Toni Aaltonen

AVOIMEEN LÄHDEKOODIIN TUKEUTUVAN AUTONOMISEN SUKELTAVAN ROBOTIN SUUNNITTELU JA TOTEUTUS

Sähkö- ja automaatiotekniikan koulutusohjelma

2019

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

AVOIMEEN LÄHDEKOODIIN TUKEUTUVAN AUTONOMISEN SUKELTAVAN ROBOTIN SUUNNITTELU JA TOTEUTUS

Aaltonen, Toni
Satakunnan ammattikorkeakoulu
Sähkö- ja automaatiotekniikan koulutusohjelmaa
Marraskuu 2019
Sivumäärä: 44
Liitteitä:

Asiasanat: vedenalainen tutkimus, autonominen, sukeltava robotti, ROS

Tämän opinnäytetyön tarkoituksena oli suunnitella ja valmistaa mahdollisimman monikäyttöinen vedenalaisen tutkimusalustan prototyyppi, jota on mahdollista käyttää etäohjauksella ja joka suorittaa yksinkertaisia autonomisia tehtäviä.

Projektin ympärille koottiin työryhmä opiskelijoista. Kokonaisuudessaan sen toteuttaminen oli niin monimutkainen ja laaja kokonaisuus, ettei sitä olisi ollut mahdollista toteuttaa yhden ihmisen työnä.

Tässä Opinnäytetyössä olen käsitellyt lähinnä osuuksia, joita olen itse tehnyt. Niitä olivat pääsuunnittelu, suurin osa ohjelmoinnista, joidenkin antureiden suunnittelu ja valmistus, datankeräyslaitteiston valmistus, testaus sekä datankeräys. Projektin johtaminen ja hankinnat olivat myös minun tehtäviäni, mutta olen rajannut niistä raportoinnin tämän opinnäytetyön ulkopuolelle.

Projekti kokonaisuudessaan ei valmistunut oppinäytetyön kirjoittamisen aikana, monet osakokonaisuudet ovat kuitenkin valmiita, ja testattuja. Tämän lisäksi kerättiin merkittävä data määrä, erilaisia koneoppis algoritmeja varten.

DESIGNING AND EXECUTING OF AN AUTONOMOUS UNDERWATER VEHICLE BASED ON OPEN SOURCE CODE

Aaltonen, Toni
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Electrical and Automation Engineering
November 2019
Number of pages: 44
Appendices:

Keywords: underwater research, autonomous, diving robot, ROS

The purpose of this thesis was to design and construct an underwater research platform, enabling both remote operation missions and simple autonomous missions.

Project as hole was so multitudinous and wide that it was not practical for only one person to carry out. I gathered a team of students for the project.

In this thesis I cover mostly the parts I made myself, the main design, most parts of software engineering, design and manufacturing of some sensors, manufacturing of data gathering device, testing and data gathering. Project management and acquisitions were my responsibilities, but I excluded them from the thesis.

The project as a whole was not completed at the time of writing the thesis, however many sub-modules are ready and tested. In addition, a significant amount of data was collected for various machine learning algorithms.

ESIPUHE

Haluan kiittää Satakunnan Ammattikorkeakoulua. Erityisesti projektipäällikkö Timo Kermistä, yliopettaja Mirka Leinoa, tutkimusjohtaja Petteri Pulkkista ja laboratoriomestari Harri Aholaa ainutkertaisesta mahdollisuudesta tehdä tutkimusta erittäin mielenkiintoisen hankkeen parissa, Satakunnan ammattikorkeakoulun automaatiotekniikan opiskelijoita Marko Taskista ja Antti Virtasta, joiden asiantuntemus ja työskentely projektissa on ollut kriittisen tärkeää hankkeen etenemiselle, Ulla Tuomisen Säätiötä sekä Satakunnan Korkean Teknologian Säätiötä rahoituksen mahdollistamisesta, Winnovan ammatillisen oppilaitoksen lehtori Antti Salmista, lehtori Isto Jokista ja lehtori Tuomas Kyyhkystä, jotka auttoivat osien valmistuksessa, Helkama Bica Oy:tä kaapelien lahjoittamisesta, Diplomi-insinööri Antti Hallaa ohjelmointiavusta, Satakunnan pelastuslaitoksen Rauman toimipistettä datankeräyksen avustamisesta, sekä kaikkia muita erikseen mainitsemattomia tahoja ja henkilöitä, jotka ovat panoksellaan mahdollistaneet hankkeen toteutumisen ja etenemisen.

