Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

LAURI LOUNASVAARA

Dynaaminen HTML-Näyttöratkaisu

SÄHKÖ- JA AUTOMAATIOTEKNIIKAN TUTKINTO-OHJELMA
2021

Tekijä(t)
Lounasvaara, Lauri

Julkaisun laji
Opinnäytetyö, AMK

Päivämäärä
syyskuu 2021

Sivumäärä
56

Julkaisun kieli
Suomi

Julkaisun nimi
Dynaaminen HTML-Näyttöratkaisu

Tutkinto-ohjelma
Sähkö- ja automaatiotekniikka

Tiivistelmä

Tämän työn tavoitteena oli tehdä HTML-näyttösovellus, joka lukee Modbus-rekistereitä, ja jonka sisältö vaihtelee käyttäjän tekemien valintojen perusteella. Työn alussa vertailtiin ja mietittiin eri vaihtoehtoja millä toteuttaa haluttu näyttösovellus. Työ tehtiin Kongsberg Maritime Finland Oy:lle.

Näyttösovellus toteutettiin Node-RED -ohjelmointityökalua käyttäen. Node-REDin lisäksi työssä tuli käytettyä myös montaa ohjelmointikieltä- ja työkalua. Eniten käytetty ohjelmointikieli oli JavaScript. Alusta, jossa näyttösovellus kehitettiin, oli Weidmüllerin kontrolleri. Weidmüllerin kontrollerissa Node-RED -ohjelmointityökalu on jo valmiiksi integroituna.

Tuloksena saatiin aikaiseksi toimiva näyttösovellus, joka on käyttäjän helposti muokattavissa. Näyttösovellusta jatkokehittämällä, siitä on mahdollisuus saada hyvin toimiva ja uudelleenkäytettävä ratkaisu paikallisnäytöksi.

Asiasanat
Node-RED, JavaScript, HTML, Modbus

Author(s)
Lounasvaara, Lauri

Type of Publication
Bachelor's thesis

Date
September 2021

Number of pages
56

Language of publication:
Finnish

Title of publication
Dynamic HTML display solution

Degree program
Electrical and automation engineering

Abstract

The aim of this thesis was to make an HTML display application that displays different things based on the choices made by the user. A few different alternatives were considered on how to carry out the development. This thesis work was done with Kongsberg Maritime Finland Oy.

The display application was made with Node-RED programming tool. In addition to Node-RED, many programming languages were used in the project. The most used programming language was JavaScript. The platform on which the application was developed, was a Weidmüller controller. The Node-RED programming tool is integrated in the Weidmüller controller itself.

The result was a functioning display application that was easily customizable by the user. By further developing the display application, it has the potential to become a well-functioning and easily reusable solution for local displays.

Key words
Node-RED, JavaScript, HTML, Modbus

ALKUSANAT

Kiitokset Kongsberg Maritime Finland Oy:lle opinnäytetyöpaikan ja -aiheen tarjoamisesta. Kiitos Timo Suvelalle opinnäytetyön ohjauksesta. Kiitos Marko Kuusenojalle opinnäytetyön kanssa neuvomisesta ja ohjaamisesta eteenpäin. Haluan myös kiittää Sami Lindgreniä, joka avusti ja neuvoi ohjelmoinnin eri kohdissa.

Raumalla 30.7.2021

Lauri Lounasvaara

SISÄLLYS

SYMBOLI- JA LYHENNELUETTELO (EI PAKOLLINEN)

SCADA – Supervisory Control And Data Acquisition, PC valvomo-ohjelmisto, jonka avulla voidaan valvoa ja ohjata prosessia tai tuotantoa

HMI – Human Machine Interface, ihmisen ja ohjelmoitavan logiikan (tai muun tekniikan) välissä oleva käyttöliittymä

Open Source – Avoin lähdekoodi

Npm – Node Package Manager, pakettien hallintaohjelmisto

IP-luokitus – Kansainvälisesti käytössä oleva järjestelmä, jolla määritellään sähkölaitteiden ja eri koteloiden kosketus ja vesitiiviyttä

IOT – Internet of Things, esineiden internet.

GPIO – General Purpose IO, yleiskäyttöinen pinni eri mikrokontrollereissa ja mikroprosessoreissa

JSON – JavaScript Object Notation, tiedostomuoto, joka on tarkoitettu tiedonvälitykseen

CSS – Cascading Style Sheets, verkkosivuja varten kehitetty tyylisivu

HTML – Hypertext Markup Language, merkintäkieli minkä avulla voidaan kirjoittaa internetsivuja

SVG – Scalable Vector Graphics, Kaksiulotteisten vektorikuvien kuvauskieli

1 JOHDANTO

Opinnäytetyö lähti käyntiin 2021 keväällä. Kongsberg Maritimen CMS-osasto etsi kesäharjoittelijaa opinnäytetyömahdollisuudella. Opinnäytetyön aihe vaikutti kiinnostavalta ja oli myös lähellä myös monia koulussa läpikäytyjä aihepiirejä. Käyttöliittymän suunnittelua ja tekemistä käytiin läpi usealla kurssilla koulussa. Koulussa tulleen käyttöliittymiin liittyvän kokemuksen perusteella luotin, että opinnäytetyön aihe on minulle itselleni sopiva, sekä uskoin selviytyväni työstä hyvin.

Opinnäytetyö tehtiin Kongsberg Maritime Finland Oy:lle Rauman yksikössä. Rauman lisäksi Kongsberg Maritimella on tehdas Kokkolassa. Yritys suunnittelee meriteollisuuden laitteita, sekä valmistaa ja myy niitä. Rauman yksikkö keskittyy propulsiolaitteisiin. Tilikaudella 2020 yritys työllisti noin 500 ihmistä.

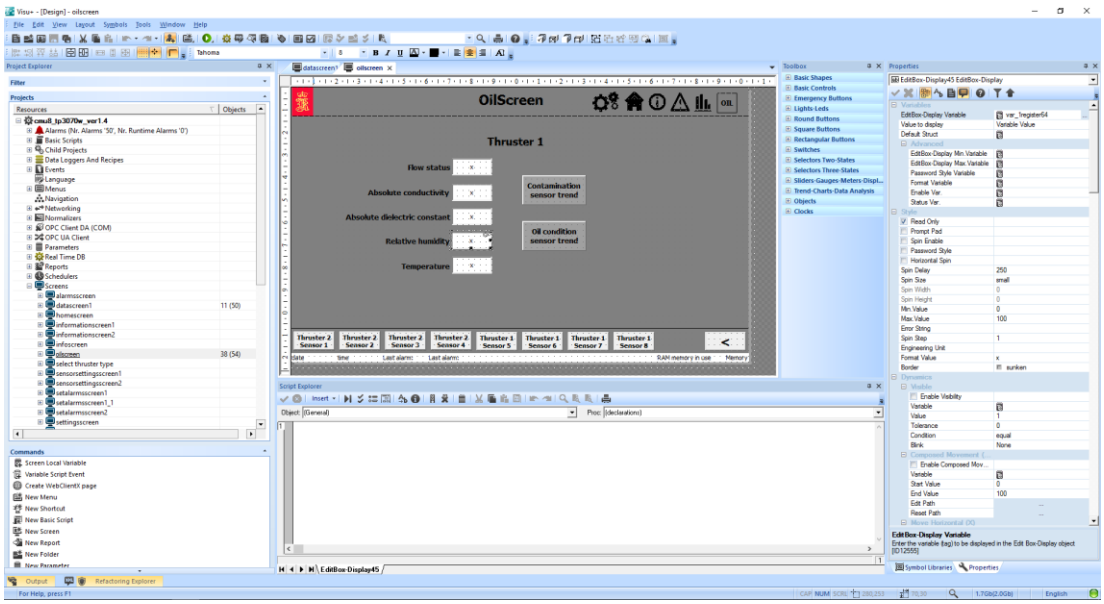
Projektin ideana oli suunnitella HTML-näyttöratkaisu, joka on helposti käyttäjän muokattavissa. Tavoitteena oli saada aikaseksi näyttö, josta käyttäjä pystyy valitsemaan halutut Modbus-rekisterit, nimeämään ne ja valitsemaan mitä tietoja niistä tulee näkyville. Tällä tavoin HTML-näyttö olisi helposti uudelleenkäytettävissä ja sovellettavissa eri käyttökohteissa. Käyttöönottaja voisi siis eri käyttökohteissa ottaa aina saman näyttöpohjan käyttöön, ja saada nopeasti haluamansa tiedot näkymään näytöllä. Tällainen näyttösovellus vähentäisi tarvetta sille, että vanhaa näyttösovellusta tarvitsee aina ohjelmoida uudelleen jokaista uutta projektia varten.

2 SOVELLUKSET

Pohtiessa toteutustapaa ja erilaisia alustoja, vastaan tuli monia eri sovelluksia. Monia eri sovelluksia harkittiin, ja pohdittiin onko niiden avulla mahdollista tehdä halutun kaltainen näyttösovellus.

2.1 VISU+ 2

VISU+ 2 on Phoenix Contactin kehittämä SCADA visualisointiohjelma. Ohjelman avulla pystytään luomaan käyttöliittymiä, joissa voidaan esittää esimerkiksi trendejä tai hälytyksiä halutusta laitteesta tai järjestelmästä. Ohjelman avulla on myös mahdollista luoda HMI-näyttö, joka tavallisesti asennetaan kosketusnäyttöpaneeliin, joka taas on yhteydessä ohjelmoitavaan logiikkaan, tai vastaavaan järjestelmään. VISU+ 2 sisältää kaksi eri ympäristöä, joista toinen on suunnittelua varten, ja toisella ”runtime” ympäristöllä kyetään ajamaan suunniteltua käyttöliittymää. (Phoenix Contactin www-sivut 2021.)



Kuva 1. VISU+ käyttöliittymän suunnitteluympäristö

VISU+:n avulla on myös mahdollista luoda tehdystä käyttöliittymästä HTML-nettisivu.

2.2 Grafana

Grafana on avoimen lähdekoodin web-sovellus, jonka avulla on mahdollista visualisoida dataa erilaisista tietokannoista. Grafanalla on mahdollista luoda monipuolisia ja interaktiivisia näyttöjä, jotka esittävät dataa esimerkiksi tauluina, kaavioina ja diagrammeina.

Kuva 2. Esimerkki Grafanan datan visualisoinnista (Grafanan www-sivut 2021.)

Torkel Ödegaard julkaisi ensimmäisen version Grafanasta vuonna 2014. Muutaman viime vuoden aikana Grafanasta on tullut yksi suosituimmista avoimen lähdekoodin projekteista GitHubissa. Ensin Grafana oli suunniteltu erilaisten aikasarjakantojen visualisointiin, mutta myöhemmin käyttö on laajentunut myös erilaisiin relaatiotietokantoihin. Grafana on hyvin suosittu keveytensä sekä avoimen lähdekoodinsa ansiosta. Monet suuret yritykset ovat ottaneet myös Grafanan käyttöönsä. Näitä yrityksiä ovat muun muassa: Paypal, ebay, Intel and Wikipedia. (Grafanan www-sivut 2021.)

2.3 Node-RED

Node-RED on IBM:n kehittämä vuopohjainen ohjelmointiympäristö. Node-REDin kehitys alkoi vuoden 2013 alkupuolella. IBM:llä työskentelevät Nick O'leary ja Dave Conway-Jones alkoivat kehittämään työkalua, millä voisi helposti yhdistää erilaisia

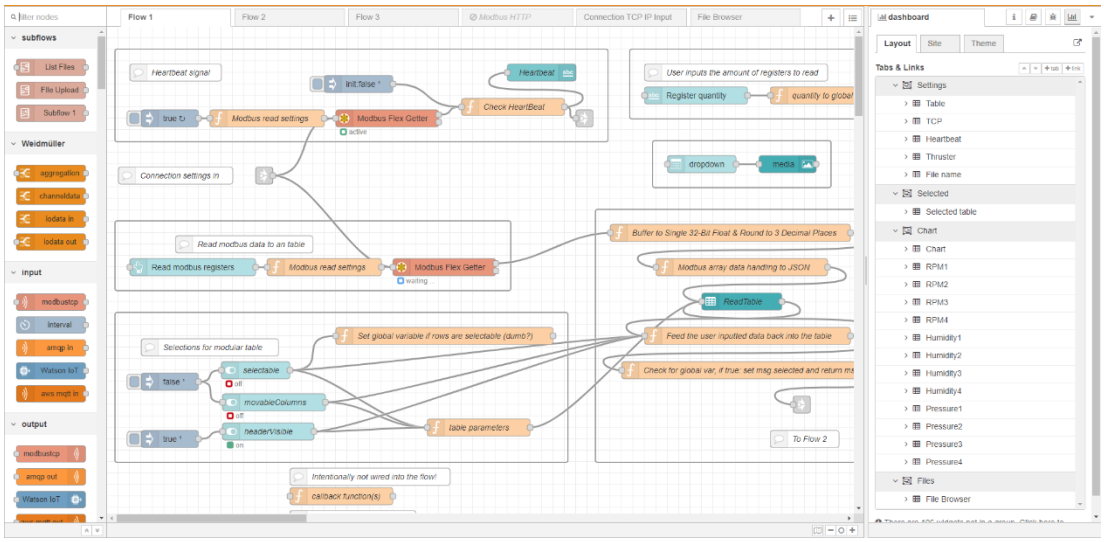
fyysisiä laitteita, ohjelmointirajapintoja, sekä erilaisia verkossa olevia palveluita. Nick O'Leary ja Dave Conway-Jones huomasivat nopeasti, että työkalu voisi olla helposti laajennettavissa ja hyödynnettävissä moneen eri asiaan ja suuntaan. Projekti muutettiin avoimen lähdekoodin projektiksi vuoden 2013 lokakuussa.

Node-RED nimenä kehittyi aluksi sanaleikkinä sanoista "Code Red" eli "punainen koodi". Node -osa nimestä viittaa siihen, että Node-RED pyörii Node.JS runtimen avulla, sekä myös siihen miten Node-RED -ohjelmointi toimii. Nimen RED -osuus ei itsessään tarkoita mitään, se on vain jäänne alkuperäisestä sanaleikistä mistä Node-RED sai nimensä. (Node-REDin www-sivut 2021.)

Node-RED -ohjelmointi tapahtuu selaimessa näkyvän visuaalisen editorin kautta. Editorissa yhdistetään toisiinsa erilaisia solmuja (node). Solmuja on monia erilaisia, ja jokainen niistä tekee eri asioita. Node-RED sisältää jo itsessään monia erilaisia solmuja, mutta niitä pystyy myös lisäämään käytettäväksi joko editorissa olevan työkalun tai komentorivin kautta. Node-RED -käyttäjät ovat myös tehneet omia solmujaan ja ne ovat myös vapaasti lisättävissä editoriin.

Ohjelmien jakaminen on tehty myös helpoksi. Koko projektin, tai osan siitä, voi muuttaa JSON-tiedostoksi. Tämän JSON-tiedoston voi jakaa ja sen voi ottaa käyttöön yhtä helposti lisäämällä sen omaan projektiinsa editorissa.

Yksi Node-REDin hyötyjä on se, että ohjelmoinnista ei tarvitse tietää juurikaan, mutta silti voi tajuta mitä ohjelmassa tapahtuu katsomalla solmuja ja niiden yhteyksiä. Solmuja voi myös nimetä vuon lukemista helpottavalla tavalla.



Kuva 3. Node-Red -ohjelmaeditorin selainnäkymä

3 OHJELMOINTIKIELET

Näyttösovelluksen toteuttamiseen tutkittiin monia eri vaihtoehtoja. Itse näyttösovelluksen ohjelmointiin oli jo monta vaihtoehtoa. Tämän lisäksi piti saada mietittyä, että millä alustalla valmista ohjelmaa on järkevintä pyörittää.

3.1 JavaScript

JavaScript on ohjelmointikieli, joka on julkaistu vuonna 1995. Se kehitettiin alun perin käytettäväksi Web-sivujen kanssa. JavaScriptin avulla Web-sivuille on mahdollista luoda erilaisia dynaamisia toiminnallisuuksia. Myöhemmin JavaScriptiä on käytetty myös laajemmin verkkoselainten ulkopuolella. JavaScriptillä on kehitetty videopelejä sekä myös erilaisia työpöytäsovelluksia, Web-sovellusten lisäksi.

JavaScript kehitettiin aluksi nimellä Mocha, mutta nimi muutettiin myöhemmin JavaScriptiksi koska yritys joka JavaScriptin kehitti, oli juuri yhdistynyt Javan

kehittäneen yrityksen kanssa. JavaScript ei kuitenkaan nimestään huolimatta muistuta Javaa juurikaan. (Peyrott, 2017.)

3.2 HTML

HTML tulee sanoista ”Hypertext Markup Language, eli käännettynä hypertekstin merkintäkieli. HTML on käytännössä se kieli, jolla valtaosa internetissä olevista nettisivuista on kirjoitettu. HTML-koodissa määritellään erilaisilla tageilla mikä ja millainen mikä teksti on. HTML kehitettiin vuonna 1989. Sen alkuperäinen tavoite oli jakaa CERNin sisäisiä dokumentteja kätevästi internetin avulla. (Washingtonin yliopiston www-sivut 2021.)

4 LAITTEISTO

4.1 Weidmüller UC20-WL2000-AC

UC20-WL2000-AC on Weidmüllerin valmistama automaatiokontrolleri. Kontrolleriin on liitettävissä monia erilaisia kortteja. Kortit voivat lisätä kontrolleriin esimerkiksi analogisia tai digitaalisia sisääntuloja. Kontrollerissa itsessään on kaksi Ethernet-porttia, paikka SD-kortille, Micro USB 2.0 -portti ja liitynnät 24 voltille. Kontrolleri voi saada virtansa, joko näistä 24 voltin sisääntuloista, tai Micro USB portin kautta. Tällöin kontrolleri pysyy päällä vaikka toinen virran lähde katkeaisikin. USB portilla on myös oma verkkokorttinsa ja sen kautta pystyy yhdistämään kontrolleriin. Halutessaan kontrolleriin voi liittää molemmat virransyötöt.

