista, käytetään ABB:n omaa sovellusta eli Robotstudio. Robotti yhdistetään läppäriin kaapelilla ja sen jälkeen sovellukseen.

Robotstudion voi ladata täältä: [Robotstudio download](https://new.abb.com/products/robotics/software-and-digital/downloads)

Kun Robotstudio on ladattu ja käynnistetty ensimmäistä kertaa, pitää myös ladata tarvittavat lisäosat.

-> sovelluksen ylälaidasta: ”Add-ins” -> RobotWare for OmniCore -> Add

## Postman

Postman on vaihtoehtoinen lisäsovellus, jolla pystytään tarkastamaan, että onko robotin muuttujien osoite oikea, mutta niistä myöhemmin dokumentaatiossa.

Postmanin voi ladata täältä: [Postman download](https://www.postman.com/downloads/)

# Ensimmäiset askeleet uutta käyttöliittymää tehdessä

Aluksia kannattaa avata kaikki sovellukset, jotka olen maininnut. Ensinnäkin Visual Studio Codessa kannatta tehdä uusi kansio koodia varten.

## Robotstudion käynnistäminen

KYSEINEN OHJE ON, KUN ROBOTTIA EI VIELÄ KYTKETÄ KONEESEEN

Kun avaa Robotstudion, se saattaa vaikuttaa mahdottomalta labyrintiltä, mutta ei suinkaan. Kun ihan ensimmäistä kertaa aloittaa tekemään robotille käyttöjärjestelmää valitaan ensi uusi Station, joka avautuu, kun sovellus käynnistyy.

Tämän jälkeen sovellus ohjaa itse itsensä seuraavaan vaiheeseen, jossa valitaan, mille robotille työtä tehdään. Tämä tapahtuu ylälaidasta ”ABB Library”, josta aukeaa valikko, jossa on kaikki robotit. Kun olet valinnut oikean robotin ja oikeat asetukset robotille (tulee automattisesti), on aika tehdä Controller robotille.

Controller avataan tai tehdään ylälaidan kohdasta ”Controller” -> ”Add Controller”. Tämän jälkeen valitaan ”New Controller”, nimetään ja valitaan oikeat asetukset (KANNATTAA MUISTAA MINNE CONTROLLERIN TALLENTAA OMALLA KONEELLA) ja ”OK”. Tämän jälkeen mennään takaisin ”Add Controller” ja valitaan haluttu Controller. Nyt sovelluksen vasempaan reunaan tulee controllerin asiat.

Tiedoissa ”HOME” on se kansio, johon koodi halutetaan laittaa. Kun on jotain koodia minkä haluaa controlleriin laittaa, niin etsii koneen resurssien hallinasta kyseisen kansion ja kopioi kooditiedostot sinne.

## Robottiin yhdistäminen

Kun robotti halutaan läppäriin. Pitää yhdistää se kaapelilla. Kun kaapeli on kiinni koneessa, voidaan mennä Robotstudion ”File” kohtaan ja ”Online”. Tämän jälkeen ”One Click Connect” ja siellä pitäisi näkyä robotti, joka on yhdistetty läppäriin. Tässä kohtaa robotissa on yleensä jo controller ja muutkin asiat kohdallaan. Joten oman koodin voi suoraan lisätä robotstudioon, MUTTA se tapahtuukin tällä kertaa ylälaidasta ”Controller” -> ”File Transfer”. Näytölle tulee kaksi ikkunaa, jossa toinen on oman koneen resurssien hallinta ja toinen on robotin resurssien hallinta. Molemmista pitää etsiä oikeat paikat ja koodin voi silloin kopioida omalta koneelta robottiin.

# Ensimmäiset askeleet vanhaa käyttöliittymää jatkaessa

Kun on tiedossa, että joku on ennenkin tehnyt robotille käyttöliittymää, nämä ohjeet koskevat sitä.

## Vanhan koodin haltuunotto

Kun joku muu on tehnyt koodia jo valmiiksi robottia, se kannattaa ensin saada haltuun ja VS-Codeen.

Tässä projektissa koodi löytyy Githubista, osoitteesta: [Leppaax robotti UI](https://github.com/tredu/robotti-ui-Leppaax)

Kun koodi on hallussasi, kannattaa tutustua koodiin ja käydä koodi läpi, jotta siitä ymmärtää siitä jotain. Yllä olevassa koodissa on esimerkiksi koodi kommentteja kertomassa ja helpottamassa ymmärtämistä, sekä tämän dokumentin loppuosassa on koodin palasia selvennettynä.

## Robottiin yhdistäminen

Kun kaapeli on kiinni koneessa, voidaan mennä Robotstudion ”File” kohtaan ja ”Online”. Tämän jälkeen ”One Click Connect” ja siellä pitäisi näkyä robotti, joka on yhdistetty läppäriin. Tässä kohtaa robotissa on yleensä jo controller ja muutkin asiat kohdallaan. Kansiosta HOME pitäisi löytyä sama koodi, jonka otit juuri netistä tai muualta. Jos näin ei ole, kannattaa tarkistaa mistä koodin otti, että on oikea.