SISÄLLYS

1 JOHDANTO JA PROJEKTIN LYHYT HISTORIA

Vuoden 2018 lopussa Satakunnan ammattikorkeakoulussa esiteltiin Marja-Riitta Saarivirran opinnäytetyö (Autonomiset vedenalaiset laitteet pähkinänkuoressa). Tilaisuudessa esittelin omia näkemyksiäni siitä, miten asiaa tulisi lähestyä, jos kyseiseen tutkimussuuntaan halutaan lähteä. Pian tilaisuuden jälkeen minulle tarjottiin mahdollisuutta ryhtyä vetämään hanketta ja valmistaa prototyyppi. Tein hieman lisää esiselvitystä ja kirjoitin tutkimussuunnitelma, jossa esitin, että 10.000 €:n budjetilla laitteen tekeminen olisi todennäköisesti mahdollista. Suunnitelma hyväksyttiin ja vuoden 2019 alusta aloitettiin laitteen suunnittelu ja toteutus.

Projektin laajuudesta johtuen kokosin työryhmän opiskelijoita tekemään yhdessä projektin eri osia. Marko Taskinen teki suurimman osan mekaniikkasuunnittelusta, 3D-kuvien piirtämisestä ja mekaanisesta sekä sähköisestä kokoonpanosta. Antti Virtanen teki moottoriohjaimiin liittyvät piirikaavioiden suunnittelut, kokoonpanon sekä suurelta osin näihin liittyvät ohjelmoinnit. Antti Halla auttoi myös ohjelmoinnissa sekä toi ohjelmistoalan osaamista ryhmään.

Tammikuusta kesäkuun alkuun tein osa-aikaisesti työtä noin 10 tuntia viikossa. Kesäkuusta eteenpäin olen tehnyt täyspäiväisesti työtä hankkeen parissa. Projektin alussa selvitettiin olemassa olevia ratkaisuja, jonka jälkeen aloitettiin tarkempi suunnittelu ja spesifioitiin tarvittavat komponentit. Samalla selvitettiin tarpeet yhteistyöverkostolle ja luotiin kontaktit yhteistyötä varten. Ohjelmistosuunnittelu, ohjelmistotuotanto ja testaus on jatkunut koko projektin ajan, ollen selvästi työläin osuus projektista. Suunnittelun jälkeen alkoi laitteen rakennus, samanaikaisesti alkoi myös datan keräys veneeseen kiinnitettyjen anturien avulla.

2 VEDENALAISTEN LAITTEIDEN HISTORIA JA NYKYTILA

Ensimmäiset etäohjauksella toimivat vedenalaiset laitteet kehitettiin 1950-luvulla. Autonomisia vedenalaisia laitteita on tehty 1980-luvulta lähtien. Ensimmäiset laitteet olivat lähinnä valtiollisten toimijoiden kehittämiä. (Saarivirta 2018, 11)

Vedenalaisilla laitteilla on hyvin monenlaisia käyttökohteita. Esimerkkeinä voidaan sanoa maanpuolustuksessa miinanetsintä ja uponneiden laitteiden palautus, tutkimuksessa suolaisuuden tarkkailu, tsunamihavainnointi sekä tulivuorten tarkkailu ja tutkiminen. Öljy- ja kaasuteollisuudessa tehdään muun muassa kunnonvalvontaa ja tarkastuksia. Sukeltavien robottien rakenteet ja kokoluokat vaihtelevat suuresti käyttökohteiden mukaan, pienimpien ollessa muutaman kilogramman laitteita, suurimpien massat tuhansia kilogrammoja. (Saarivirta 2018, 27)

Nykyään laitteita kehittävät niin kaupalliset tahot kuin tutkimuslaitokset. Autonomisille laitteille järjestetään myös kilpailuja kuten vuosittainen robosub, joka järjestetään Kaliforniassa, Yhdysvalloissa. Vuosittain kilpailun tehtävien vaativuustaso on noussut, mikä on kannustanut laitteiden jatkuvaan kehittämiseen. Kilpailuun osallistuu opiskelijaryhmiä eri puolilta maailmaa. Monet ryhmät ovat jakaneet avoimesti tietoa laitteistaan, ja niiden ohjausjärjestelmistä. Kilpailuun osallistuneiden laitteiden joukosta löytyi myös hyvä esikuva tälle projektille. (Robonation 2019).

3 PROJEKTIN ESIKUVA

Esikuvaksi projektille otettiin Floridan yliopiston Subjugator-projekti. Floridan yliopistossa on yli 20 vuoden kokemus autonomisista vedenalaisista laitteista. Yliopisto on menestynyt Robosub-kilpailuissa, joissa se on sijoittunut 7 kertaa kolmen parhaan joukkoon sekä voittanut kilpailun kolme kertaa kilpailun (University of Florida 2017). Floridan yliopiston laite perustuu ROS-järjestelmään (Robot operating system) ja lähdekoodi on avointa (Machine Intelligence Laboratory University of Florida 2019). Tämä oli hyvä lähtökohta tälle projektille. Toki suuria eroavaisuuksiakin laitteille tuli. Osittain eroavaisuudet selittyvät budjetin erovaisuuksilla. Pelkästään floridalaisten käyttämä kaikuluotain maksaa noin 60.000 euroa ja ohjelmistopaketti siihen 7000 euroa. Samoin Subjugaattorin nopeuden mittaukseen käyttämä dopler-nopeusloki maksaa noin 20.000 euroa.