Kuva 5. Weidmüller UC20-WL2000-AC (Weidmüllerin www-sivut 2021.)

Kontrolleri kiinnittyy DIN-kiskoon, joten kontrolleri soveltuu hyvin teolliseen ympäristöön. Kontrollerin IP luokitus on IP20, eli se on suojattu halkaisijaltaan 12,5 mm suuremmilta kappaleilta. Vedeltä laitetta ei ole suojattu lainkaan.

Laitteella itsessään on sisäistä Flash muistia 4 Gigatavua. Muistia on lisättävissä SD-kortilla 32 Gigatavua lisää. Prosessori kontrollerissa on Dual Core ARM Cortex A9, 624 MHz. Keskusmuistia kontrollerissa on 512 Megatavua. (Weidmüllerin www-sivut 2021.)

4.2 SKF Multilog on-line system IMx-8 ja IMx-16Plus

IMx8 ja IMx-16 ovat nimensä mukaisesti 8- ja 16-kanavaisia datankeräyslaitteita. Laitteet keräävät eri kanavista tulevan datan, ja joko lähettävät sitä eteenpäin tai säilövät paikallisesti - riippuen asetuksista. Laitteen tallentamaa dataa voi analysoida ja sen perusteella suunnitella huoltotoimenpiteitä laitteille. Laitteessa on 4GB sisäistä muistia datan tallentamista varten.

Laite käyttää 24 tai 48 voltin tasajännitettä. Laitteelle voidaan tuoda virta joko omaan paikkaansa liitäntään, tai vaihtoehtoisesti voidaan käyttää laitteen tukemaa Power over Ethernet tekniikkaa (PoE), ja tuoda tarvittava virta Ethernet liitännän kautta. Laite yhdistyy verkkoon Ethernet-liitännän avulla. IMx-16 tukee myös WiFi-yhteyttä.

IMx-laitteiden konfiguroiminen käyttöönotettaessa tapahtuu käyttämällä SKF:n tekemää mobiilisovellusta. Sovelluksella otetaan Bluetooth-yhteys laitteeseen. IMx-8 ei sisällä itsessään Bluetooth-yhteyttä, mutta laitteen mukana lähetetään Bluetooth-mokkula, jonka avulla laitteen konfiguraatio tapahtuu.

Laitteet ovat sertifioituja tuulivoima ja meriteollisuuteen, mutta niitä voidaan käyttää myös muilla tekniikan aloilla. (SKF www-sivut 2021.)

4.3 Weidmüller UV66-BAS-10-RES-W

Weidmüller UV66-BAS-10-RES-W on web-käyttöön suunniteltu kosketusnäyttö. Näyttö pyörii Linux-käyttöjärjestelmällä ja tukee HTML5-nettisivuja. Näytön asetuksissa näytölle asennetaan nettisivu, jonka näyttö avaa aina käynnistettäessä. Näin näytöllä on helppo esittää myös esimerkiksi paikallinen, oman verkon sisällä oleva nettisivu. Näytön koko on 10.1 tuumaa, mutta Weidmüllerin vastaavia näyttöjä on saatavilla 4.3” ja 15.6” väliltä. Keskusmuistia näytöllä on 1 Gigatavu. Näytön prosessorina toimii ARM Cortex A9 1 GHz NXP ® i.MX6 DualLite.

Näyttöpaneeli on suunniteltu upotettavaksi, ja sen etuosalla onkin IP66-luokitus. Ympäristön lämpötila on määritelty näytölle 0 - 50 °C välille.

Kuva 6. Weidmüller UV66-BAS-10-RES-W (Weidmüllerin www-sivut 2021.)

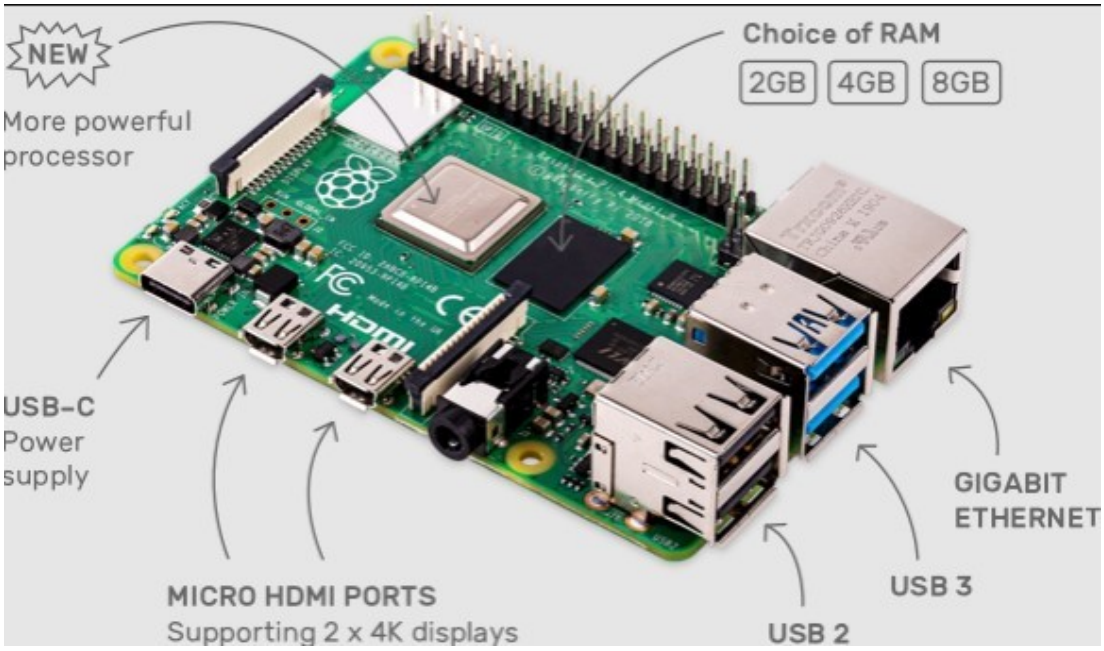
Näytössä on useita eri liitäntöjä. Kaksi Ethernetporttia (RJ45), kaksi USB 2.0 -porttia, yksi RS-232/RS-485/RS-422 sarjaliikenneportti. Näyttö ottaa virtansa 24 voltin syötöstä. Tämän lisäksi näytössä on fyysiset RESET- ja RESTORE-näppäimet, jolla näytön saa nollattua tai käynnistettyä uudelleen ongelmatilanteissa. (Weidmüllerin www-sivut 2021.)

4.4 Raspberry Pi 4 Model B

Raspberry Pi on pieni, tehokas, kämmenen kokoinen tietokone. Eri Raspberry Pi tietokoneet ovat olleet todella suosittuja jo useita vuosia, halvan hintansa, helppokäyttöisyytensä ja pienen kokonsa ansiosta. Alun perin Raspberry Pi:t olivat erityisen suosittuja elektroniikka ja tekniikkaharrastajien keskuudessa. Laitteella onkin todella helppoa tehdä ja testata monia erilaisia projekteja hyvin nopeasti. Tämän takia Raspberry Pi on todella suosittu IoT- sovelluksissa.

Raspberry Pi:n liitännät, osat ja ominaisuudet vaihtelevat hieman mallista ja vuodesta toiseen. Raspberry Pi 4 Model B:n prosessorina toimii Broadcom BCM2711, Quad core Cortext-A72 (ARM v8) 64-bit SoC @ 1.5GHz. Keskusmuistin määrä on itse valittavissa ja sen voi valita tarpeidensa mukaan. Vaihtoehdot ovat 2GB, 4GB tai 8GB. Raspberry Pi 4 tukee lisäksi 2.4GHz ja 5.0Ghz IEEE 802.11ac langatonta verkkoa ja 5.0 bluetooth yhteyttä.
Muut Raspberry Pi:n liitännät ovat:
2x USB 2.0
2x USB 3.0
1x Gigabit Ethernet
2x Micro-HDMI
1x Stereo ääni ja komposiitti video
1x Micro-SD portti käyttöliittymää ja tallennustilaa varten
1x MIPI DSI näyttö liitäntä
1x MIPI CSI kamera liitäntä
40x GPIO
1x USB-C portti

Laite voi saada virtansa joko USB-C-portin kautta tai syöttämällä virta suoraan GPIO-porttiin. Laite vaatii 5 voltin ja 3 ampeerin virtalähteen. Raspberry Pi 4 tuke myös PoE-virransyöttöä, eli laitteelle voidaan syöttää virta Ethernet portin kautta. Tämä kuitenkin vaatii erillisen PoE HAT -lisäkomponentin toimiakseen. (Raspberry Pin www-sivut 2021.)



Kuva 7. Raspberry Pi 4 (Raspberry Pin www-sivut 2021.)

Samalla kun IoT on kasvattanut suosiotaan, myös erilaiset Raspberry Pi -sovellukset ovat päätyneet teollisuuden maailmaan. Raspberry Pi:tä voi hyvin käyttää myös teollisuudessa esimerkiksi automaation kontrollerina, tai vaikka tiedonkeruulaitteena erilaisille mittaustiedoille.

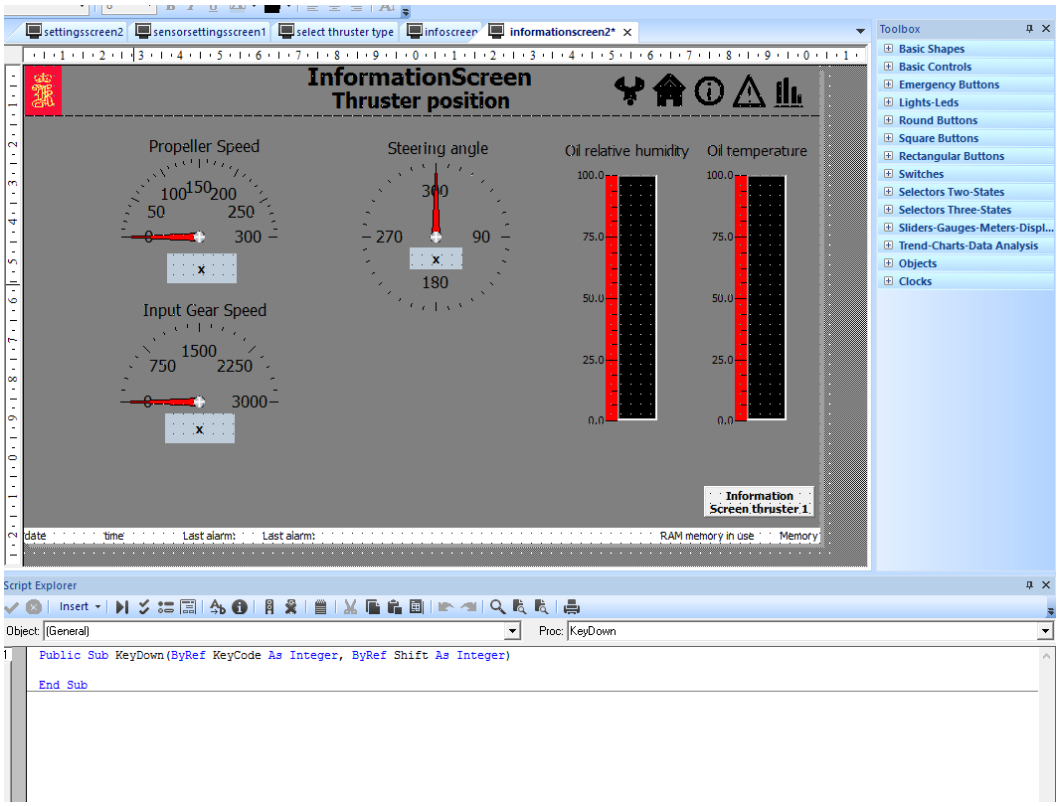
5 TOTEUTUSTAPOJEN VERTAILU

5.1 VISU+ käyttö

Ensimmäisenä mahdollisuutena toteuttaa halutun kaltainen HTML-näyttöratkaisu tutkin Phoenix Contactin VISU+ 2-sovellusta. Samalla sovelluksella olin itse tehnyt aikaisemmin yhden näyttösovelluksen juuri ennen opinnäytetyön aloittamista, joten se oli minulle tuttu ja halusin ottaa selvää, saako sen avulla aikaseksi halutun kaltaista dynaamista näyttöä.

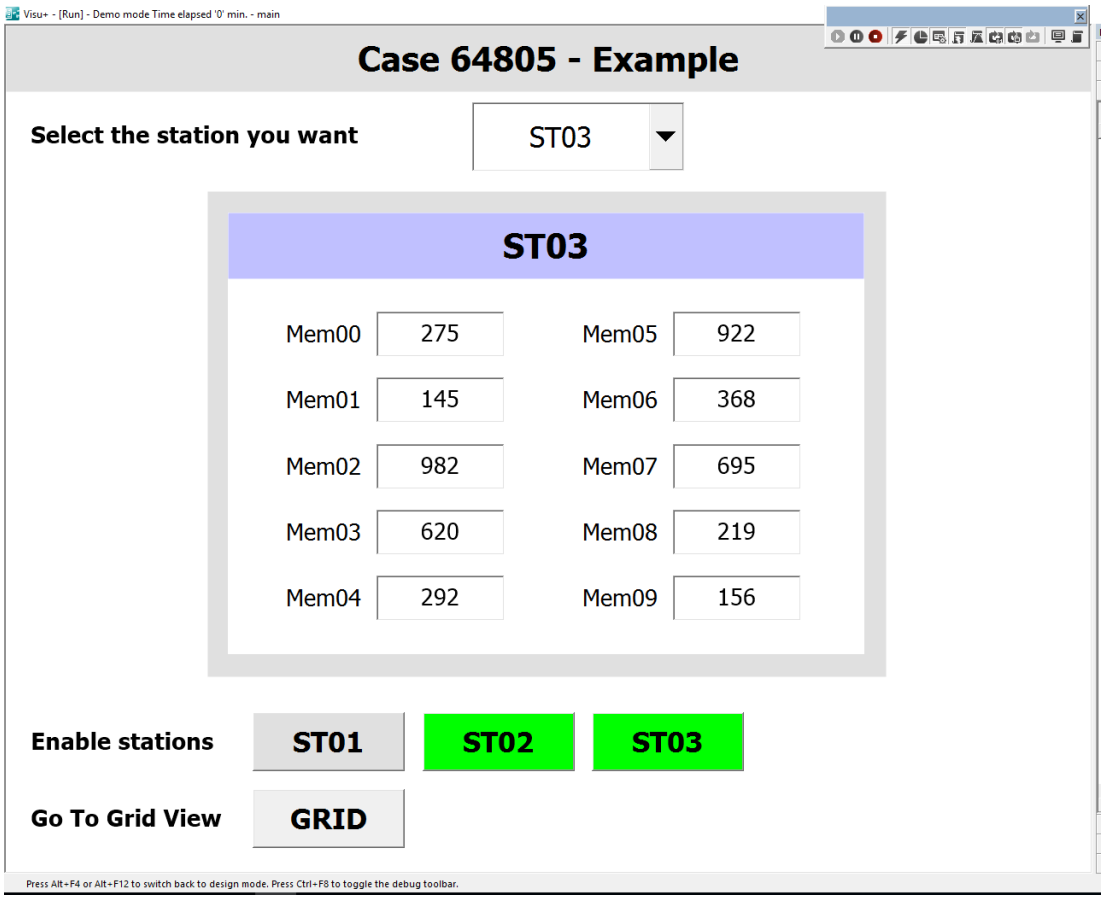
Verrattuna moneen muuhun vastaavaan ohjelmistoon, VISU:sta löytyy heikosti esimerkkejä ja keskustelua internetin keskustelufoorumeilta. Tästä syystä myös eri hakukoneilla ei löytynyt juurikaan tietoa, että kukaan olisi koskaan rakentanut vastaavaa sovellusta. Phoenix Contactin sivuilta löytyy kuitenkin kattavat dokumentaatiot ja käyttöohjeet.

Ohjelman käyttäminen on kohtuullisen yksinkertaista, työkaluvalikoista raahataan elementtejä ruuduille, ja niille asetetaan eri parametrejä ja muuttujia. VISU+ pystyy kuitenkin myös monimutkaisempiinkin sovelluksiin. Visual Basicin avulla pystyy luomaan monipuolisempia ohjelmia kuin pelkästään valmiiden rakennuspalikoiden avulla.