# Miten robotin koodi noin niin kuin teoriassa toimii ja mitä missäkin?

Hyvä dokumentti koodin ymmärtämiseen on Githubin kansiossa ”Dokumentaatio”: ” OmniCoreAppSDK-manual.pdf”.

Hyvä videosarja koodin ymmärtämiseen on: [How to create an OmniCore™ Web App](https://www.youtube.com/watch?v=0g3mQZseDLo), videosarjassa on 4 osaa.

## Controllerin ”luuranko”

Controller alkaa tiedostosta appinfo.xml, jossa kerrotaan oman controllerin perustiedot, eli nimi, kuva ja se, että mistä tiedostosta controller ottaa koodin. Pää tiedosto koodille on EW.html, jossa alussa otetaan käyttöön kaikki fp-componentit, eli jokainen valmis pala, jota ABB tarjoaa. Tässä tiedostossa on myös yleistä tyylitystä ja jokainen ”palkki” käyttöliittymässä.

Jokaiselle omalle robotin osalle on oma tiedosto: Arm, Belt ja Cylinder. Näissä koodataan erikseen, mitä kyseisessä ”palkissa” tehdään.

## RAPID KOODI

Githubissa Dokumentaation alla on RAPID-koodi.

RAPID-koodin alussa kerrotaan kaikki muuttujat, jota halutaan muuttaa. Tämän jälkeen tehdään jokaiselle funktiolla/muuttujalle oma PROC eli ajettava funktio. ” SetDO ABB\_IO\_X1\_0\_DO4, 1;”, SetDO on rapidin sana muuttamiselle, ABB\_IO\_X1\_0\_DO4 on robotin lähtö, jota muutetaan ja perään laitetaan joko 1 eli päällä tai 0 eli pois. Loppuun laitetaan Main PROC, jossa kutsutaan jokaista pienenpää PROC:ia.

## Selitys Funktiolle, joka yhdistää napin tai jonkun muun RAPID-muuttujaan

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto

Tekoälyn generoima sisältö voi olla virheellistä.

const response = await fetch (<https://192.168.125.1:443/rw/rapid/symbol/RAPID/T_ROB1/MainModule/AutomaticModeOn/data>”

Tämä ottaa yhteyden robotin RAPID-muuttujaan. Alussa on robotin ip osoite, sitten tulee osoite RAPID- koodiin.

Headers kohtaan tulee kaikkia vaikuttavia asetuksia kuten sisäänkirjautumiseen tarvittavat tiedot.

## EW.HTML controlbox komponentti

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Tekoälyn generoima sisältö voi olla virheellistä.

Alussa nimi palikalle. Sitten jokaiselle napille tarvitaan oma <div> joka sitten yhdistetään johonkin nappulaan.

## Valikko/Burgermenu

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Tekoälyn generoima sisältö voi olla virheellistä.

Alussa tehdään se menu niillä FP-komponenteilla ja alapuolella nimi.

AddView lisää yhden uuden sivun valikkoon, ensimmäisiin hakoihin tulee nimi sille sivulle sinne valikkoon, sitten yhdistetään se EW.HTML tiedostosta löytyviin controlbox palikoihin ja viimeiseen tulee kuva tai logo sinne valikkoon. Hakojen jälkeen tulee true tai false sen perusteella, onko se sivu ensimmäinen, joka aukeaa, kun käyttöliittymä avataan (eli vain yhteen voi tulla true).

## Nappulan luominen

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Tekoälyn generoima sisältö voi olla virheellistä.

Alussa luodaan nappula FP-komponenteilla ja sen jälkeen mahdollinen teksti nappulalle.

…attachToId(”…”); on se, joka yhdistää napin EW.html:stä löytyvään nappulaan, eli nimen pitää olla sama.

Kaikki .\_root.style… on nappulan omaa tyyliä.

## Postman

Kuva, joka sisältää kohteen teksti, ohjelmisto, Multimediaohjelmisto, kuvakaappaus

Tekoälyn generoima sisältö voi olla virheellistä.

Jos et ole varma toimiiko tai onko RAPID-koodin osoite oikea, pystyt käyttämään postmania testaamiseen.

Jos halutaan vain katsoa, että toimiiko osoite, laitetaan vasempaan yläkulmaan GET

Osoiteriville laitetaan haluttu osoite RAPID koodille

Authorization kohtaan laitetaan robotin käyttäjätunnus ja salasana. Jos niitä ei ole muutettu, laitetaan ”Basic Auth” ja käyttäjätunnus ”Default User” ja salasana: ”robotics”.

Headers kohtaan samat mitä yllä olevassa kuvassa.

ja body kohdassa ylälaidasta laitetaan sama, mitä Content-Type kohdassa Headers.