Toinen suuri eroavaisuus on, että Robosub-kilpailu järjestetään altaassa. Kirkkaassa vedessä kamerapohjaiset konenäköjärjestelmät ovat kustannustehokas ja helppo tapa objektien tunnistukseen. Tämän projektin laitteen on tarkoitus toimia Suomen sameissa vesissä, joissa kameroiden käytön mahdollisuudet ovat hyvin rajalliset. Näkyvyys rantavesissä lasketaan usein joissain kymmenissä senteissä, ja se saattaa rajoittua jopa kahteenkymmeneen senttimetriin.

Myös tämän projektin potkurijärjestelmästä tuli melko erilainen. Tosin viime aikoina floridalaiset ovat vaihtaneet oman järjestelmänsä lähemmäksi tässä projektissa valittua järjestelmää. (Suhlman 2019). Kuvassa 1 näkyvät vielä vanhat Videorayn M5-potkurijärjestelmät.

Kuva 1. Floridan yliopiston Subjugator (University of Florida 2016)

4 FYYSINEN ARKKITEHTUURI

Projektin mekaanista suunnittelua tehnyt Marko Taskinen piirsi järjestelmän 3D-mallin Solidworks-ohjelmalla. Laite rakennettiin tämän mallin pohjalta. Kuvassa (kuva 2) näkyy laitteen runko ilman potkurilaitteita, antureita ja johdotuksia.

Kuva 2. Laitteen 3D-malli

4.1 Runkorakenteet ja potkurijärjestelmät

Laitteen rungon kehikko rakennettiin alumiiniprofiilista. Kehikon sisään sijoitettiin kaksi alumiinista valmistettua painerunkoa elektroniikkaa varten. Näiden hitsaus suoritettiin Winnovalla Antti Salmisen toimesta. Ylempään suurempaan painerunkoon sijoitettiin kaikuluotainyksikkö ja päätietokone. Alempaan painerunkoon taas sijoitettiin virtalähteet, akusto ja moottoriohjaimet. Laitteesta suunniteltiin lievästi kelluva, jotta se palautuu vikatilanteessa automaattisesti pintaan.

Valinta etäkäytön ja akkukäytön välillä tapahtuu alemman painerungon laippaa vaihtamalla. Toinen niistä on umpilaippa ja toisesta lähtee maasähkökaapeli sekä valokaapeli, joka toimii tietoliikennelinkkinä ja toimii vetolujuutensa ansiosta sähkökaapelin

vetosuojana. Lisäksi samassa nipussa kulkee paineistettu ilmaletku, jonka tarkoituksena on tehdä kaapelista lievästi kelluva, jotta se ei takerru pohjan muotoihin. Samalla kelluvuus vähentää myös laitteeseen kohdistuvaa vastusta.

Potkurilaitteita on esikuvan mukaisesti kahdeksan kappaletta ja ne on sijoitettu kuvan (kuva 3) mukaisesti. Rakenteeseen päädyttiin siksi, että vaikka mikä tahansa potkureista vioittuisi, voidaan tehtävä silti suorittaa loppuun ja liikkuminen onnistuu edelleen mihin tahansa suuntaan. Joissain tapauksissa kahdenkaan potkurin vikaantuminen ei vielä rajoita laitteen liikkuvuutta. Potkurijärjestelmät perustuvat Ole Hermanin suunnitelmiin, joita on tässä jonkun verran muokattu. Potkurijärjestelmät tehtiin 3D-tulostettuna PLA-muovista.

Kuva 3. Potkurilaite kiinnitettynä runkoon

4.2 Päätietokone

Päätietokoneena toimii AMD-pohjainen x86-tietokone. Valinnan AMD:n ja Intelin välillä ratkaisi halvempi hinta ja parempi päivitettävyys. Jo valintaa tehtäessä oli varmaa, että am4-kannalle olisi myöhemmin saatavissa vähintään 12-ytimisiä ja pienemmällä viivaleveydellä valmistettuja prosessoreja. Tarvittaessa on siis mahdollista lisätä laskentatehoa tai pienentää virrankulutusta pelkällä prosessorin vaihdolla. Tässä projektissa käytetyillä kellotaajuuksilla AMD 2700 ja Intel 8700 ovat kuitenkin käytännössä tasoissa tehojen suhteen. (Pasmark 2019).

Prosessorina on Ryzen 2700 kahdeksalla ytimellä ja kuudellatoista säikeellä. Emolevynä on Gigabyten B450 kapea mATX-emolevy kahdella muistipaikalla. Keskusmuistina on 32 Gigatavua ddr4-muistia 3000 Mhz:n taajuudella. Näytönohjaimena toimii Asus Dual 1070 Gtx 8 Gigatavun muistilla. Kovalevynä on 512 Gigatavun Samsung Evo 970 pcie 4x ssd. Tietokoneen virtalähteenä on Hd plex 400w dc-dc. Tätä voidaan käyttää niin akuista saatavalla virralla kuin myös päävirtalähteen 24v dc jännitteellä. Päävirtalähteenä, joka muuttaa 240 Vac maasähkön 24 Vdc muotoon, toimii Meanwell DPU-3200. Sen hyötysuhteeksi luvataan yli 94% täydelläkin 3192w teholla. (Meanwell 2019, 3).