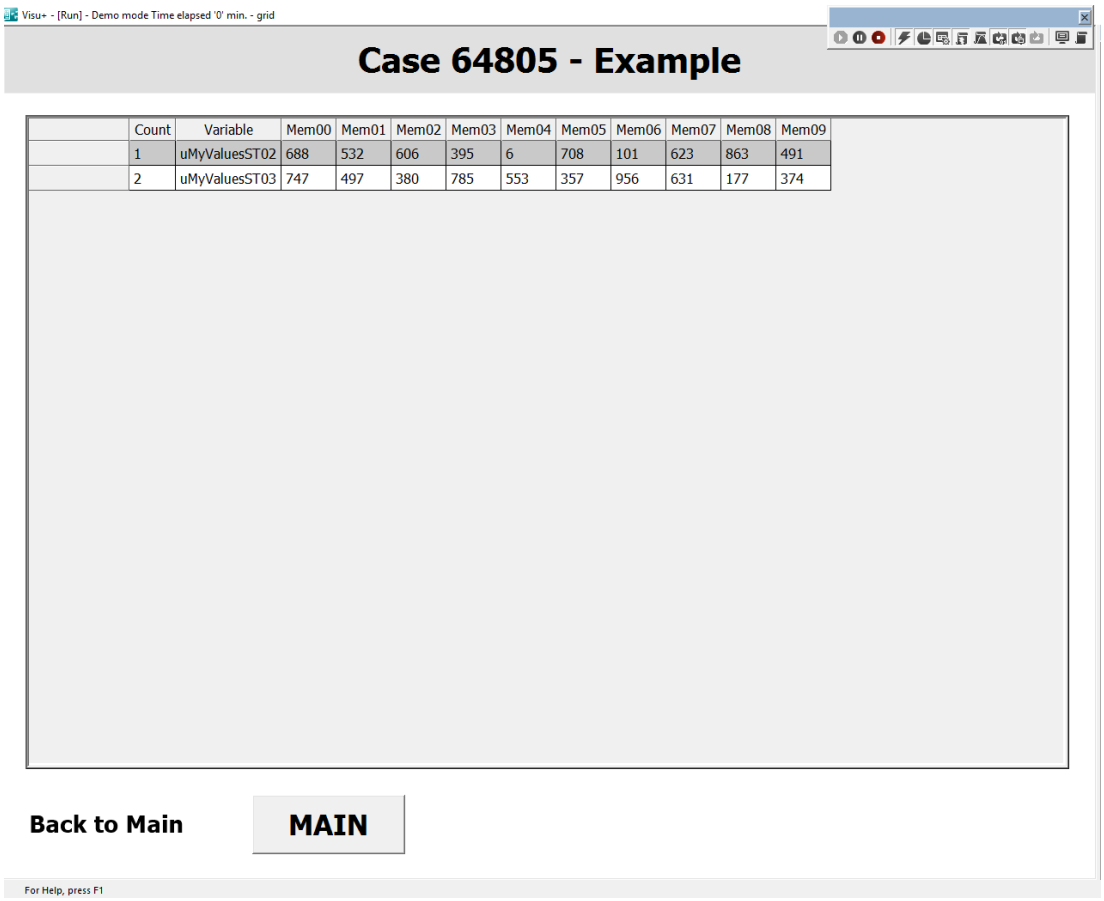
Kuva 8. VISU+ 2-suunnitteluohjelma

Phoenix Contactilta löytyy esimerkki missä luodaan taulukko dynaamisesti valintojen perusteella. Esimerkin perusteella käy selväksi, että eri elementtien luominen näytölle on mahdollista dynaamisesti. Nopeasti kävi myös ilmi, että ohjelmasta tuli hyvin hidas jo yhtä taulukkoa näytölle generoitaessa. Valmiissa näyttösovelluksessa generoituja elementtejä tulisi olemaan samaan aikaan useita kymmeniä.



Kuva 9. VISU+ 2 Runtime ympäristö

Näyttösovellus luo esimerkkinä kolme eri asemaa ja jokaiselle asemalle kahdeksan eri muisti arvoa. Esimerkkisovelluksessa ohjelma ei lue Modbus arvoja, vaan luo satunnaisia lukuja taulukkoon. Käyttäjä pystyy valitsemaan, mitkä eri asemista ovat valittuina, ja näytetään taulukossa. ”GRID” nappulaa painamalla aukeaa taulukkonäkymä, jossa näkyvät edellisellä sivulla valitut asemat.



Kuva 10. VISU+ 2 dynaamisen taulukkoluonnin esimerkki

Näyttösovellus oli hidas jo luotaessa yhtä taulukkoa, ja Visual Basic skripti oli jo suhteellisen pitkä ja monimutkainen. Tämän lisäksi minulla ei ollut juurikaan kokemusta Visual Basicin käytöstä, joten siirryin tutkimaan muita vaihtoehtoja.

5.2 Grafanan käyttö

Seuraavaksi siirryin tutkimaan olisiko Grafana sopiva ratkaisu näyttösovelluksen toteuttamiseksi. Hakemalla löytyi muutama projekti, joissa oli luotu näyttösovelluksia, joissa käyttäjä tekee valinnan mitä elementtejä ja objekteja näytöllä näkyy. Grafanalla on kohtuullisen helppoa luoda valikko, josta voi valita mitä tietoja, taulukoita tai graafeja näkyy näytöllä.

Myös automaattisesti näytölle luotavat elementit ovat mahdollisia, mutta vaativat enemmän JavaScriptin osaamista.

Grafanaa ei kuitenkaan ole suoranaisesti suunniteltu tämänkaltaisten näyttösovelluksien tekoon. Valmiissa ohjelmassa tulee olemaan monenlaisia käyttäjän syöttämiä tietoja ja eri asetusvalikoita, joiden avulla päätetään mitä näytetään ja mistä. Tämän kaltaiseen käyttöön Grafana ei sovellu kovin hyvin.

5.3 Node-RED ohjelmointiympäristön käyttö

Node-RED vaikutti heti lupaavalta vaihtoehdolta yksinkertaisuutensa ja aloittelijaystävällisyytensä ansiosta. Node-REDin kotisivuilla (nodered.org) on kattava dokumentaatio millä pääsee alkuun. Tämän lisäksi Node-REDin kotisivuilla on keskustelufoorumi missä käyttäjät voivat kysyä kysymyksiä, jakaa tekemiään ohjelmia ja auttaa toisia ohjelmoijia jakamalla neuvoja ja esimerkkejä.

Sen lisäksi että käyttäjät ja käyttäjäryhmät jakavat omatekemiään ohjelmiaan, jakavat he omatekemiä solmujaan. Solmut voivat olla hyvinkin tarkkoja käyttötarkoitukseltaan. Esimerkkinä yksi solmu, joka hakee netin tietokannasta Ison-Britannian junien aikataulut (Node-RED www-sivut, 2021). Toisena esimerkkinä solmujen monikäyttöisyydestä voisi esiin nostaa funktiosolmun. Funktiosolmun sisään voi kirjoittaa vapaasti JavaScript -koodia, ja koodi suoritetaan, kun solmu aktivoituu.

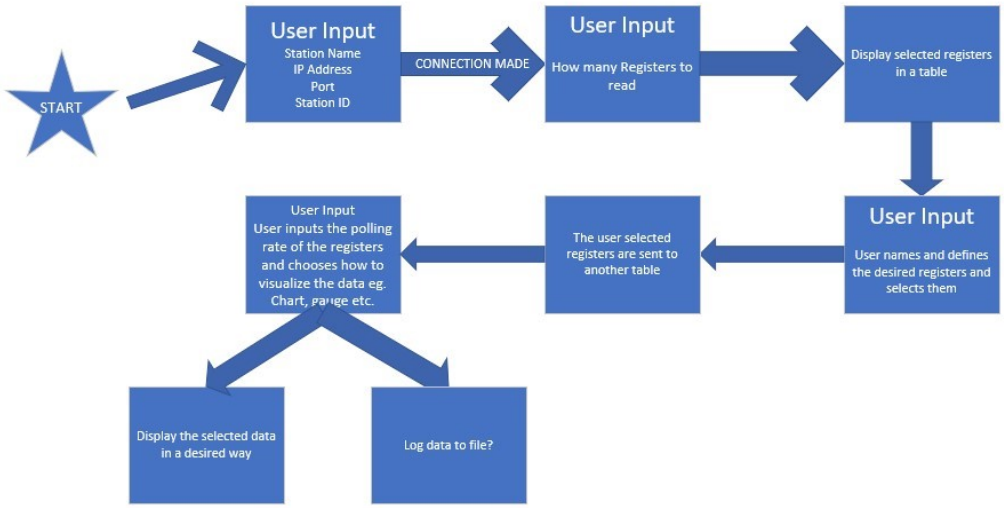
Keskustelufoorumia selaamalla ja internetistä etsimällä ei tälläkään kertaa löytynyt esimerkkiä vastaavasta valmiista sovelluksesta. Etsimällä löysin kuitenkin osia toisista ohjelmia, tai kokonaisia ohjelmia, joista voisi olla hyötyä oman näyttösovelluksen tekemisessä.

Yksi tärkeä työkalu näyttösovelluksen tekemiseen olivat erilaiset ”dashboard” solmut, joita Node-REDin solmukirjastosta löytyi. Node-RED itsessään ei ole näyttösovelluksen tekemiseen suunniteltu työkalu, vaikkakin se on sillä myös hyvin mahdollista. Näiden ”dashboard” solmujen lisäksi toiset tärkeät solmut olivat ”template” ja ”ui_template”. Näiden solmujen avuilla pystyi kirjoittamaan tavallista HTML- ja JavaScript -kieltä, ja tulos näkyisi tekemälläsi nettisivulla.

Node-RED valikoitui lopulta käytettäväksi projektissa juuri yksinkertaisuutensa ja aloittelijaystävällisyytensä ansiosta. Node-RED on myös helppokäyttöisyytensä lisäksi alustariippumaton. Tämän ansiosta valmista näyttösovellusta pystyy käyttämään millä vain alustalla, mikä valikoituukin lopulliseksi käyttöalustaksi.

6 NÄYTTÖSOVELLUKSEN SUUNNITTELU

Alustava kaaviosuunnitelma siitä millainen näyttösovelluksen toiminta voisi olla tehtiin Microsoft Visio Professionalin avulla.



Kuva 11. Suunnitelma ohjelman rakenteesta

Näyttösovelluksen suunnittelussa ja teossa on pyritty noudattamaan ja luomaan hyviä käyttöliittymän ominaisuuksia kuten selkeyttä, toiminnallisuutta ja yhdenmukaisuutta.

Web-sovelluksen ulkonäölle on Kongsberg Maritimella oma ohjesääntö, ja sitä seurattiin sovelluksen teossa.

7 NÄYTTÖSOVELLUKSEN TOTEUTUS

Näyttösovellusta tehtiin Weidmullerin kontrollerilla, sillä se sisälsi Node-REDin valmiiksi asennettuna osana sulautettua järjestelmää. Weidmüllerin kontrollerissa on myös omia IO-liityntöjä, joita on mahdollista hyödyntää myös Node-RED ohjelmassa. Lisäksi se oli yksi potentiaalisista alustoista, joilla tulevaa valmista näyttösovellusta mahdollisesti pyöritettäisiin, joten työn aikana pystyi tutustumaan laitteen ominaisuuksiin ja mahdollisiin rajoituksiin.

7.1 Käytetyt solmupaketit

node-red
node-red-contrib-modbus
node-red-contrib-modbustcp
node-red-contrib-fs
node-red-contrib-ui-media
node-red-contrib-uibuilder
node-red-dashboard
node-red-ui-table
node-red-contrib-ui-svg

7.2 Modbus-rekisterin lukeminen

7.2.1 Modbus-protokolla

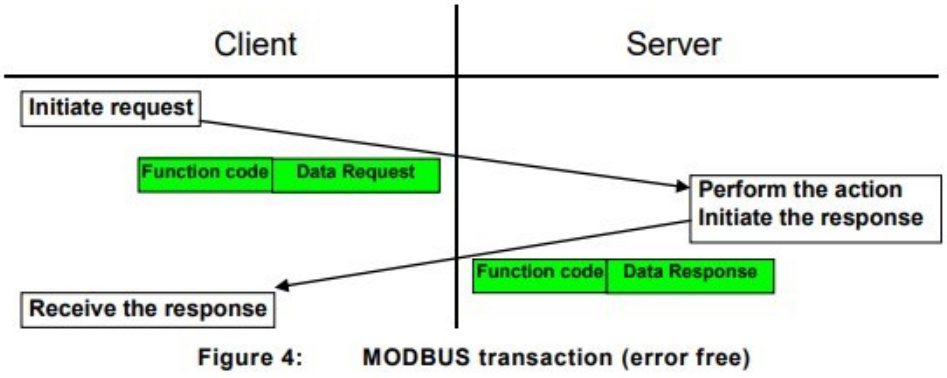
Modbus on vuonna 1979 Modiconin julkaisema sarjaliikenneprotokolla. Modbus oli alun perin tarkoitettu käytettäväksi yrityksen omien ohjelmoitavien logiikoiden (PLC) kanssa. Modbus perustuu isäntä/orja (master/slave) -malliin. Modbus protokollasta on tullut hyvin suosittu yksinkertaisuutensa, luotettavuutensa ja nopeutensa ansiosta.

Vaikka Modbus alun perin kehitettiinkin 70-luvun lopulla, on se yhä laajasti käytetty standardi teollisuudessa. Modbus on myös suosittu siksi, että se on tehty avoimeksi

käyttää ja lisenssimaksuja ole. Protokolla ei myöskään ole riippuvainen laitevalmistajasta, joten laitteet pystyvät keskustelemaan keskenään riippumatta siitä kuka laitteet on valmistanut.

Alun perin Modbus toimi sarjaporttiliitäntöjen kautta (esimerkiksi: RS232, RS485). Viime vuosina on yleistynyt Ethernet verkon kautta toimiva Modbus TCP/IP protokolla. Modbus Ethernet TCP/IP on noussut suosituksi vaihtoehdoksi helppoutensa takia, TCP/IP on jo laajalti käytössä.

Modbus TCP/IP protokollassa kaksi laitetta keskustelee keskenään isäntä/renki periaatteella.



Kuva 12. Modbus viestinvaihto (Modbus Organization, 2012.)

Laitelleelle lähetettävässä viestissä ovat sisällä haluttu funktio koodi, osoite mistä rekisterien luku aloitetaan, ja montako rekisteriä luetaan.

Primary tables	Object type	Type of	Comments
Discretes Input	Single bit	Read-Only	This type of data can be provided by an I/O system.
Coils	Single bit	Read-Write	This type of data can be alterable by an application program.
Input Registers	16-bit word	Read-Only	This type of data can be provided by an I/O system
Holding Registers	16-bit word	Read-Write	This type of data can be alterable by an application program.

Kuva 13. Modbus päätaulut (Modbus Organization, 2006.)

Laitteelle lähetettävässä viestissä määritellään Funktio koodilla, mistä taulusta tietoa halutaan lukea ja/tai kirjoittaa. Samassa viestissä lähetetään myös tieto, että mistä rekisteristä lukeminen aloitetaan, ja että montako rekisteriä siitä eteenpäin luetaan. Kuvassa 14 näkyvässä esimerkissä Funktio koodi (fc) on 4, eli viestillä halutaan lukea

input rekistereitä. ”Address: 0” kertoo sen, että rekistereitä lähdetään lukemaan nollasta lähtien, ja niitä luetaan niin monta kuin ”quantity” kohdassa määritellään. (Modbus Organization, 2006, 2012)

Kuvan 14 esimerkissä rekisterien määrä on tallennettu globaaliin muuttujaan ja luettavien rekisterien määrä määritellään muualla. Viestissä määritellään mikä on laitteen unit id, tässä tapauksessa se on ”1”. Kuvassa näkyvä ”msg.payload” on Node-RED -ohjelmissa solmusta toiseen lähetettävä viesti, joka voi sisältää erilaisia tietoja ja komentoja. Kuvassa asetetaan ”msg.payload” arvoksi objekti, joka sisältää tarvittavat komennot, jotka lähetetään Modbus laitteelle.

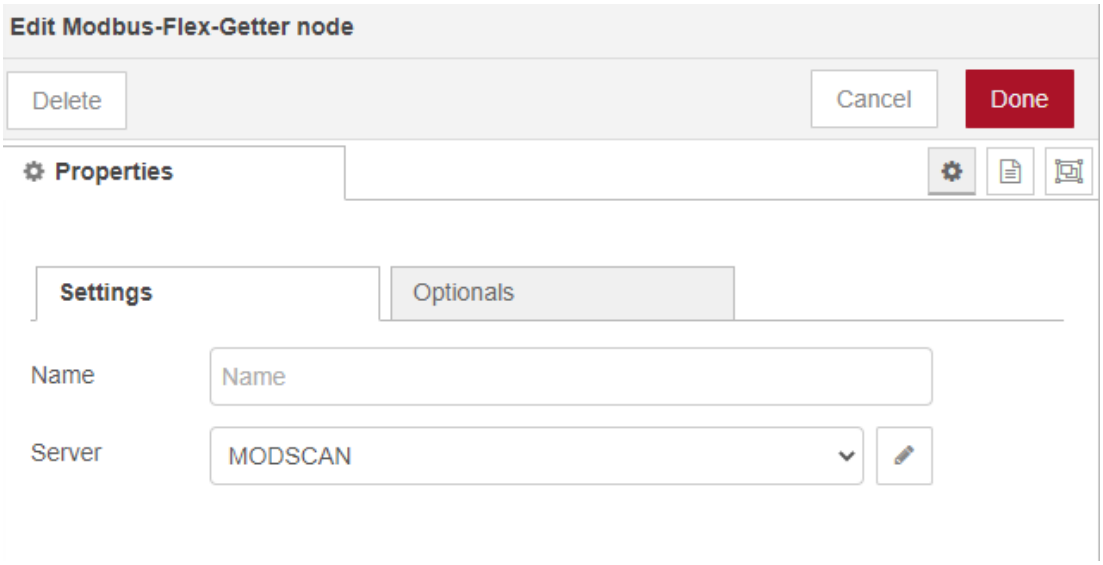
```
msg.payload = {
                value: msg.payload,
                'fc': 4,
                'unitid': 1,
                'address': 0 ,
                'quantity': global.get("QuantTemp"),
};
```

Kuva 14. Katkelma ohjelman funktio -solmusta

Tässä opinnäytetyössä käytettiin Modbus TCP/IP protokollaa.

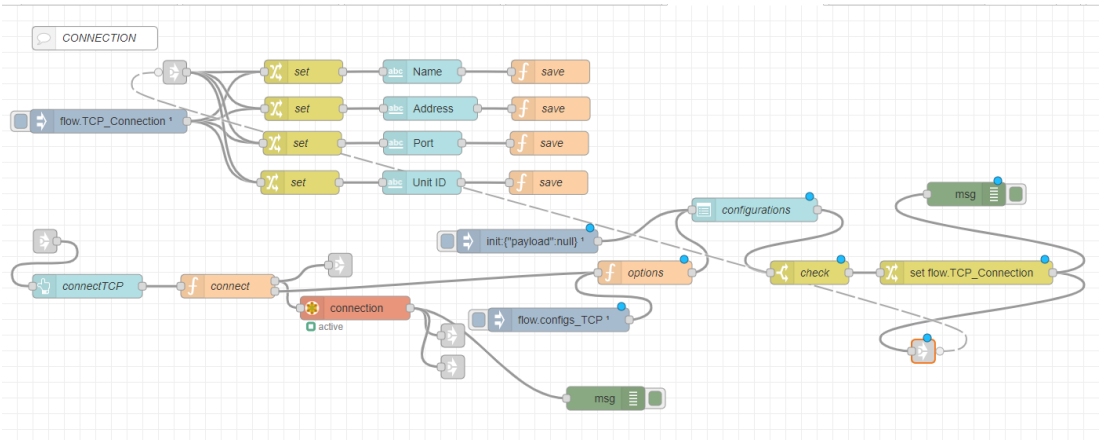
7.2.2 Yhteysasetusten antaminen

Node-RED -solmukirjastosta löytyy useampi eri vaihtoehto Modbus-rekisterien lukemiseen. Niistä monipuolisin, käytetyin ja useimmin päivitetty on node-red-contrib-modbus. Kyseinen kirjasto sisältää 13 erilaista solmua, mutta tässä sovelluksessa niistä käytetään vain kolmea: modbus-flex-connector, modbus-flex-getter ja modbus-read. Kuvassa 15 näkyy Modbus-Flex-Getter solmun asetusikkuna. Asetuksissa on mahdollista määritellä yhteysasetuksien lisäksi solmulle oma nimi.