4.3 Kaikuluotaimet

Kuten edellä esitettiin, robotiikkakäyttöön tarkoitetut kaikuluotainyksiköt maksavat kertaluokkia enemmän kuin tämän projektin budjetilla oli mahdollista hankkia. Tästä johtuen oli etsittävä vähemmän tavanomaisia vaihtoehtoja. Haasteellista tässä oli se, ettei löytynyt mitään viitteitä siitä, että kukaan olisi ennen kokeillut jotain vastaavaa. Veneilyssä käytetyt kaikuluotaimet ovat melko edullisia, mutta kuvanlaadultaan silti melko hyviä. Lisäksi ne ovat helppokäyttöisiä, jos verrataan esimerkiksi floridalaisten käyttämään kaikuluotaimeen, jota he eivät ole vieläkään saaneet toimimaan täysin halutulla tavalla (Volya 2018), vaikka heillä on huomattavat resurssit käytettävissään. Yksityisille ihmisille myytävät laitteet ovat helppokäyttöisiä, ”asenna ja käytä” -tyyppisiä laitteita. Ongelmana näissä on se, ettei niitä ole tarkoitettu tämän kaltaiseen käyttöön ja siitä johtuen kuvan saaminen tietokoneelle on haasteellista. Tässä projektissa

käytiin läpi eri valmistajia, kuten Humminbird, Lowrance, Raymarin ja Garmin, etsien vaihtoehtoja. Tämän jälkeen todettiin, että siinä missä vanhemmat sarjamuotoiset kenttäväylät, joita venepuolella on käytetty, olivat suhteellisen avoimia ja kaikkien käyttämiä, kuten NMEA0183 (National Marine Electronics Association 2002) ja uudempi NMEA2000 (National Marine Electronics Association 2009), ovat Ethernet-pohjaiset väylät valmistajien omia suljettuja ja ne keskustelevat vain valmistajan omien laitteiden välillä. Edes siitä ei ollut tietoa, onko liikenne jollain tapaa salattua vai ei.

Modernien kaikuluotaimien tuottama data on niin tarkkaa ja kaistaa vievää, että kaikkien valmistajien kaikuluotaindata kulkee laitteiden välillä Ethernet-pohjaisissa väylissä. Datan saamiseksi laitteelta voidaan käyttää useampia tapoja. Voidaan selvittää käytetty Ethernet-protokolla liikennettä kuuntelemalla ja kirjoittaa tätä käyttävä ohjelma, joka esiintyy valmistajan laitteena verkossa niin, että luotain saadaan lähettämään sille datansa. Ongelmallista tässä on se, että tarvitaan vähintään kaksi kaikuluotainyksikköä, jotka keskustelevat keskenään Ethernet-pohjaisen väylän yli, jotta Ethernet-liikennettä voidaan kuunnella. Toinen vaihtoehto on yksi kaikuluotainyksikkö ja jokin muu laite, joka väylässä toimii esimerkiksi Ethernet-pohjainen kaikuanturi. Toinen ongelma on, ettei ole varmuutta siitä onko liikenne salattua vai ei. Nämä kaikki asiat huomioon ottaen ryhdyttiin miettimään vaihtoehtoista tapaa saada data tietokoneelle.

Osa kaikuluotaimista tukee kuvan lähettämistä wifi-yhteyden yli Android-laitteelle. Tällöin ongelmaksi muodostuu langattomien yhteyksien luotettavuus ja viive. Tämän lisäksi tabletilta olisi pitänyt jotenkin siirtää kuva tietokoneelle, tai käyttää jotain virtuaaliandroid-ympäristöä tietokoneelta. Tähän ratkaisuun ei päädytty, koska sitä ei pidetty luotettavana.

Joistain Simradin ja Garminin paremmista mallisarjoista löytyy HDMI-ulostulo ja sen sopivuutta alettiinkin selvittää. Alle sadan euron hintaluokassa on saatavilla HDMI-USB3-kaappauskortteja, jotka on tarkoitettu videon käsittelyyn, tallennukseen sekä uudelleenlähetykseen. Tästä syystä voitiin olettaa, että niissä viive olisi kohtuullinen. Toteutuksen vaikeusaste tässä ratkaisussa on huomattavasti pienempi kuin väyläprotokollan selvittämisessä. Vaikka tarkkuudessa hieman jäätäisiinkin, on se säästetyn

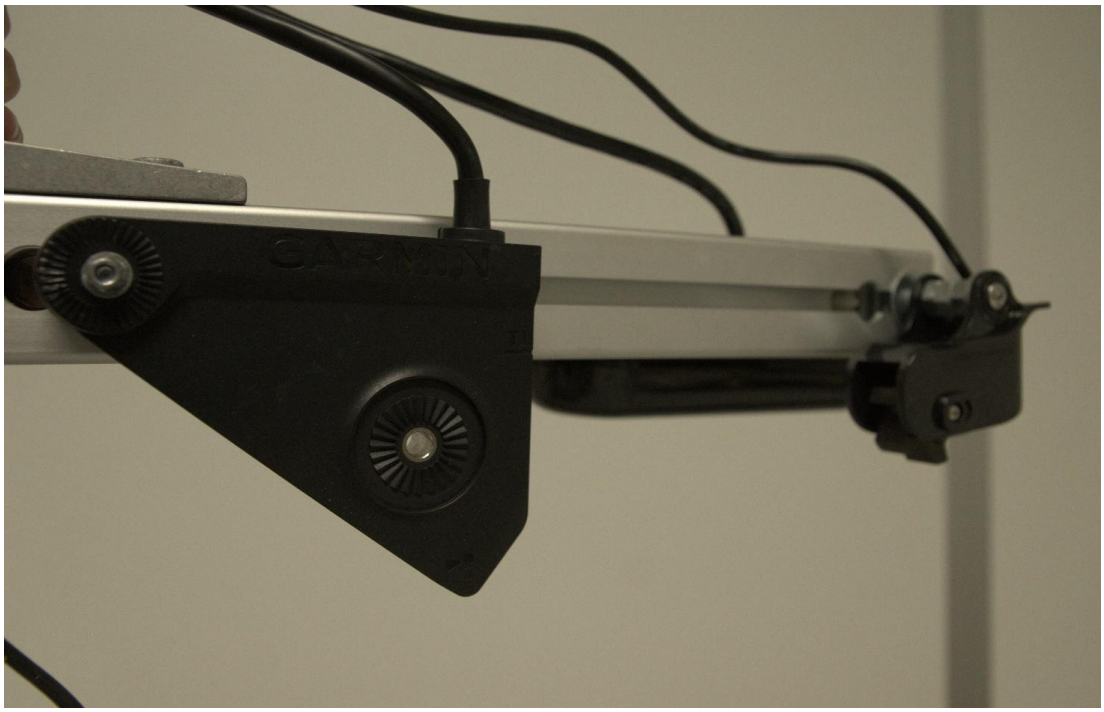
vaivan sekä pienentyneiden riskien arvoista. Näistä syistä päädyttiin tähän ratkaisuun. Toimivuudesta ei ollut päätöstä tehdessä varmuutta, koska mistään ei löytynyt tietoa siitä, että kukaan olisi aiemmin yrittänyt vastaavaa. Ratkaisu on kuitenkin toiminut odotetulla tavalla, eikä ainakaan silmin havaittavaa viivettä ole kaikuluotainyksikön kuvan ja tietokoneelle siirretyn kuvan välillä esiintynyt. Myös vaikutus kuvanlaatuun on osoittautunut merkityksettömän pieneksi.

Harkinnassa oli myös neljäs tapa, jolla olisi voitu käyttää vielä jonkin verran halvempia kaikuyksiköitä. Koska niihin ei olisi suoraan saatu kiinnitettyä haluttuja antureita, olisi lisää hintaa kertynyt niin sanotuista black box -laitteista, jotka toimivat erillisinä kaikuluotainyksiköinä antureille. Tässä tavassa olisi purettu näyttö kokonaan pois kaikuluotainyksiköstä, kaapattu näytön signaali suoraan näyttöliittimeltä ja muunnettu se tietokoneelle sopivaksi. Haasteeksi tässä olisi muodostunut se, että datankeräysvaiheessa kaikuluotainyksikköä olisi ollut vaikeampi käyttää. Näistä halvemmista laitteista puuttui myös yleensä GPS-yksikkö, joka on tärkeä opetusvaiheessa. Teknisessä mielessä ei myöskään ollut tiedossa, minkä tyyppistä signaalia näyttö käyttää. Tämä asia olisi kuitenkin ollut varmasti selvitettävissä laitetta purettaessa. Kaikki asiat huomioon ottaen tätä tapaa ei kuitenkaan valittu. Joskin tätä lähestymistä voidaan edelleen pitää varteenotettavana vaihtoehtona.

Kaikuluotainyksiköksi valikoitui lopulta Garminin 8400 Xsv. Sen valintaan Simrad Nss12 evo3- mallien sijaan vaikuttivat lopulta seuraavat seikat:

1. Hinta, joka Garminissa oli hieman halvempi.
2. Näytön tarkkuus, joka tarkoittaa, että myös HMDI-lähdön tarkkuus on suurempi, kun Garminissa se on 1080p eli FullHd ja Simrad:illa se on pienempi eli 1280 x 800.
3. Garminin mallistosta löytyvä edullinen reaaliaikakuvaa tuottava kaikuanturi Lvs12, joka toimii suoraan valitulla kaikuluotainyksiköllä, vieläpä ilman erillisiä black box -laitteita.
4. Garminin tekninen tuki, joka oli aina tavoitettavissa nopeasti puhelimella ja sähköpostilla, ja jossa he selvittivät vaativammatkin heille esitetyt kysymykset.