Kuva 15. Modbus-Flex-Getter solmun asetukset

Ensimmäinen haaste näyttösovelluksen tekemisessä tuli ilmi jo tässä kohtaa. Yhteysasetukset olisi tarvinnut muuttaa solmujen sisällä jokainen kerta, kun näyttöä käytettäisiin uudessa kohteessa. Yhteysasetukset pitäisi pystyä asettamaan suoraan näyttösovelluksessa. Ongelma saatiin ratkottua modbus-flex-connector solmun avulla.



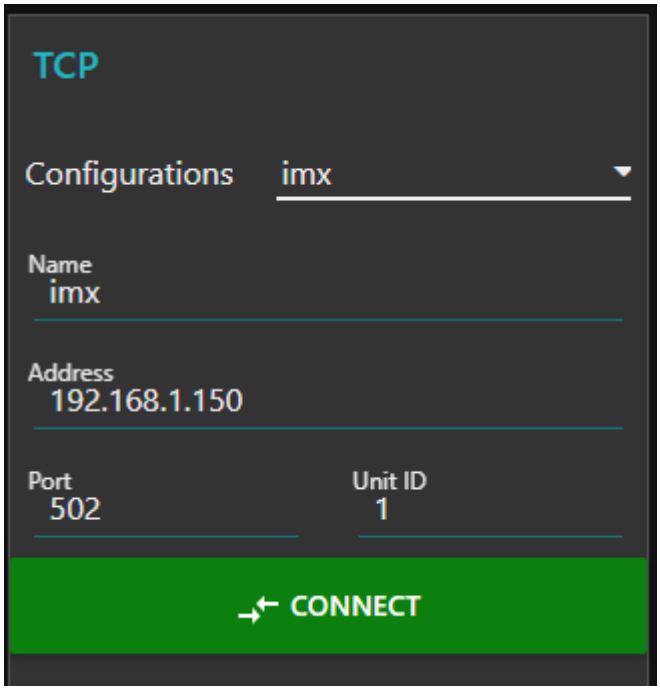
Kuva 16. Vuo, jossa käyttäjä asettaa käyttöliittymästä yhteysasetukset

Kuvassa 16. vaaleansiniset solmut ovat ”dashboard” solmuja, ja ne näkyvät käyttäjälle käyttöliittymässä. ”Name”, ”Address”, ”Port” ja ”Unit ID” ovat tekstinsyöttö kenttiä. Käyttäjä syöttää kyseisiin kenttiin vaadittavat tiedot, ja ne tallennetaan seuraavassa funktio solmussa.

Ohjelma myös tallentaa 30 sekunnin välein kaikki muuttujissa olevat tiedot laiteen paikalliseen muistiin. Ohjelman käynnistyessä tarkastetaan, onko tallennettuja tietoja

jo olemassa. Jos tietoja on, viedään ne takaisin ohjelmaan. Näin näyttö palaa takaisin normaaliin toimintaansa myös sähkökatkon jälkeen. Tietojen tallennus on mahdollista kytkeä päälle Node-REDin tiedostoissa muokkaamalla ”settings.json” tiedostoa. Tallennustaajuutta on myös mahdollista muokata samasta asetustiedostosta. Asetussivulle on luotu painike, joka tyhjentää kaikki talletetut tiedot, jotta tyhjältä pöydältä aloittaminen on helpompaa.

Kuvassa 16. alemmalla rivillä oleva ”connectTCP” solmu on kuvassa 17. näkyvä ”Connect” painike.



Kuva 17. Näkymä käyttöliittymästä, TCP yhteysasetukset

Connect painiketta painettaessa, painike lähettää käyttäjän hetki sitten kirjoittamat tiedot seuraavalle solmulle, joka on funktiosolmu nimeltään “connect” (Kuva 16.).

```
1  var options = msg.payload;
2  var error = {topic: "TCP Connect"};
3  var configs = flow.get('configs_TCP') || [];
4
5  options.connectorType = "TCP";
6
7  if(!options.name || options.name === ""){
8      options.name = "No name";
9  }
10
11 if(!options.tcpPort)
12     options.tcpPort = 502;
13
14 if(!options.tcpHost){
15     error.payload = "TCP Host address not Valid";
16     node.error("Error",error);
17     return;
18 }
19
20 var index = configs.findIndex(function(conf) {
21     return conf.name == options.name;
22 });
23
24 if(index < 0)
25     configs.push(options);
26 else
27     configs[index] = options;
28
29 flow.set('configs_TCP', configs);
30
31 node.send([msg, {payload:configs, topic:"init"}]);
```

Outputs 2

Kuva 18. Funktio solmu ”connect” joka käsittelee käyttäjän antamat tiedot, joilla yhdistetään Modbus palvelimeen

Funktion ensimmäisellä rivillä asetetaan edellisestä solmusta tullut viesti (msg.payload) ”options” nimiseen muuttujaan. Rivillä viisi samaan ”options” muuttujaan asetetaan yhteystyypiksi ”TCP”.

Riveillä 7–18 suoritetaan tarkistuksia, että käyttäjä on antanut jokaiseen kenttään sopivaa tietoa. Jos käyttäjä ei ole syöttänyt yhteydelle nimeä, asetetaan yhteyden nimeksi ”No name”. Jos taasen käyttäjä ei ole antanut porttinumeroa, asetetaan se

automaattisesti portiksi ”502”, koska se on Modbus/TCP protokollan oletusportti. Osoitekentän tyhjäksi jäädessä lähetetään solmusta virheilmoitus.

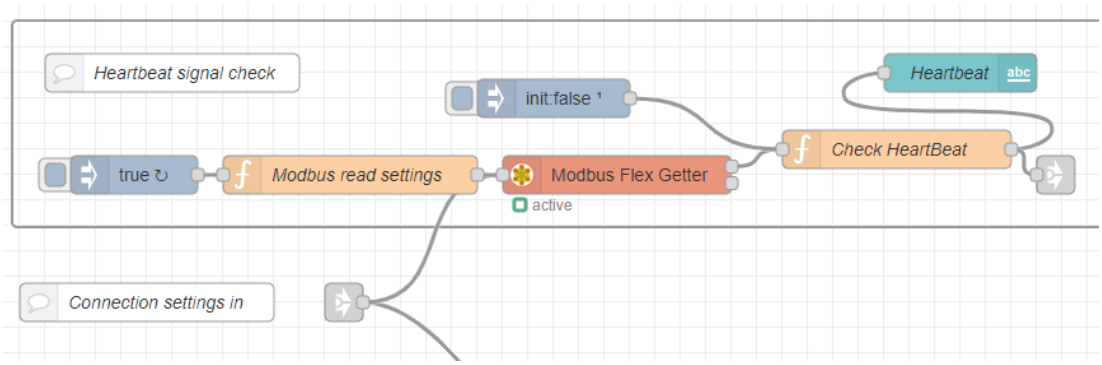
Lopussa yhteys asetukset tallennetaan, tai jos saman niminen yhteysasetus on jo olemassa, korvataan sen nimen alla olevat arvot uusilla.

Lopuksi yhteysasetukset lähetetään eteenpäin valmiina viestinä. Funktio solmun ensimmäisestä ulostulosta lähetetään viesti modbus ”connection” solmulle. Nämä yhteysasetukset viedään eteenpäin jokaiselle modbus-solmulle.

Toisesta ulostulosta yhteysasetukset lähtevät seuraaville solmuilla, missä ne tallennetaan, ja esitetään uudelleen käyttöliittymässä.

7.2.3 Heartbeat-signaalin lukeminen ja tarkistaminen

Yhteyden varmistamiseksi voidaan käyttää hyväksi IMX-8 tarjoamaa ”Heartbeat” rekisteriarvoa. Tämä rekisteri vaihtaa arvoaan joka lukukerralla. Arvo on kuitenkin aina positiivinen kokonaisluku, eikä koskaan nolla. Yhteyden voi varmistaa siis vertaamalla edellisen ja nykyisen lukukerran arvoja toisiinsa. Jos luvut eivät ole yhtä suuria, yhteys toimii oikein.



Kuva 19. Vuo, joka lukee heartbeat -rekisterin

”Modbus read settings” solmussa määritellään viesti, joka lähetetään Modbus Flex Getter -solmulle. Viestissä määritellään se mikä on luettava rekisterinumero. SKF:n IMx manuaalista käy selville, että heartbeat -rekisteri löytyy paikalta 30028. 3xxxx rekisterit ovat sisääntulorekistereitä, ja niitä pystyy ainoastaan lukemaan. Node-

REDin modbus -solmujen dokumentaatiosta selviää, että sisääntulorekistereitä luetaan käyttämällä funktio koodia 4.

Function codes (1:4) currently supported include:

- FC 1: Read Coil Status
- FC 2: Read Input Status
- FC 3: Read Holding Registers
- FC 4: Read Input Registers

Kuva 20. Funktiokoodit Node-REDissä

Funktiosolmussa osoitteeksi on määritelty 27, koska rekisterien luettelointi alkaa nollasta. Luettavien rekisterien määräksi on valittu 1, koska heartbeat-signaali on ainoa mitä halutaan lukea. Inject -solmu lähettää sekunnin välein käskyn lukea heartbeat-rekisterin.

Kun rekisteri on luettu, se viedään funktio solmuun, jossa vertaillaan uutta arvoa sekunnin takaiseen.

```
1  //Reads the heartbeat signal, and validates it by comparing it
2  //to the previous signal. Previous signal is saved in context data
3  //and compared to the new signal. The new signal must always be not
4  //equal from the previous.
5
6  temp_data = context.get("temp_data") || []  //get context data
7
8  if(msg.topic == "init"){        //if the topic is init, return msg
9      return msg;
10 }
11
12 if(temp_data != msg.payload[0]){ //checking if the previous heartbeat message is not equal to the new one
13     temp_data = msg.payload[0];
14     msg.topic = "Connected!";     //If heartbeat is ok, set topic to Connected!
15     msg.background = "Green";     //Also set the background of the button to Green
16 }else{
17     msg.topic = "Not Connected";  //if not ok, set topic to Not Connected
18     msg.background = "Grey";      //Also set the background to Grey
19 }
20
21 context.set("temp_data", temp_data);    //save context data
22
23
24 return msg;
```

Kuva 21. Ohjelmakoodi, joka tarkistaa heartbeat arvon

Ohjelman alussa haetaan ”temp_data” nimiseen muuttujaan edellisen lukukierroksen heartbeat -arvo. Riveillä 12–19 vertaillaan uutta heartbeat -arvoa vanhaan. Jos uusi arvo on erisuuruinen kuin vanha arvo, yhteys on onnistunut ja viestin otsikoksi

asetetaan ”Connected!”. Samalla käyttöliittymässä olevan ”Connect” painikkeen väri muutetaan vihreäksi.

Jos taas uusi heartbeat -arvo on sama kuin vanha, tai arvoa ei ole lainkaan, asetetaan otsikoksi ”Not Connected” ja ”Connect” painikkeen taustaväri muutetaan harmaaksi.

Arvojen vertailun jälkeen asetetaan rivillä 21 nykyinen heartbeat -arvo taas temp_data -muuttujaan. Heartbeat -lukuarvo myös esitetään käyttöliittymässä sen lisäksi että ”Connect” painike vaihtaa väriä yhteyden onnistuessa.

7.2.4 Potkurin valinta

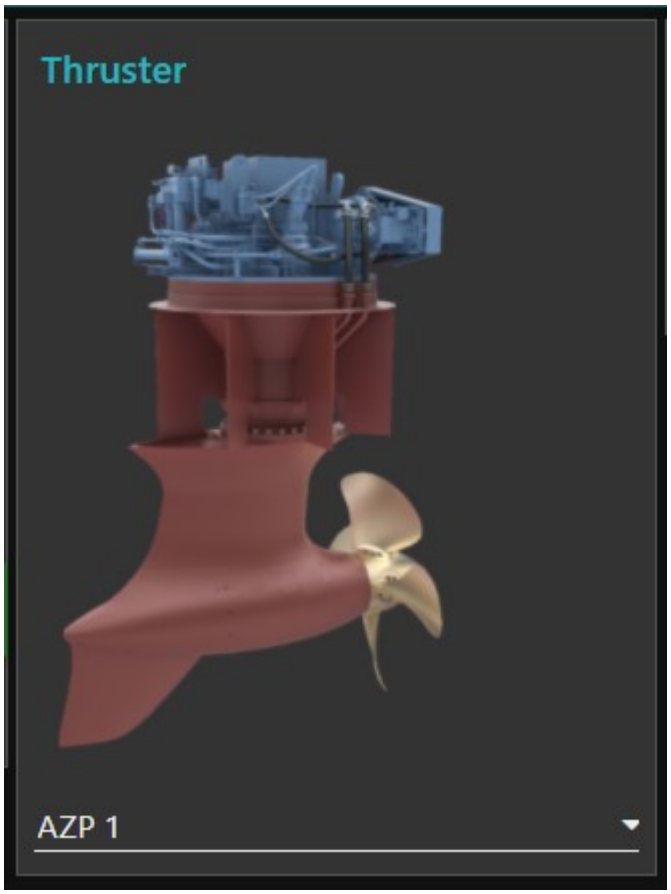
Potkurivalinnan toteuttaminen oli hyvin yksinkertaista koska solmukirjastosta löytyi juuri kaksi sopivaa solmua. Node-RED-dashboard paketin sisältä löytyvä ”ui-dropdown” solmu luo käyttöliittymään alasvetovalikon, josta käyttäjä voi valita eri vaihtoehdoista. Käyttäjä ei voi itse syöttää tämän solmun avulla tekstiä.

Toinen tarvittavista solmuista on node-red-contrib-ui-media nimisestä paketista löytyvä ”ui-media” solmu. Solmun avulla näytöllä esittää erilaisia kuvia, äänitiedostoja tai videoita.

Alasvetovalikkoon on ennaltamääritelty lista potkureista. Kun käyttäjä valitsee listasta potkurin, kyseisen potkurin kuva vaihtuu näkyviin näytölle. Käyttäjän potkurivalinta välitetään myös toiselle sivulle. Potkurin voi kuitenkin valita listasta ainoastaan ”Settings” sivulla.

”Ui-media” solmuun voi ladata kuvia tietokoneelta, ja näytettävä kuva vaihtuu sillä perusteella, millainen viesti kyseiseen solmuun lähetetään. Viestin tarvii olla tarkasti muotoa ”Category/file name”. Kategoria pysyy samana jokaisessa vaihtoehdossa, mutta riippuen käyttäjän valinnasta viestissä oleva potkurin nimi vaihtuu.

Pudotusvalikossa tehty valinta viedään myös viimeiselle näyttösivulle ”Link out” solmun avulla.



Kuva 22. Käyttöliittymän näkymä potkurilaitteen valinnasta

7.2.5 Taulukko

Käyttäjälle näkyvä taulukko, jossa luetut Modbus rekisterit näytetään, on Node-REDin dashboard kirjastosta löytyvä ”ui_table” niminen solmu. Kyseiseen solmuun voi määritellä rivejä ja sarakkeita joko itse solmun asetuksista, tai sitten taulukon asetukset voidaan tuoda taulukolle viestissä.

Taulukosta voidaan myös halutessaan tehdä monilla eri tavoilla muokattava. Sarakkeista esimerkiksi voidaan tehdä siirreltäviä, riveistä poistettavia tai valittavia. Rivien valinta soveltuu hyvin tähän käyttötarkoitukseen ja se otettiinkin käyttöön taulukkoon. Käyttäjä voi valita käyttöliittymän painikkeen avulla ovatko rivit valittavissa vai ei. Taulukon parametreissä pystytään määrittelemään jokaiselle eri sarakkeelle erilaisia ominaisuuksia. Esimerkiksi määrittämällä ”editor” parametriksi

”input”, pystyy käyttäjä syöttämään kyseiseen sarakkeeseen itse tekstiä. Samalla tavalla toimii ”Measurement” sarakkeen alasvetovalikko; parametriksi annetaan ”select” ja lisäparametreina annetaan lista alasvetovalikkoon tulevista vaihtoehdoista.

7.2.6 Rekisterien lukeminen ja esittäminen taulukossa

Ennen kuin voidaan lähteä lukemaan rekistereitä, käyttäjän tarvitsee määritellä, että kuinka monta rekisteriä halutaan lukea. Tämä onnistuu yksinkertaisesti käyttämällä ”ui_text_input” nimistä solmua, joka luo käyttöliittymään tekstikentän, johon käyttäjä kykenee syöttämään tekstiä. Tekstin lisäksi solmun voi määritellä hyväksymään vain numeroita, tai vaikka sähköpostiosoitteita.