Lvs12-anturin lisäksi hankittiin vielä GT54UHD-TM -viistokaikuanturi. Datankeräystä varten kaikuluotausanturit kiinnitettiin yhteen alumiiniprofiiliin (kuva 4).



Kuva 4. Vasemmalla lvs12 reaaliaika-anturi, oikealla taka-alalla Gt54UHD-TM ja oikealla etualalla siipiratasnopeusanturi

5 OHJELMISTOARKKITEHTUURI

5.1 Ubuntu

Koska laitteeseen tulee tehokas x86-tietokone neuroverkkolaskentoja varten sekä muita laskennallisesti vaativia ohjauksia, tarvitaan niitä varten käyttöjärjestelmä. Linux valittiin käyttöjärjestelmäksi johtuen sen parhaasta yhteensopivuudesta ROS-järjestelmän kanssa. Linux-jakeluista Ubuntu valikoitui samasta syystä. ROS toimii myös jossain määrin Windows 10 -käyttöjärjestelmässä, mutta siinäkin tapauksessa se nojautuu Windowsin Linux-subsysteemin varaan. Myös macOS on jossain määrin tuettu, mutta raudan kalliimpi hinta, vaihtoehtojen rajallisuus sekä kuitenkin rajallinen tuki rajasivat tämän vaihtoehdon pois.

Käyttöjärjestelmä itsessään ei vaatinut muokkausta, ohjelmien ja kirjastojen asennusta lukuun ottamatta. Myös joitain tiedostoja oli tarpeen kirjoittaa, yhtenä esimerkkinä näistä udev.rules -tiedostot. Näitä käytetään tässä tapauksessa luomaan symbolinen osoite tietyille laitteille. Kun esimerkiksi koneen usb-paikkaan laitetaan kaksi virtuaali- com -porttilaitetta voivat nämä saada ”usbtty0”- ja ”usbtty1”-nimet sattumanvaraisesti. Tämä taas aiheuttaa ongelman, kun ROS-launch -tiedostossa tietty laite käynnistetään usbtty-osoitteella, voi osoitteessa olla odotettu laite tai joku toinen. Datankeräysvaiheessa sama ongelma oli myös esimerkiksi videokaappauslaitteella, joka Ubuntulle näyttäytyy web-kamerana, ja käynnistyksessä sai joko osoitteen /dev/video0 tai /dev/video1. Ohessa esimerkiksi videokaappauskortille kirjoitetusta ”20-videocapture.rules”-tiedostosta, joka antaa symbolisen staattisen nimen ”sonarvideo” laitteelle, jota kutsumalla saadaan laitteeseen yhteys.

KERNEL=="video*", ATTRS{idVendor}=="1bcf", ATTRS{idProduct}=="2c99", GROUP="video", SYMLINK+="sonarvideo"

Tiedosto on hyvin lyhyt, mutta siitä huolimatta oli kohtalaisen haasteellista löytää oikea tapa sen kirjoittamiseksi. Toisin kuin esimerkiksi virtuaali-com -porteille, joille ohjeita oli helposti löydettävissä. Valmistajan numeron ja tuotenumeron perusteella sidotaan laite symboliseen nimeen.

5.2 ROS

ROS eli robottikäyttöjärjestelmä on avoimeen lähdekoodiin perustuva väliohjelmisto, joka yleisimmin toimii Linuxin päällä, eikä siinä mielessä ole nimestään huolimatta varsinainen käyttöjärjestelmä. ROS määrittelee tavan kommunikoida robotin eri osien välillä. Se pitää myös huolta viestien aikaleimoista ja antaa yhtenäisen tavan datan tallennukseen ja simulointiin, sekä helpon tavan kääriä muita ohjelmistoja ROS-ohjelmistoon. Ros on alun perin kehitetty Willow Garagessa Yhdysvalloissa 2007. Sen jälkeen sitä on alkanut kehittää hyvin moninainen joukko erilaisia robotiikkakehityksestä kiinnostuneita tahoja. Alusta asti kehityksessä on ollut useita instituutioita ja monenlaisia robottialustoja. Tunnetuista yrityksistä esimerkiksi BMW käyttää ainoastaan ROS-pohjaisia ohjelmistoja autonomisten autojen kehitystyössä. (Aeberhard 2016)

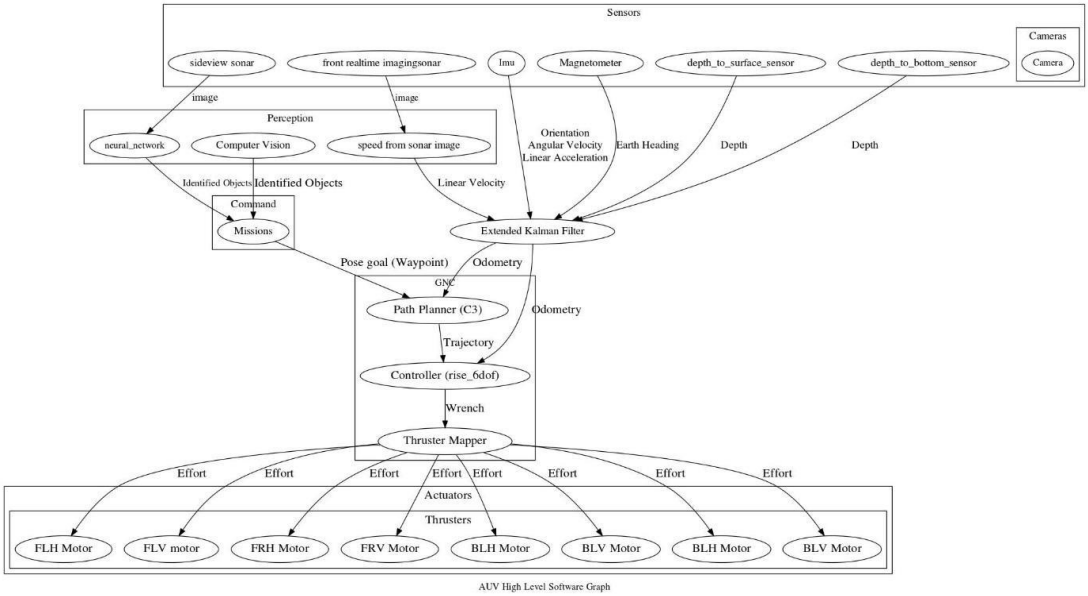
ROS-jakeluversioista tässä projektissa päädyttiin Kinetic Kane-versioon. Vaikka tämä ei olekaan uusin versio eikä edes uusin pitkän tuen versio, on se edelleen erittäin paljon käytetty versio. Sille löytyy myös eniten esimerkkikoodia. Iästään huolimatta sille on saatavilla tuki vielä huhtikuulle 2021 asti. Valintapäätökseen vaikutti myös se, että tämän laitteen esikuva Subjugator käyttää Kinetic Kane- versiota.

ROS on tarkoitettu hyvin modulaariseksi järjestelmäksi, jossa on paljon eri nodeja eli solmuja, jotka kommunikoivat keskenään. Ne käyttävät joko julkaisija-tilaaja-periaatetta tai palvelu-asiakas-mallia. Julkaisija-tilaaja-malli on monelta monelle. Tämä on hyvä asia koska viestejä on helppo kuunnella esimerkiksi vianmäärityksessä ja kehitystyössä, sillä viestejä voidaan kuunnella kaikista väleistä häiritsemättä toimintaa. Datan keräyksessä tällä voitiin helposti tarkkailla, että kaikki halutut viestit saapuivat, ja olivat oikean muotoisia. Viestit voidaan myöskin helposti tallentaa rosbag-työkalulla. ”Rosbag info”-komennolla (kuva 5) voi myös tarkastella, mitä viestejä paketti sisältää.

```
$ rosbag info subset_2019-08-30-12-31-01.bag
path:        subset_2019-08-30-12-31-01.bag
version:     2.0
duration:    10:59s (659s)
start:       Aug 30 2019 12:31:01.82 (1567157461.82)
end:         Aug 30 2019 12:42:01.63 (1567158121.63)
size:        4.1 GB
messages:    170916
compression: none [5330/5330 chunks]
types:       nmea2000/gnss               [3d8f155d15b7c57095d9dcc3eb698cf5]
             nmea2000/speed_og           [fc4cac29a2cc6aff9d72ed87b3dcdd46]
             nmea2000/waterdepth         [897b9a9c715b538ceefd4f8bc741af70]
             padweelsow/speed_ow         [2e4cd182388dbc3eaaa2b97801341a22]
             sensor_msgs/CompressedImage [8f7a12909da2c9d3332d540a0977563f]
             sensor_msgs/Imu             [6a62c6daae103f4ff57a132d6f95cec2]
topics:      /camera_crop/image_rect_color/compressed    19794 msgs    : sensor_msgs/CompressedImage
             /camera_crop2/image_rect_color/compressed   19794 msgs    : sensor_msgs/CompressedImage
             /camera_crop3/image_rect_color/compressed   19794 msgs    : sensor_msgs/CompressedImage
             /camera_crop4/image_rect_color/compressed   19794 msgs    : sensor_msgs/CompressedImage
             /camera_crop5/image_rect_color/compressed   19794 msgs    : sensor_msgs/CompressedImage
             /gnss                                         659 msgs    : nmea2000/gnss
             /imu_bosch/data                             61521 msgs    : sensor_msgs/Imu
             /padweelsow                                  6469 msgs    : padweelsow/speed_ow
             /speed                                       2640 msgs    : nmea2000/speed_og
             /waterdepth                                   657 msgs    : nmea2000/waterdepth
```

Kuva 5. Yhden datankeräys-rosbagin infotiedot

Alla korkean tason arkkitehtuurikuva (kuva 6) siitä miten tieto liikkuu ROS-järjestelmässä. Kaikuluotaimet tuottavat dataa löydetyistä kohteista ja mahdollisista esteistä. Niiden avulla mission planner kertoo halutun paikan. Sensorit kerätään laajennetulle kalman-suotimelle, jonka tuottaman tiedon ja halutun paikan avulla luodaan haluttu reitti. Sen jälkeen lasketaan, miten paljon kullekin moottorille halutaan tehoa, jotta haluttu siirtymä tapahtuu. Rise-kontrolleri tarkkailee, että haluttua siirtymää tapahtuu.