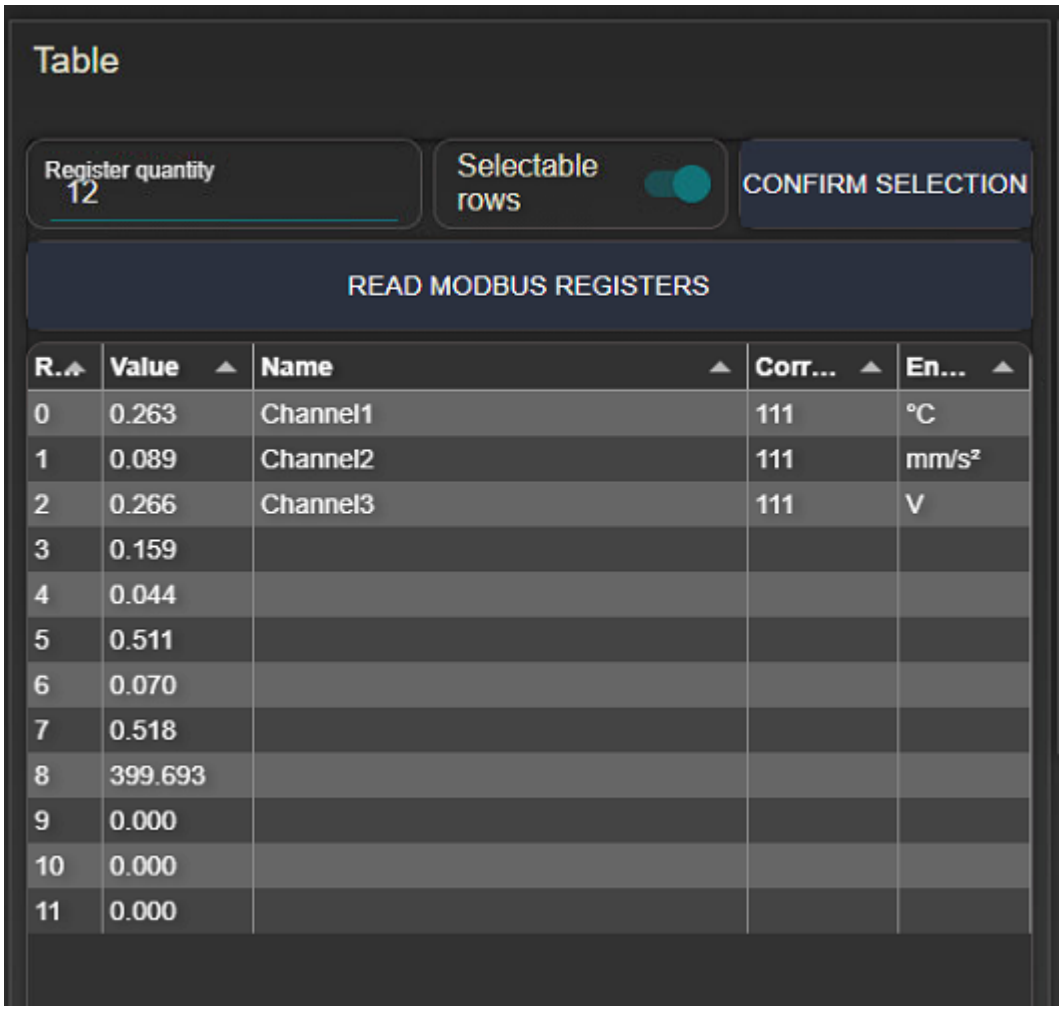
Modbus -viestissä määritellään, että rekisterien lukeminen aloitetaan nollasta, ja että niitä luetaan käyttäjän määrittelemä määrä. Ennen taulukkoon viemistä, käsitellään dataa muuttamalla se luettavaan desimaalimuotoon ja pyöristetään kolmen desimaalin tarkkuuteen. Data muunnetaan kahdesta 16-bittisestä numerosta yhdeksi 32-bittiseksi Float -numeroksi.

Tämän jälkeen tiedosta kasataan JSON-viesti, jotta data pystytään esittämään taulussa helposti.

```
1  //Formats the data to a form that can be displayed in a table
2
3  var Value,SensorNmb,Position;
4  var temp = [];
5
6  msg.payload.forEach(myFunction) //do for each member of an array
7
8  function myFunction(item, index, arr) {
9
10
11     arr[index] = SensorNmb = index;     //get the SensorNmb from the current index
12     arr[index] = Value = item;          //get the Value from the current index
13     arr[index] = Position = "";         //Set Position to empty string.
14     msg.payload = ({"textValue":SensorNmb,
15                     "numberValue":Value});      //Create an object from the variables
16
17     temp.push(msg.payload);             //push the created object to the temp array
18     return temp,msg;                    //return the temp string
19
20 }
21
22 msg.payload = temp;                     //set the temp array to the payload
23
24
25 return msg;                             //return the msg
```

Kuva 23. Funktiosolmu, jolla luettu Modbus data muunnetaan JSON -viestiksi

Käyttäjän syötettyä luettavien rekisterien lukumäärän, kerrotaan luku ensin kahdella, jotta saadaan oikea määrä luettavia rekistereitä. Tämän jälkeen lukumäärä tallennetaan globaaliin muuttujaan, jotta se voidaan käyttää modbus -viestin rakentamisessa.
Kun yhteys on muodostettu, ja luettavien rekisterien lukumäärä valittu, voidaan lukea Modbus-rekisterit taulukkoon. Käyttäjän painaessa ”READ MODBUS REGISTERS” painiketta, taulukko luo automaattisesti rivin jokaiselle rekisterille, ja näyttää kyseisen rekisterin arvon.



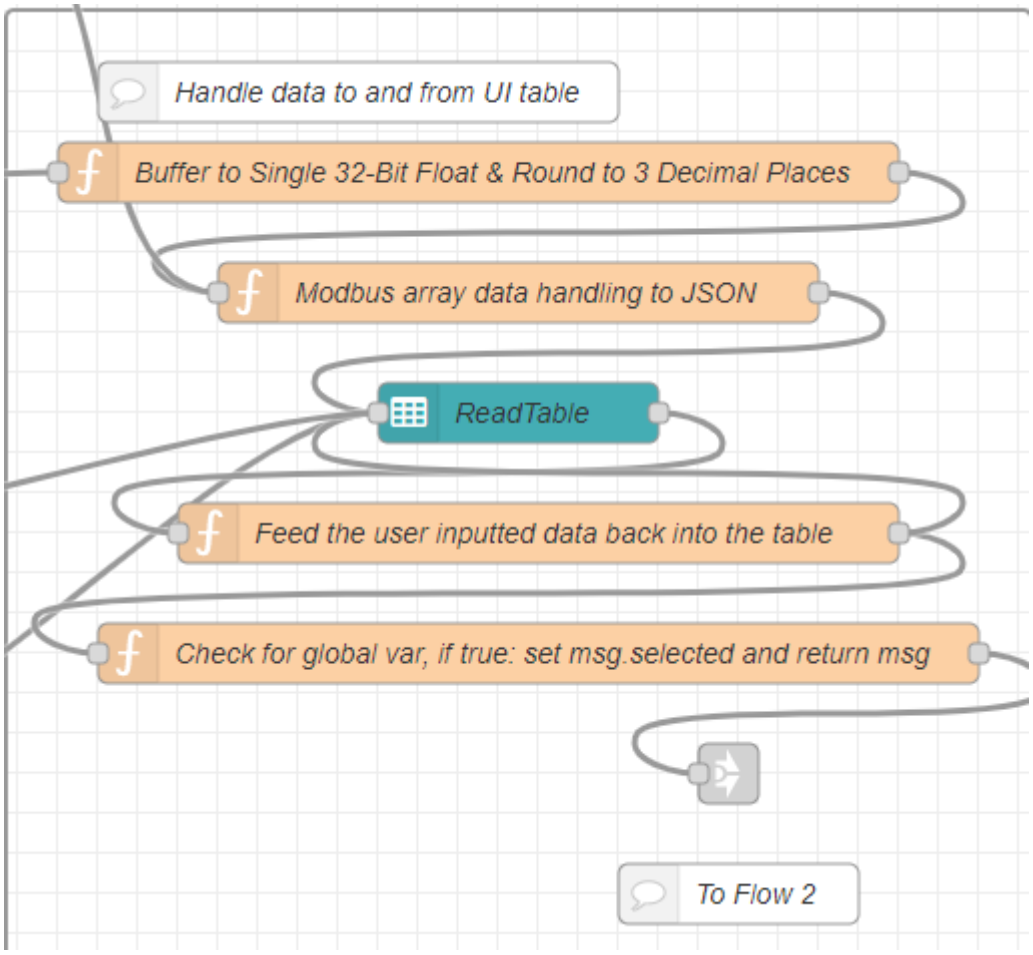
Kuva 24. Taulukko, johon on luettu 12 rekisterin arvot

”Name”, ”Correction” ja ”Engineering Unit” kentät ovat käyttäjän muokattavissa. ”Name” kenttään syötetään anturin nimi, tai sen paikka kentällä. Vaihtoehtoja ovat kiihtyvyys, nopeus, kosteus tai paine. ”Correction” kenttään syötetään numeerinen arvo, jonka avulla saadaan muunnettua arvo oikeaksi. ”Engineering Unit” kenttää painamalla aukeaa alasvetovalikko, josta voi valita halutun yksikön kyseiselle arvolle.

Taulukkoon syötetään haluttuihin rekistereihin tarvittavat tiedot. Kun kaikki halutut tiedot on syötetty taulukkoon, valitaan rivit aktivoimalla rivien valinta taulukon yläpuolella olevan ”Select rows” painikkeen avulla. Painikkeen painamisen jälkeen, rivit muuttuvat valittaviksi.

Taulukkoihin syötettäviä arvoja syötetään koko ajan myös takaisin samaan taulukkoon. Tämä johtuen siitä, että aina kun taulukkoon tuodaan uusi parametri,

esimerkiksi se, että rivit ovat nyt valittavissa, tapahtuma pyyhkii kaikki taulun sarakkeet.



Kuva 25. Taulukko ja sitä ympäröivät funktio solmut

Kun kaikki halutut rivit ovat valittuja, lähetetään valinnat eteenpäin painamalla taulukon yläpuolella olevaa ”SEND SELECTED” painiketta. Ainoastaan valittuihin riveihin asetetaan myös viesti ”selected”. Viimeisessä funktio solmussa tarkastetaan, että jos tämä viesti on asetettu, jatkaa viesti matkaansa seuraavaan vuohon. Jos ”selected” viestiä ei ole asetettu, kyseinen rivi ei pääse seuraavaan vuohon.

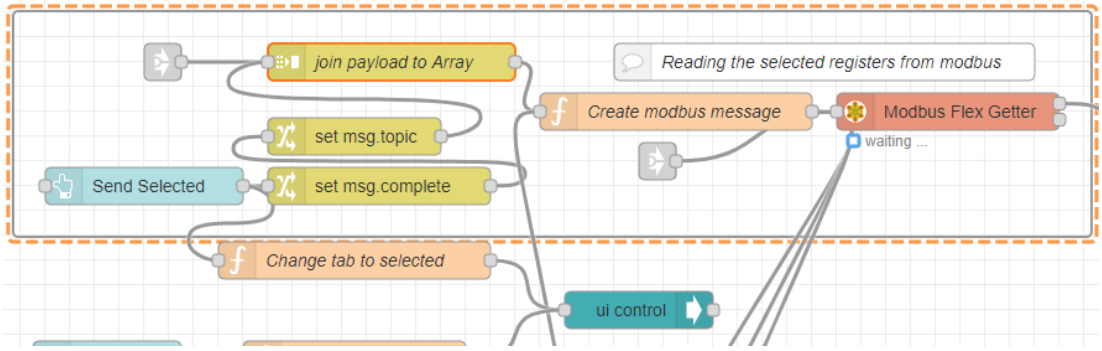
Asetussivulla on myös ”Polling Rate” -lomake, johon käyttäjä syöttää halutun lukuvälin sekunteina. Kun lomakkeeseen on annettu haluttu lukuaika, esimerkiksi 5 sekuntia, ohjelma lukee Modbus-rekisterit 5 sekunnin välein.

Jos käyttäjä palaa asetussivulle sen jälkeen, kun on jo kerran valinnut rekisterit, tuodaan taulukkoon sivunlatauksessa jo tehdyt valinnat, jotka ovat talletetut muuttujaan. Sivun latauksessa tapahtuvia asioita on mahdollista luoda ”ui_control” solmulla, joka mainitaan myös myöhemmin.

7.3 Valittujen rekisterien esitys ja päivitys

7.3.1 Valittujen rekisterien esitys taulukossa

Ensimmäisen taulukon ”Send Selected” painiketta painettaessa valitut rekisterit lähetetään eteenpäin ja samalla välilehti vaihdetaan automaattisesti ”Selected” välilehdelle.



Kuva 26. Vuo, jossa valitut rekisterit kerätään yhteen

”Send Selected” näppäintä painettaessa lähtee samanaikaisesti viesti kahdelle eri solmulle. Toinen solmuista on funktiosolmu, joka lähettää viestin ”ui control” solmulle vaihtaa välilehteä. ”Ui control” solmu vaihtaa käyttöliittymän välilehteä, kun sille lähetetään viesti muotoa ”{’tab’:’my_tab_name’}”.

Ylempänä oleva ”join payload to Array” -niminen solmu kerää tulevia viestejä niin kauan kunnes saa viestin ”msg.complete”. Tämän jälkeen kyseinen solmu lähettää kaikki kasatut viestit yhtenä viestinä eteenpäin. ”Send selected” näppäintä painettaessa ”Change” solmu asettaa ”complete” viestin arvoon true. Tämän jälkeen tulee toinen ”Change” solmu, jossa asetetaan viestin otsikoksi ”load”. Tämä sen vuoksi, että myöhemmässä vaiheessa on helpompi erottaa eri viestit toisistaan. Tämän jälkeen viesti lähtee viimein eteenpäin ”Create modbus message” nimiseen solmuun, missä taas kasataan viesti Modbus kyselyä varten.

Funktion sisällä tarkastetaan ensin mikä on viestin otsikko, ja sen jälkeen rakennetaan Modbus-viesti. Modbus-rekistereitä luetaan rakennetussa viestissä sama määrä kuin aikaisemminkin. Tämä johtuu siitä että ”Modbus flex getter” solmussa ei ole mahdollista määritellä yksitellen mitkä rekisterit luetaan. On mahdollista vain määrittää alkurekisteri, mistä lähtien luetaan, ja luettavien rekisterien määrä. Ylimääräiset rekisterit poistetaan myöhemmin vertaamalla valittuja rekistereitä luettuihin rekistereihin. Taulukkoon viedään ainoastaan valitut rekisterit, mutta päivitetyillä arvoilla.

Kun rekisterit ovat taas luetut, käsitellään ne aluksi samalla tavalla kuin aikaisemmin. Luvut muutetaan yhdeksi 32-bittiseksi luvuksi ja pyöristetään kolmeen desimaalilukuun.

Seuraavaksi tarkistetaan, onko rekisteriarvoihin päätynyt kaksoiskappaleita joistakin rekistereistä. Se on mahdollista, jos käyttäjä on valintasivuilla valinnut yhden rekisterin, mutta lopulta päättänyt olla ottamatta sitä mukaan. Tällaisessa tilanteessa ohjelma on tulkinnut, että rekisteri on valittu kahdesti. Tarkastuksessa siis käydään läpi jokainen valittu rekisteri ja verrataan niitä muihin valittuihin. Jos rekisteri on uniikki, ei tehdä mitään. Jos samaa rekisteriä löytyy kaksi, poistetaan molemmat instanssit kyseistä rekisteriä.

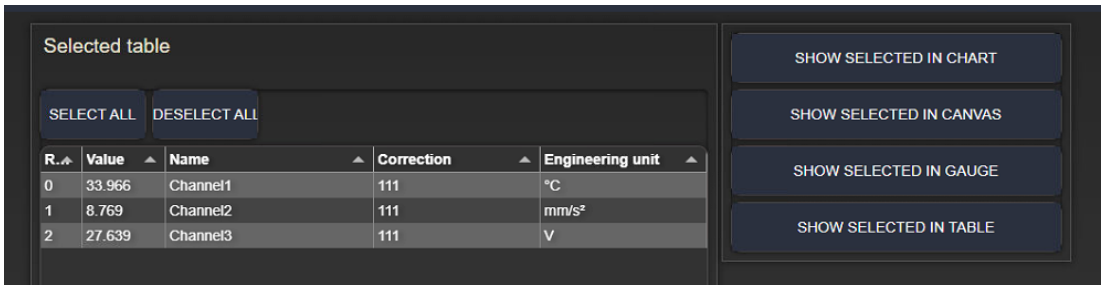
```
msg.tempvalue = msg.payload;
var myArr = msg.modbusRequest.value;
var newArr = myArr;

//Checks if there are any duplicate values in the array
//Loops through every item in the array and compares the textValue to other items in the array
//If a duplicate value is found, 1 is added to the foundCount variable
for(var h = 0; h < myArr.length; h++) {
    var curItem = myArr[h];
    var foundCount = 0;
    // search array for item
    for(var i = 0; i < myArr.length; i++) {
        if (myArr[i].textValue == myArr[h].textValue)
            foundCount++;
    }
//If the foundCount variable is larger than 1, enter the if and loop through the array
//eliminating duplicate registers. Only one instance of any registers will be left in
//the array.
    if(foundCount > 1) {
        // remove repeated item from new array
        for(var j = 0; j < newArr.length; j++) {
            if(newArr[j] == curItem) {
                newArr.splice(j, 1);
                myArr.splice(j, 1);
                j = j - 1;
            }
        }
    }
}
```

Kuva 27. Kaksoiskappaleiden tarkastus ja poisto

Kun kaksoiskappaleet on saatu poistettua, asetetaan arvot uudelleen viestiin, jotta ne on helposti vietävissä taulukkoon näkyville. Vielä ennen lopullisesti taulukkoon viemistä, muokataan modbus arvoja hieman. Kaikki arvot kerrotaan niille annetulla korjauskertoimella. Näin arvoista saadaan muokattuja haluttuja suureita, tai saadaan korjattua arvoa halutulla tavalla muuten.

Taulukko itsessään on toteutettu samalla tavalla kuin aikaisemmalla asetussivulla. Taulukolle tuodaan samalla tavalla parametrit eri sarakkeita varten. Tällä kertaa taulukon arvoja vain ei ole mahdollista enää muokata, eikä rivejä voi enää tehdä valittavaksi.

Ensimmäisen taulukon jälkeen kerätään valitut rekisterit ja ne esitetään toisessa taulukossa. Tässä taulukossa nimiä tai muita arvoja ei voi enää muokata.



Kuva 28. Välilehti, jossa esitetään valitut rekisterit ja valitaan esitystapa

Halutessaan käyttäjä voi vielä palata aikaisempaan ”Settings” välilehteen muuttamaan annettuja nimiä, mittauksia tai ”Correction” korjausarvoa.

7.3.2 Modbus arvojen päivitys

Se kuinka usein Modbus-rekisterit luetaan, määritellään käyttöliittymässä käyttäjän toimesta. Käyttöliittymässä on kenttä, johon syötetään haluttu päivitystaajuus millisekunteina. Kyseinen kenttä on toteutettu ”dashboard” -kirjastosta löytyvällä ”ui_form” -solmulla. Käyttäjän antama arvo viedään viestinä seuraavaan solmuun, joka on funktio solmu. Funktio solmussa olevan JavaScript -koodin avulla lähetetään annetun päivitystaajuuden välein viesti otsikolla ”update”. Vuossa myöhemmin tuleva

funktiosolmu kasaa ja lähettää aina uuden Modbus -pyynnön saadessaan viestin, jonka otsikkona on ”update”.

```
1  //User sets the interval from the dashboard, and the node will send a message forward in a set interval
2
3  msg.topic = "update"            //setting the topic to update to tell the messages apart later
4  let timer = context.get('intervalFunction')
5  if (timer) clearInterval(timer)
6
7  timer = setInterval(function() {
8      msg.topic = "update"
9      node.send(msg)
10 }, msg.payload.Rate);
11
12 context.set('intervalFunction', timer)
13
14 return msg;
```

Kuva 29. Intervallifunktio, lähettää viestin ”update” annetun päivitystaajuuden välein

7.4 Tietojen keräys muistikortille ja tiedostoselain

Weidmueller UC20-WL2000-AC sisältää mahdollisuuden liittää laitteeseen SD-muistikortin. Tällä muistikortilla on mahdollisuus laajentaa laitteen omaa muistikapasiteettia. Laite tukee enimmissään 32 Gigatavun muistikorttia.

Muistikortille on myös mahdollista tallentaa, ja myös noutaa, Node-REDin avulla erilaisia tiedostoja. Näin on mahdollista myös tallentaa myös ohjelman avulla kerättävää tietoa suoraan muistikortille.

7.4.1 Tiedostoselain käyttöliittymään

Node-REDin kotisivuilta löytyy monenlaisia esimerkkejä moneen eri käyttötarkoitukseen. Sieltä löytyi myös hyvä pohja Node-RED -käyttöliittymässä olevaan tiedostoselaimeen. Esimerkin sai toimimaan omassa sovelluksella hyvin vähäisellä muokkauksella. (File browser in dashboard, 2020.)

Suurin osa tiedostoselaimen toteutuksessa käytetyistä solmuista on jo aikaisemmin käytettyjä ja mainittuja solmuja tutuista kirjastoista. Uutena solmuna tulee ”node-red-contrib-fs” kirjastosta tuleva ”fs-file-lister” niminen solmu. ”Fs-file-lister” solmu etsii

yhden valitun kansion, ja halutessaan myös kyseisen kansion alakansiot, tiedostojen varalta.

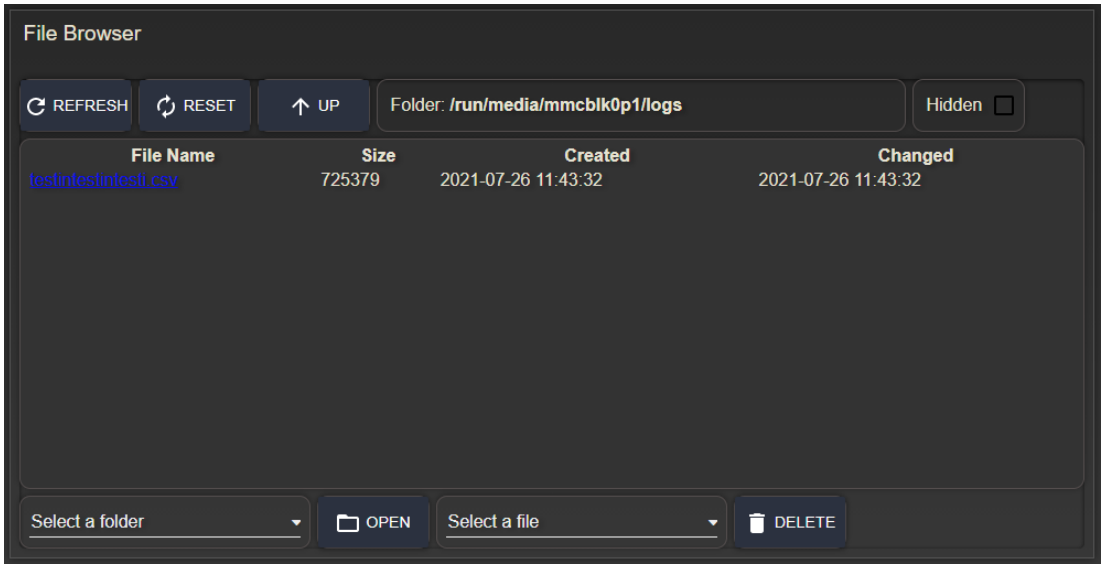
Selitettynä tämä tiedostoselain- vuo toimii niin, että ohjelmalle on annettu aloitushakemisto, joka näkyy aina oletuksena ensimmäisenä, kun kyseisen välilehden avaa. Tässä tapauksessa kyseinen hakemistopolku on ”/run/media/mmcblk0p1/logs”. Kyseisessä polussa ”mmcblk0p1” on laitteen muistikortti, ja sen jälkeen tuleva ”logs”, on kansio, joka on luotu kyseiselle muistikortille tiedostojen keräystä varten.

Käyttöliittymässä on erilaisia painikkeita, joiden avulla käyttäjä voi liikkua tiedostojärjestelmässä. ”Refresh” painikkeella voidaan päivittää kyseinen kansio, ja tarkastaa onko sinne ilmestynyt uusia tiedostoja, tai onko tiedostoja poistunut. ”Reset” painike vie käyttäjän takaisin määriteltyyn oletuspolkuun. ”Up” painikkeella käyttäjä voi siirtyä yhden kansion ”ylöspäin”. Esimerkiksi jos käyttäjä painaa oletuspolussa ”Up” painiketta, siirtyy tiedostoselain kansioon ”/run/media/mmcblk0pi/”, eli yhden kansion ylöspäin. Tämän lisäksi on vielä ”Hidden” painike, jonka avulla voi valita näytetäänkö piilotettuja tiedostoja vai ei.

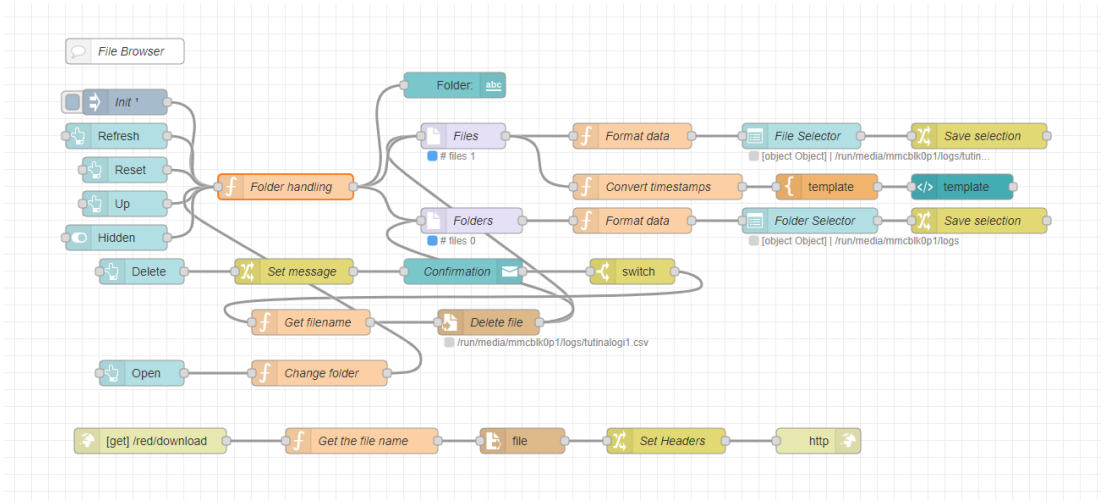
Tiedostoselaimen alalaidassa on alasvetovalikko, jota painamalla aukeaa lista kansoista, joihin on mahdollista siirtyä nykyisestä kansiosta. Esimerkiksi jos käyttäjä on hakemistossa ”/run/media/mmcblk0p1/”, alasvetovalikosta löytyy kansio ”logs”. Kyseiseen kansioon voidaan siirtyä valitsemalla valikosta tämä kansio ja painamalla alasvetovalikon vieressä olevaa ”Open” -painiketta. Tämän jälkeen tiedostoselain siirtyy kyseiseen kansioon. Tiedostoselaimen yläreunassa näkyy tekstinä koko ajan missä kansiossa milloinkin ollaan.

Alareunassa on myös toinen alasvetovalikko. Tämän alasvetovalikon avulla voidaan valita kyseisessä kansiossa olevia tiedostoja. Kun listasta on valittu jokin kyseisessä kansiossa olevista tiedostoista, voidaan painaa valikon vieressä olevaa ”Delete” -painiketta. Tämä toiminto poistaa valitun tiedoston. Ennen tiedoston poistamista, kysytään käyttäjältä vielä varmistusikkunassa, että kyseinen tiedosto halutaan varmasti poistaa.

Tiedostoja on myös mahdollista ladata käyttöliittymän kautta. Tiedostoselaimessa näkyviä tiedostonimiä painamalla ne pystytään lataamaan laitteelle, jolla käyttöliittymää käytetään.



Kuva 30. Tiedostoselain sellaisena kuin se näkyy käyttöliittymässä

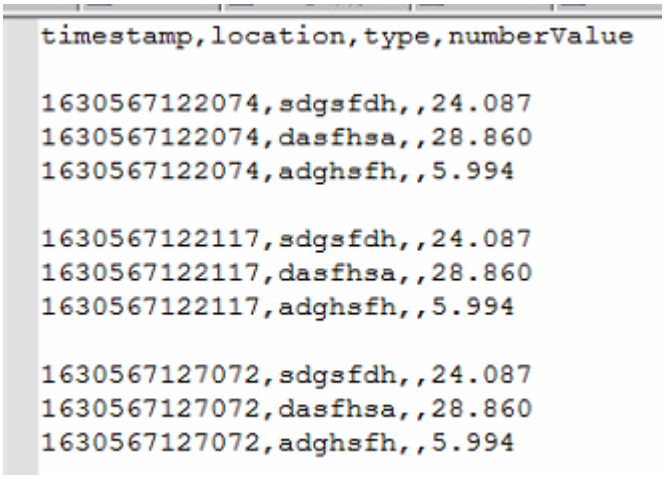


Kuva 31. Vuo, joka luo käyttöliittymään tiedostoselaimen

7.4.2 Tiedon keräys CSV-tiedostoon ja tallennus

CSV on tiedostomuoto, johon on helppo tallettaa yksinkertaista taulukkomuotoista tietoa. CSV-nimi on lyhenne sanoista ”Comma-Separated Values”, eli suomeksi

pilkulla erotellut arvot. Tämä nimi myös kuvaa hyvin tiedoston rakennetta. Se sisältää tekstimuodossa pilkuilla ja rivinvaihdoilla eroteltuja arvoja.



```
timestamp,location,type,numberValue

1630567122074,sdgsfdh,,24.087
1630567122074,dasfhsa,,28.860
1630567122074,adghsfh,,5.994

1630567122117,sdgsfdh,,24.087
1630567122117,dasfhsa,,28.860
1630567122117,adghsfh,,5.994

1630567127072,sdgsfdh,,24.087
1630567127072,dasfhsa,,28.860
1630567127072,adghsfh,,5.994
```

Kuva 32. Esimerkki CSV tiedoston sisällöstä

CSV tiedostot ovat yksinkertaisuutensa vuoksi helposti käytettävissä. Usein ensimmäisellä rivillä olevat tiedot kertovat kyseisen sarakkeen sisällön (Kuva 32.). CSV tiedostoja on helppo avata muokattavaksi ja nähtäväksi esimerkiksi Microsoft Excelissä.

Node-REDin vakiosolmuista löytyy solmu nimeltään ”csv” joka kykenee muuttamaan merkkijonon CSV tiedostoksi, tai vaihtoehtoisesti toisinpäin. Samalla tavalla Node-RED asennuksen mukana suoraan tuli ”file” niminen solmu, joka kykenee kirjoittamaan tiedostoja laitteen muistiin. Solmu kykenee joko kirjoittamaan jo olemassa olevaan tiedostoon lisää, tai luomaan kokonaan uuden tiedoston, jos annetun nimistä tiedostoa ei vielä ole olemassa kyseisessä hakemistossa.

Näiden kahden solmun avulla on jo mahdollista kirjoittaa tiedostoja muistikortille. Tiedostolle on mahdollista määritellä tiedostonimi asettamalla ”file” solmulle menevään viestiin ”msg.filename” parametrille jokin nimi. Käyttöliittymän asetus sivulle asetettuun keskikenttään on mahdollista keksiä tiedostolle nimi. Jos tiedostonimeä ei syötetä, tietoja ei myöskään kerätä eikä kirjoiteta muistikortille.

Käyttäjän antamaa tiedostonimeä pitää hieman muokata ennen kuin se lähetetään ”file” solmulle. Tiedostonimelle pitää antaa absoluuttinen tiedostopolku, joten nimen

eteen lisätään tarvittavat kansionimet, sekä tiedoston perään lisätään CSV tiedostotyypin pääte ”.csv”. Kenttä mihin käyttäjä syöttää halutun tiedostonimen on taas toteutettu ”form” solmulla, joka on mainittu aiemmin. Valmis tiedostonimi tallennetaan globaaliin muuttujaan ja liitetään viestiin jokaiseen viestiin, joka tuodaan ”file” solmulle. (Kuva 33.) Käytännössä sama viesti menee taulukkoon, jossa näytetään valitut rekisterit, sekä ”csv” solmun kautta ”file” solmulle.

```
1  //Compose a filename from user input. The filename must contain a full absolute filepath
2  //so the path, and the filetype is added to the user inputted name.
3
4  msg.payload = msg.payload.filename;
5
6  temp1 = "/run/media/mmcblk0p1/logs/";
7  temp2 = ".csv";
8
9  temp3 = temp1 + msg.payload + temp2;
10 global.set("fname",temp3);
11
12
13 return msg;
```

Kuva 33. Tiedostonimi ohjelmakoodi

Käyttöliittymän tiedostoselaimen tiedostojen latauksen lisäksi CSV-tiedostojen tarkastelu on mahdollista irrottamalla SD-muistikortti laitteesta ja liittämällä tietokoneeseen tiedostojen tarkastelua varten.

7.5 Sovelluksen ulkonäkö ja muoto

Näyttösovelluksen ulkonäkö on muokattavissa tiettyyn rajaan asti suoraan Node-REDin käyttöliittymästä. Rajan tullessa vastaan, ulkonäköä on mahdollista muokata lisää käyttämällä myös CSS dokumentilla. CSS tulee sanoista Cascading Style Sheets ja tarkoittaa erilaisia tyyliohjeita, joiden avulla voidaan muokata HTML- sivuston ulkonäköä eri tavoin.

Yhteen CSS tiedostoon voi kerätä useita eri sääntöjä millainen ulkonäön pitää olla. Sääntöjen avulla voi määrittää esimerkiksi elementtien värejä, muotoa, kokoa, periaatteessa kaikkea mitä nettisivulla näkyy, kyetään muuttamaan CSS sääntöjen avulla. CSS säännöt ovat kuitenkin vain ehdotuksia, miltä mikäkin elementti näyttää.

Uudet säännöt voivat korvata vanhat määritykset, tai ne on myös mahdollista kiertää kokonaan.

Kongsberg Maritimella on oma ohjesääntö sille, miltä näyttösovelluksen kuuluu näyttää. Ohjeessa on määritelty käytettävät värit, fontti, eri elementtien ulkonäkö ja kaikki muu mahdollinen. Tämän näyttösovelluksen ulkonäkö on muokattu ohjeen mukaiseksi mahdollisimman hyvin.

CSS tyylitiedostoja on mahdollista käyttää Node-RED näyttösovelluksessa käyttämällä ”dashboard” kirjastossa mukana tulevaa ”ui_template” solmua. Solmun sisälle voi suoraan kirjoittaa joko HTML-, JavaScript- tai CSS-kieltä. Solmussa määritellään ensin, että mihin kohtaan näyttösovellusta kyseinen koodi sijoitetaan. Se voidaan sijoittaa johonkin olemassa olevaan ”layout” ryhmään, tai vaihtoehtoisesti sivuston ”<head>” osioon, jossa yleensä annetaan sivuston tyylitiedot.

```
127 .nr-dashboard-theme .nr-dashboard-button .md-button {
128     background-color: #2B313F;
129 }
130
131 body.nr-dashboard-theme md-toolbar .md-toolbar-tools{
132     background-color: #2B313F;
133     color: #ffffff;
134
135 }
136
137
138 .nr-dashboard-theme .nr-dashboard-button .md-button:hover {
139     background-color: #58677A;
140 }
141
```

Kuva 34. CSS sääntöjä, joilla asetetaan erilaisia värejä eri elementeille

Joka kerran, kun nettisivu ladataan, latautuu myös sivuston ”head” osio, ja asettaa kaikki ulkonäköasetukset.

Samalla ”ui_template” solmulla lisättiin myös sivuston ylälaitaan Kongsbergin logo, päivämäärä ja kellonaika. (Kuva 35.) Näiden lisäys tosin tapahtui JavaScriptiä käyttämällä koska pelkkien tyylisääntöjen avulla tämä ei ollut mahdollista. Tämä JavaScript -koodi on sijoitettu samalla tavalla sivuston ”head” osioon, joten se ajetaan

jokainen kerta, kun nettisivu ladataan tai päivitetään. Logon lisäys toimii samalla periaatteella, kun alla oleva päivämäärän ja kellonajan lisäys.

```
<script id="titleScript" type="text/javascript">

$(function() {
    if($('.md-toolbar-tools').length != 0){
        loadClock();
    }else setTimeout(loadClock, 500)
});
function loadClock(){
    $('#clock').remove();
    var toolbar = $('.md-toolbar-tools');
    var div = $('<div/>');
    var p = $('<p/ id="clock">');

    div.append(p);
    div[0].style.margin = '5px 5px 5px 5px';
    toolbar.append(div);

    function displayTitle(lh) {
        p.text(lh);
    }
    function upTime() {
        var d = new Date();
        p.text(d.toLocaleString());
    }
    if(document.clockInterval){
            clearInterval(document.clockInterval);
            document.clockInterval = null;
    }

    document.clockInterval = setInterval(upTime,1000);
}
```

Kuva 35. JavaScript -ohjelma

7.6 Tiedon esittäminen

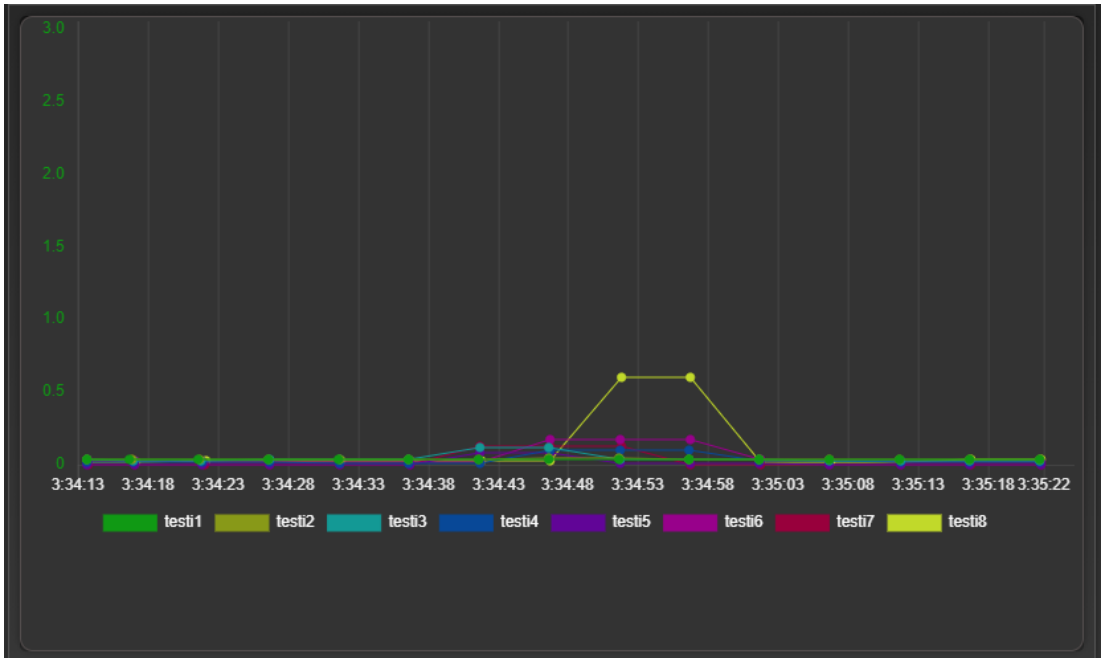
Ohjelman ideana on, että kun halutut arvot ovat valittuina taulukossa, piirtää ohjelma automaattisesti näytölle erilaisia visualisointeja, joista on helppo seurata kyseisiä arvoja ja prosesseja.

7.6.1 Kuvaaja

Arvoja esitetään kuvaajassa ajan funktiona. Kuvaaja on rakennettu samalla ”ui_template” solmulla, jolla oli tehty myös näyttösovelluksen ulkonäkömuutokset aikaisemmin. Yhdellä ”template” solmulla viedään ”ui_template” solmulle tiedot, että millainen kuvaaja piirretään. Toisen funktio solmun kautta tuodaan sitten päivitystaajuuden välein itse data kuvaajaan. Data käydään JavaScript -funktiolla läpi, ja lisätään oikeaan kohtaan kuvaajaa.

Kuvaajassa näkyy yhtä monta arvoa, kuin käyttäjä on asetussivuilla valinnut näytettäväksi.

Eri mittapisteitä pystyy piilottamaan kuvaajasta napauttamalla kuvaajan alla olevia legendoja. Eri mittauspisteiden arvoja pystyy näkemään myös tarkemmin napauttamalla mittapisteitä suoraan kuvaajasta. Tällöin esiin nousee tekstilaatikko, jossa näkyy aikaleima, ja tuossa aikaleimassa saatu tarkka arvo.



Kuva 36. Kuvaaja, jossa arvoja ajan funktiona

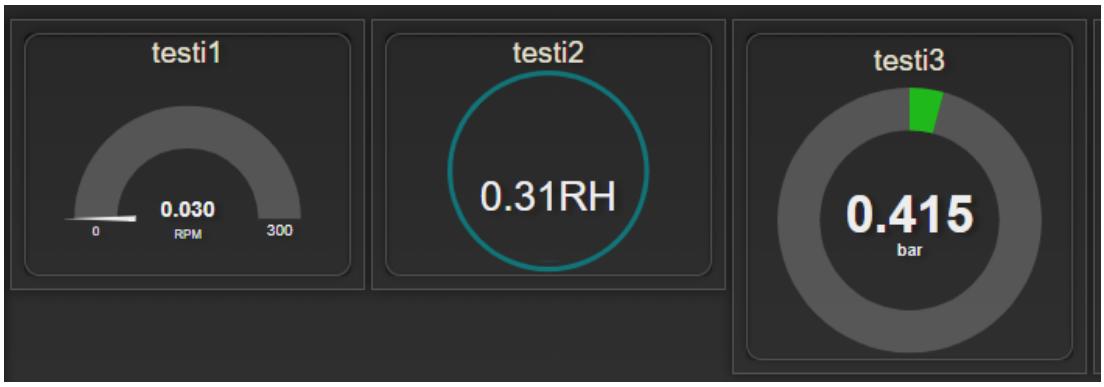
Kuvaajan koodissa on määriteltävissä, kuinka monta mittausarvoa kuvaajassa voi näkyä yhdellä kertaa. Määrä on hyvä pitää tarpeeksi alhaisena, että kuvaajan luettavuus säilyy hyvänä, eikä esittäminen käy selaimelle liian raskaaksi. Arvot

luetaan tässä kohtaa suoraan Modbus-rekisteristä eikä niitä tallenneta mihinkään välillä. Tästä johtuen, jos sivu päivitetään, kuvaajassa olevat tiedot poistuvat ja ainoastaan uudet Modbusin kautta tulevat arvot piirtyvät kuvaajaan. Tämä on kuitenkin muutettavissa kuvaajaksi, joka esimerkiksi lukee kuvaajassa esitetyt arvot aikaisemmin tallennetusta csv -tiedostosta.

7.6.2 Mittarit

Arvoja voidaan visualisoida myös Node-REDin ”dashboard” kirjastossa valmiina mukana tulevalla ”gauge” solmun avulla. Tämän solmun avulla näyttösovelluksen pystyy lisäämään erilaisia ja erinäköisiä mittareita esittämään numeerisia suureita. Mittareille on valittavissa neljä erilaista valmista ulkonäköä. ”Gauge”, ”Donut”, ”Compass” ja ” Level”.

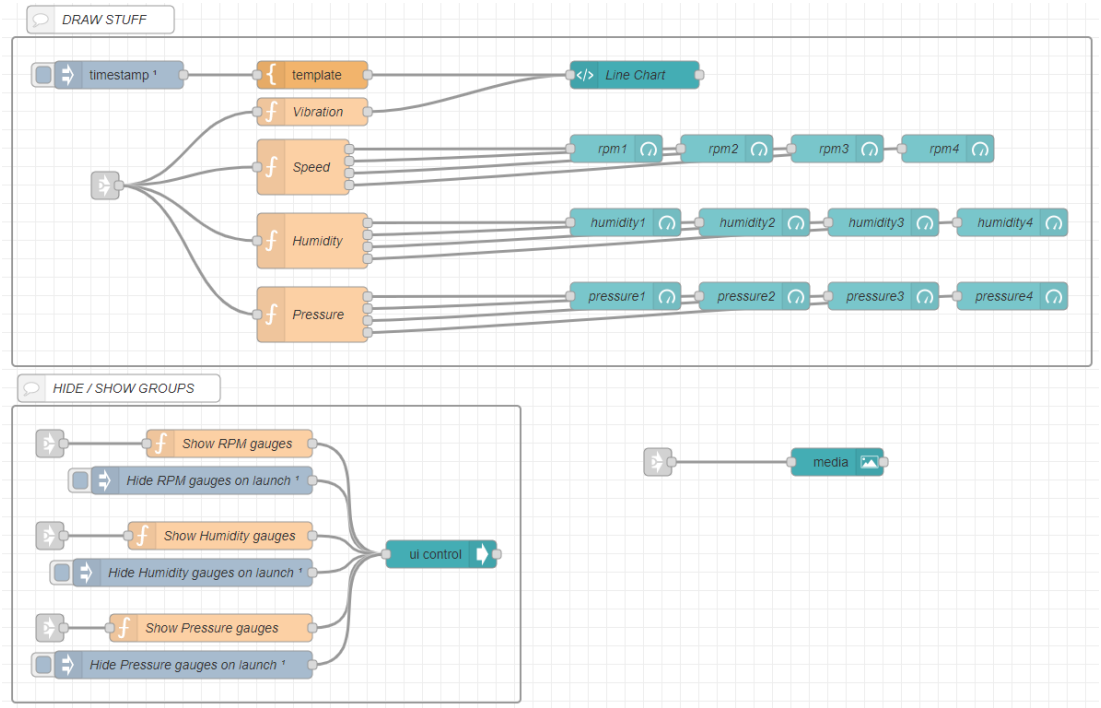
”Gauge” solmulle lähetettävässä viestissä voidaan määritellä mittarille myös otsikko, esitettävän numeroarvon lisäksi. Mittausyksikkö voidaan myös määritellä näytettäväksi lukuarvon kanssa kirjoittamalla solmun ”Units” kenttään haluttu yksikkö.



Kuva 37. Eri mittarien ulkonäöt näyttösovelluksessa

Jokainen mittari on sijoitettu omaan ryhmäänsä näyttösovelluksessa. Oletuksena Node-RED piirtää kaikki elementit, vaikka niihin ei tuotaisi mitään viestiä, eivätkä ne näyttäisi tietoa. Näyttösovelluksesa halutaan näyttää ainoastaan ne suureet mitä käyttäjä on aikaisemmin valinnut. Ongelma ratkaistiin käyttämällä jo aikaisemmin mainittua ”ui-control” solmua.

Kyseisen solmun avulla on mahdollista piilottaa eri elementtejä sivulta. (Kuva 38.) Oletuksena ohjelman alussa piilotetaan jokainen elementti. Kun käyttäjä on tehnyt valintansa, asetetaan ne ryhmät näkyviksi, joissa on esitettävää tietoa. Näin näyttösivulla näkyy ainoastaan ne ryhmät, joissa on tietoa.



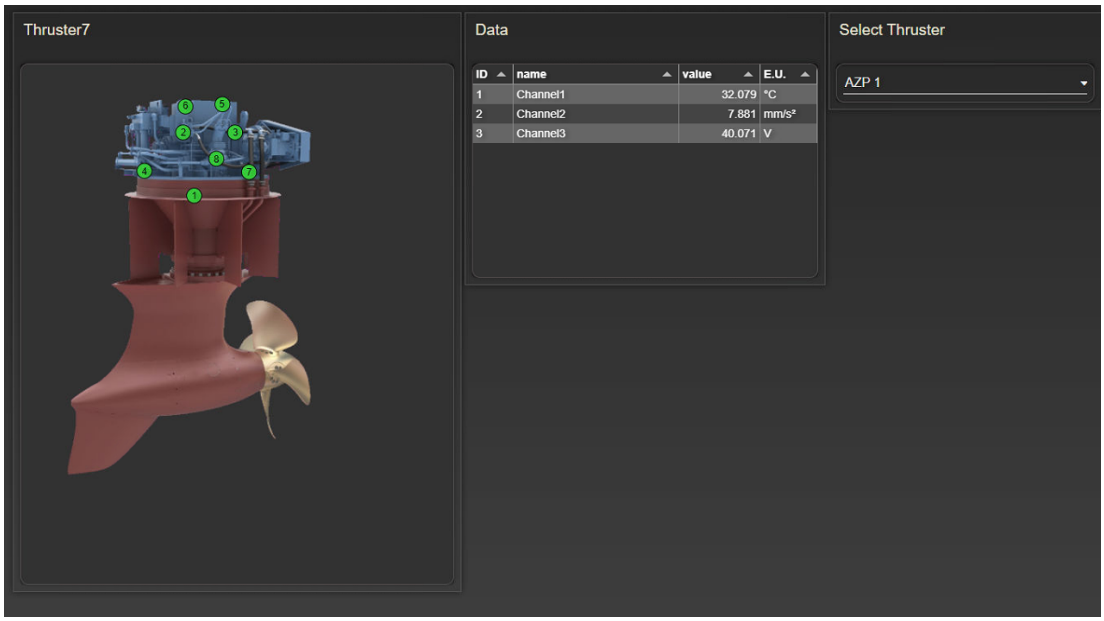
Kuva 38. Vuo käyttöliittymän komponenteista

7.6.3 Taulukko

Halutessaan käyttäjä voi esittää myös asetussivulta valitsemansa arvot uudessa taulukossa. Tällöin valitut arvot näkyvät täysin samanlaisena myös uudessa taulukossa.

7.6.4 Canvas

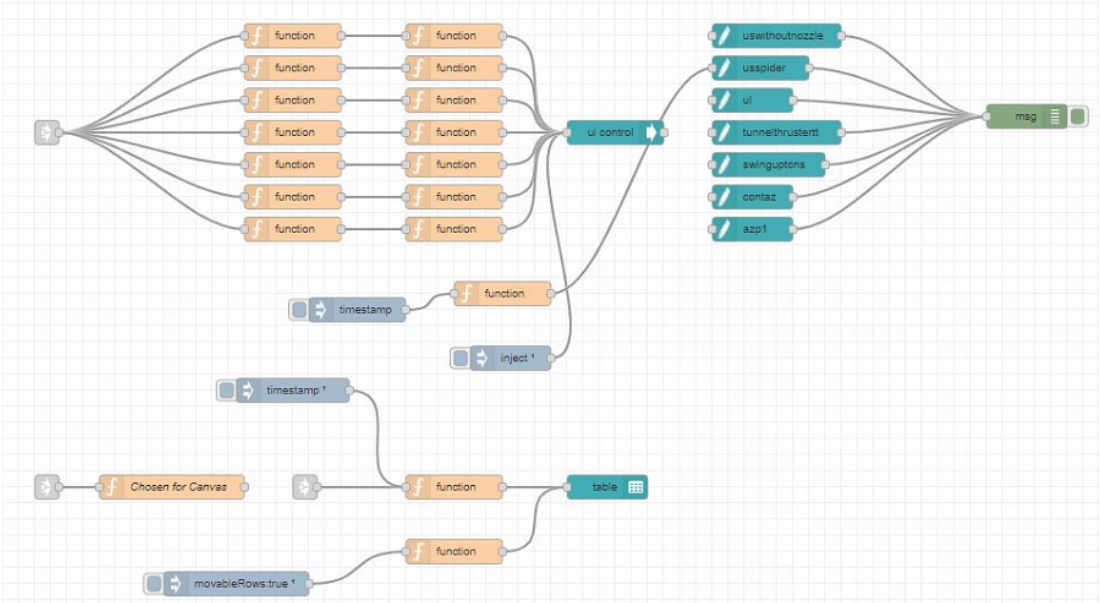
Canvas -sivulle tulee SCADA eli ”Supervisory Control And Data Acquisition” tyylinen pohja. Pohjalle tulee iso kuva potkurista, tai muusta mitattavasta laitteesta, ja sen päälle asetetaan numeroiduilla palloilla anturien paikat. Kuvan viereen viedyt valitut arvot ovat samalla tavalla numeroituja. Näin taulukon arvot ovat yhdistettävissä kuvassa näkyviin numeroihin.



Kuva 39. Canvas näyttösivu

Potkuri ja siinä olevat numeroidut pallot ovat toteutettu SVG avulla. Kirjastosta saatavana olevalla ”node-red-contrib-ui-svg” solmulla, on mahdollista tuoda näyttösovellukseen SVG -kuvia. Kuvat voi joko hakea tiedostosta, kirjoittaa itse koodina, tai tehdä solmussa itsessään mukana tulevalla editorilla. Tässä tapauksessa taustalla oleva potkurikuva on haettu tiedostosta, ja numeroidut pallot on tehty mukana tulevalla editorilla.

Kuvassa 40. näkyy, että jokaiselle potkurille on luotu oma turkoosi SVG -solmunsa. Käyttäjä valitsee haluamansa potkurin, ja se potkuri näytetään näyttösovelluksessa, muut piilotetaan. Arvot viedään vastaavanlaiseen taulukkoon, jota on jo käytetty aikaisemmilla näyttösivuilla.



Kuva 40. Canvas -sivun ohjelma

Tulevaisuudessa Canvas -sivua voi parantaa tekemällä SVG -elementeistä raahattavia, joten käyttäjä voi itse määritellä näyttösovelluksessa anturien paikan, ja tallentaa ne.

8 LOPPUTULOKSET JA YHTEENVETO

Lopputuloksena saatiin luotua halutun kaltainen sovellus. Sovellus lukee Modbus-rekisterit ja käyttäjä pystyy valitsemaan ja nimeämään rekistereitä suoraan käyttöliittymästä. Ohjelma tehtiin Weidmüllerin UC20-WL200-AC kontrollerin ja sen valmiiksi asennetulla Node-RED -paketin avulla. Ohjelma ei kuitenkaan ole sidottuna kyseiseen laitteeseen, vaan on siirrettävissä hyvin vähällä vaivalla muihin laitteisiin. Yksi tällainen laite on esimerkiksi Raspberry Pi, tai joku muu vastaava pieni, yhden piirilevyn tietokone. Ainoa asia mitä ohjelmassa tiettävästi tarvitsisi siirrettäessä muuttaa, on tiedostojärjestelmän oletuspolku.

Node-REDin avulla oli kohtuullisen helppo päästä alkuun näyttösovelluksen tekemisessä, vaikka ohjelmointikokemusta ei hirvittävästi ollut. Missään kohtaan ei tullut vastaan ongelmaa tai ominaisuutta mitä ei olisi pystynyt toteuttamaan. Ohjelman ja kirjastojen valmiit ominaisuudet ja mahdollisuudet loppuivat välillä, mutta aina löytyi keino tehdä sama asia eri tavalla.

Iso etu oli siitä, että periaatteessa kaikki mikä on mahdollista tehdä JavaScriptillä, on myös mahdollista tehdä Node-REDin avulla. Miltei puolet ohjelmasta on tästä syystä funktiosolmuja, jotka sisältävät JavaScript -koodia. Opinnäytetyön aikana opin tästä syystä paljon JavaScriptistä, vaikka siitä olikin vähän kokemusta aikaisemmin.

Jatkokehittämällä ohjelmaa siitä on mahdollista saada hyvin toimiva ja joustava ratkaisu paikallisnäytöksi eri kohteisiin.

Työ oli kokonaisuudessaan sopivasti haastava ja sisälsi sopivissa määrin tuttua, sekä uutta asiaa. Työ oli mielenkiintoinen sekä opettavainen.

LÄHTEET

Phoenix Contactin www-sivut 2021. Visu+ tuotesivu. Haettu 20.7.2021 https://www.phoenixcontact.com/online/portal/us?uri=pxc-oc-itemdetail:pid=2988544&library=usen&tab=1

Grafanan www-sivut 2021. Haettu 20.7.2021 https://grafana.com/grafana/

Weidmüllerin www-sivut 2021. UC20-WL2000-AC tuotesivu. Haettu 20.7.2021. https://catalog.weidmueller.com/catalog/Start.do?localeId=en&ObjectID=1334950000

Weidmüllerin www-sivut 2021. UV66-BAS-10-RES-W tuotesivu. Haettu 21.7.2021. https://catalog.weidmueller.com/catalog/Start.do?localeId=en&ObjectID=2555820000

Raspberry Pin www-sivut 2021. Raspberry Pi 4 Model B tuotesivu. Haettu 20.7.2021. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

Node-REDin www-sivut 2021. Haettu 22.7.2021. https://nodered.org/about/

Peyrott, Sebastian, 2017. A Brief History of JavaScript. https://auth0.com/blog/a-brief-history-of-javascript/

Washingtonin yliopiston www-sivut 2021. A Brief History of HTML. https://www.washington.edu/accesscomputing/webd2/student/unit1/module3/html_history.html

SKF www-sivut 2021. Haettu 22.7.2021. https://www.skf.com/group/products/condition-monitoring-systems/surveillance-systems/on-line-monitoring/imx

Modbus Organization, 2006. Haettu 23.7.2021. https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf

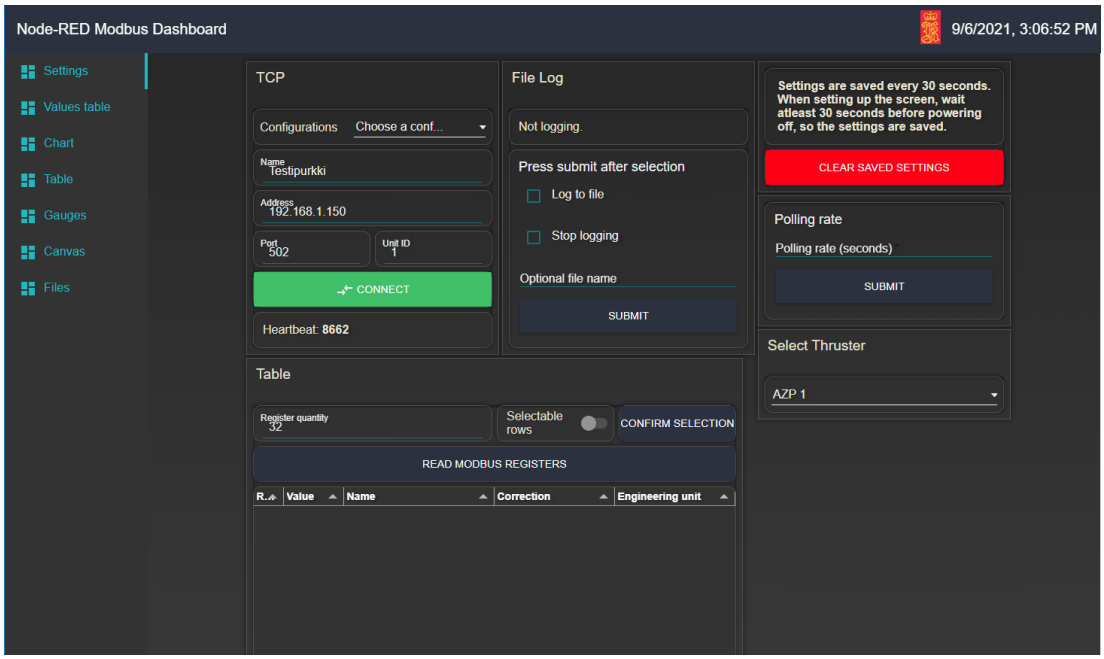
Node-RED www-sivut 2021. Haettu 6.9.2021.
https://flows.nodered.org/node/node-red-contrib-uk-national-rail

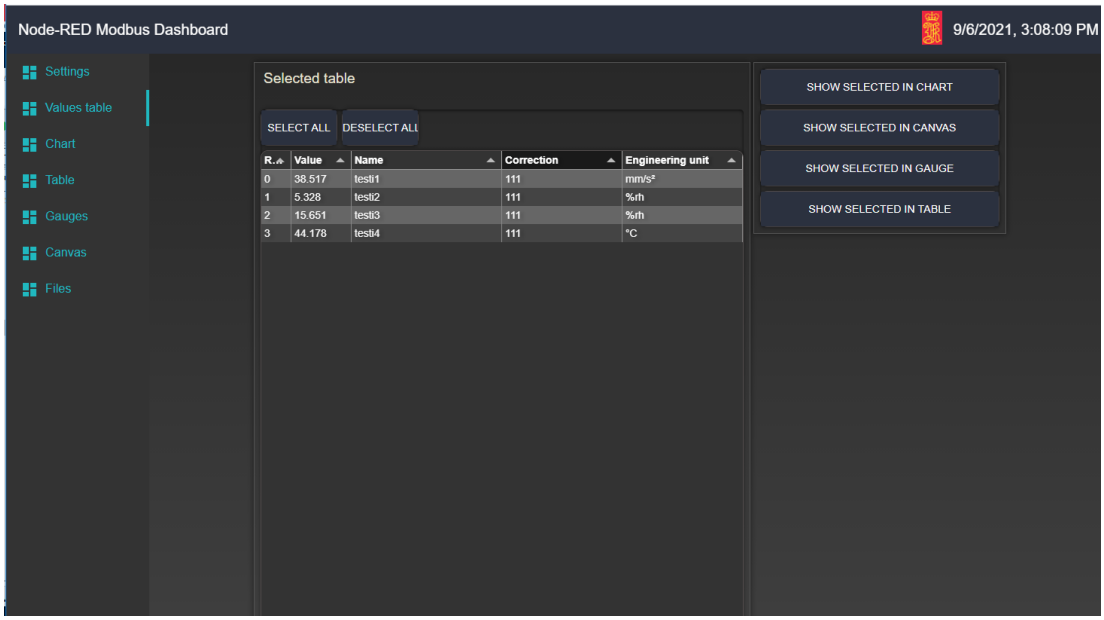
Modbus Organization, 2012. Haettu 23.7.2021. https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

File browser in dashboard, (2020). Haettu 30.7.2021. https://flows.nodered.org/flow/44bc7ad491aacb4253dd8a5f757b5407

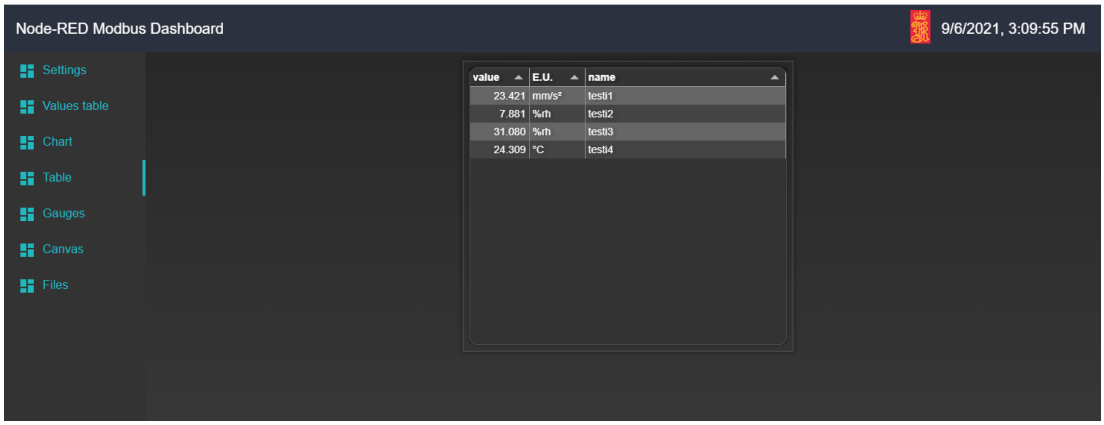
Shafranovich, Y, 2005. Common Format and MIME Type for Comma-Separated Values (CSV) Files. Haettu 30.7.2021 https://www.ietf.org/rfc/rfc4180.txt

LIITTEET

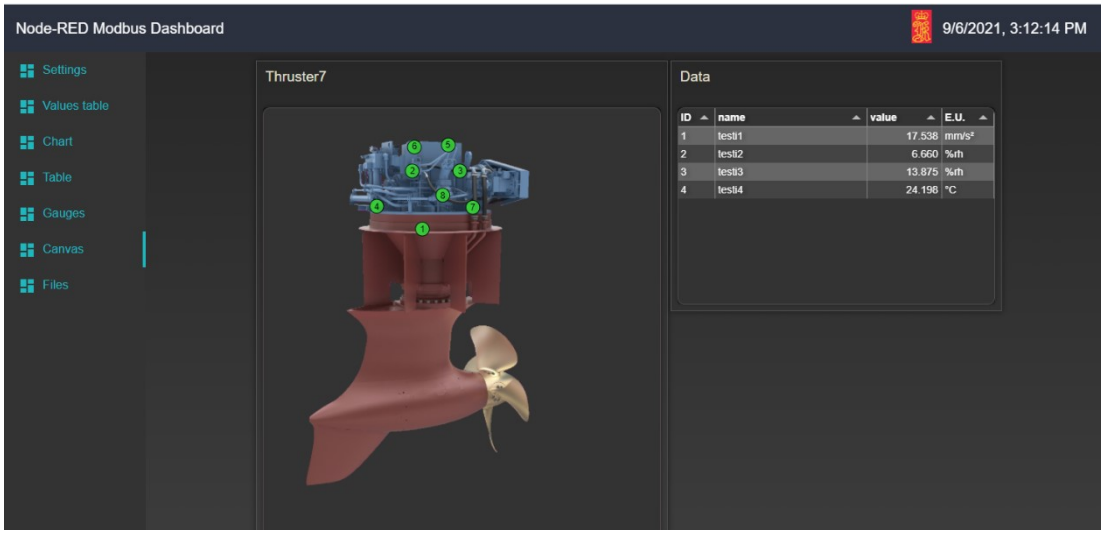

Node-RED Modbus Dashboard
9/6/2021, 3:06:52 PM
Settings
Values table
Chart
Table
Gauges
Canvas
Files
TCP
Configurations
Choose a conf...
Name
Testipurkki
Address
192.168.1.150
Port
502
Unit ID
1
CONNECT
Heartbeat: 8662
File Log
Not logging.
Press submit after selection
Log to file
Stop logging
Optional file name
SUBMIT
Settings are saved every 30 seconds. When setting up the screen, wait atleast 30 seconds before powering off, so the settings are saved.
CLEAR SAVED SETTINGS
Polling rate
Polling rate (seconds)
SUBMIT
Select Thruster
AZP 1
Table
Register quantity
32
Selectable rows
CONFIRM SELECTION
READ MODBUS REGISTERS
R..
Value
Name
Correction
Engineering unit

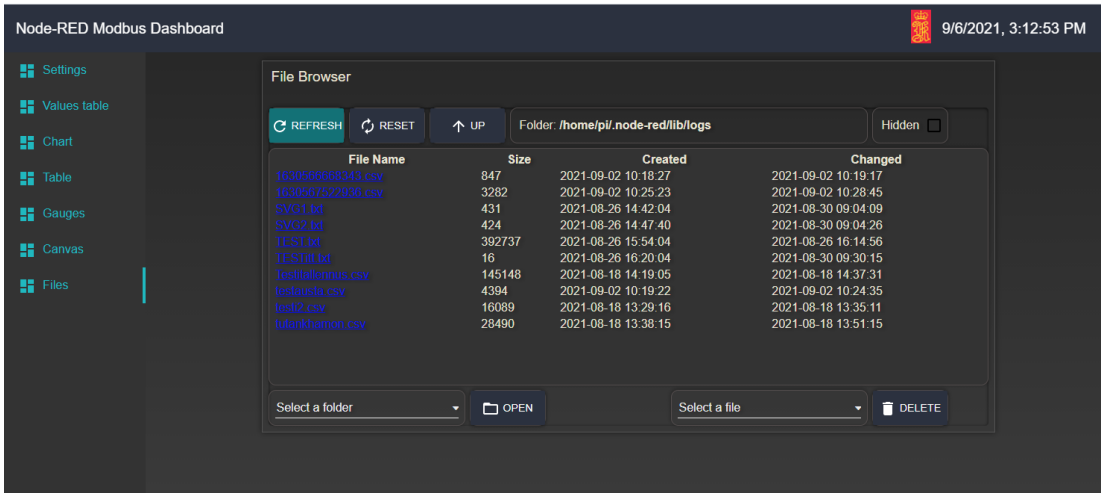

Node-RED Modbus Dashboard
9/6/2021, 3:08:09 PM
Settings
Values table
Chart
Table
Gauges
Canvas
Files
Selected table
SELECT ALL
DESELECT ALL
R..
Value
Name
Correction
Engineering unit
0 38.517 testi1 111 mm/s²
1 5.328 testi2 111 %rh
2 15.651 testi3 111 %rh
3 44.178 testi4 111 °C
SHOW SELECTED IN CHART
SHOW SELECTED IN CANVAS
SHOW SELECTED IN GAUGE
SHOW SELECTED IN TABLE


Node-RED Modbus Dashboard
9/6/2021, 3:09:02 PM
Settings
Values table
Chart
Table
Gauges
Canvas
Files
testi1
testi2
testi3
testi4
35
30
25
20
15
10
5
0
3:08:43
3:08:45
3:08:47
3:08:49
3:08:51
3:08:53
3:08:55
3:08:57
3:08:59
3:09:01
3:09:03
3:09:04


Node-RED Modbus Dashboard
9/6/2021, 3:09:55 PM
Settings
Values table
Chart
Table
Gauges
Canvas
Files
value
E.U.
name
23.421
mm/s²
testi1
7.881
%rh
testi2
31.080
%rh
testi3
24.309
°C
testi4


Node-RED Modbus Dashboard
9/6/2021, 3:11:31 PM
Settings
Values table
Chart
Table
Gauges
Canvas
Files
testi1
57.720
0
300
testi2
4.329
0
300
testi3
25.419
0
300
testi4
20.757
0
300


Node-RED Modbus Dashboard
9/6/2021, 3:12:14 PM
Settings
Values table
Chart
Table
Gauges
Canvas
Files
Thruster7
Data
ID
name
value
E.U.
1
testi1
17.538
mm/s²
2
testi2
6.660
%rh
3
testi3
13.875
%rh
4
testi4
24.198
°C


Node-RED Modbus Dashboard
9/6/2021, 3:12:53 PM
Settings
Values table
Chart
Table
Gauges
Canvas
Files
File Browser
REFRESH
RESET
UP
Folder: /home/pi/.node-red/lib/logs
Hidden
File Name
Size
Created
Changed
1630569668344.csv
847
2021-09-02 10:18:27
2021-09-02 10:19:17
1630567422938.csv
3282
2021-09-02 10:25:23
2021-09-02 10:28:45
SVG1.txt
431
2021-08-26 14:42:04
2021-08-30 09:04:09
SVG2.txt
424
2021-08-26 14:47:40
2021-08-30 09:04:26
TEST.txt
392737
2021-08-26 15:54:04
2021-08-26 16:14:56
TESTfitt.txt
16
2021-08-26 16:20:04
2021-08-30 09:30:15
testilallemus.csv
145148
2021-08-18 14:19:05
2021-08-18 14:37:31
testausta.csv
4394
2021-09-02 10:19:22
2021-09-02 10:24:35
test2.csv
16089
2021-08-18 13:29:16
2021-08-18 13:35:11
tulankhamon.csv
28490
2021-08-18 13:38:15
2021-08-18 13:51:15
Select a folder
OPEN
Select a file
DELETE