Kuva 6. Korkean tason arkkitehtuuri

5.2.1 ROS-koordinaatistot

TF-kirjasto on ROSin tapa pitää kirjaa eri koordinaatistoista ja niiden välisistä muutoksista. Tässä vaiheessa tällä laitteistolla ei ole käytössä manipulaattoreita tai muita liikkuvia niveliä, joten kaikki muutokset ovat staattisia. Jokaiselle anturille annetaan oma koordinaatisto sekä sen sijainti ja asento suhteessa pääkoordinaatistoon. Pääkoordinaatiston origo sijaitsee laitteen keskipisteessä.

5.2.2 Laajennettu kalman-suodin

Kalman-suodin on digitaalinen suodin, joka kykenee arvioimaan dynaamisen järjestelmän tilaa aikaisempien mittaustulosten perusteella, vaikka nämä olisivat epätäsmällisiä ja sisältäisivät kohinaa. Ratkaisu on rekursiivinen, jokainen päivitetty arvio lasketaan edellisestä tilasta sekä uudesta sisään tulevasta datasta, joten kaikkea aikaisempaa mittausdataa ei tarvita arvion laskemiseksi (kalman-suodin 2019). Kalman-suodin on rajoitettu lineaariseen oletukseen. Tähän ongelmaan on kehitetty laajennettu kalman-suodin, joka toimii myös epälineaarisella datalla (Kalman filter 2019). (Moreno & Pigazo 2009, 71).

Alun perin oli tarkoituksena käyttää Subjugator-projektissa käytettyä suodinta. Sen dokumentaatio oli kuitenkin niin rajallista ja muokattavuus heikkoa, että päädyttiin etsimään muita ratkaisuja. Lopulta päädyttiin ”robot localisation ROS”-pakettiin. Sen hyvä puoli on, että se on tarkoitettu käytettäväksi hyvin erilaisissa roboteissa, mistä johtuen erilaisten anturitietojen liittäminen siihen oli kohtuullisen yksinkertaista. Paketti tukee myös lähes rajatonta määrää antureita. Myös samoja antureita voi olla useampia, esimerkiksi inertia-mittauksia. Anturien asennot tulevat tf-muunnosten kautta. Anturitietojen pitää olla tyypiltään aikaleimattuja sekä sisältää kovarianssi-matriisin, joka kertoo anturitiedon luotettavuuden. (Moore 2018)

Teknisesti Subjugator-projektin suodin on parempi ainakin siinä mielessä, että laskut toteutetaan quaternioneilla, kun taas robot localization-paketin filtteri käyttää sisäisesti euler-kulmia, vaikka sisääntulona on siinäkin quoternionit (Moore 2015). Tämä voi

tietyissä tilanteissa, esimerkiksi kolmiulotteisissa liikkeissä aiheuttaa niin sanotun gimbal lock -tilanteen, jossa menetetään yksi vapausasteista. (CHRobotics n.d,).

5.2.3 Missions

Mission planner eli tehtäväsuunnittelija on suoraan lainattu Subjugator-projektista. Siinä on myös hyvä dokumentaatio erilaisten tehtävien kirjoittamiseen. Tehtävien avulla suoritetaan haluttu autonominen toiminta. Tehtäviä voidaan myös suorittaa rinnakkain. Esimerkiksi silloin kun suoritetaan kohteen etsintää pohjasta halutulla etsintäkuviolla, laajenevalla neliöllä, voidaan myös saman aikaisesti suorittaa tehtävää, jossa väistetään eteen tulevia esteitä. Tehtäväsuunnittelija käyttää python-kielen coroutine-kirjastoa ja floridalaisten muokattua ROS-asiakaskirjastoa txROS.

5.2.4 Path planner

Path planner eli reitin suunnittelija on suoraan lainattu Subjugator-projektista. Tämän tehtävänä on laskea nykyisen paikan ja tehtäväsuunnittelijalta tulevan halutun paikan avulla reitti halutulle paikalle, ottaen huomioon laitteen liikkumisen rajat.

5.2.5 Controller

”Robust integral of the sign of the error Control” on floridalaisten kehittämä kontrolleri, joka ottaa tiedot reitin suunnittelijalta ja kalman-suotimelta ja laskee näiden perusteella tarvittavan suunnan ja nopeuden. (Fischer Hughes Walters Schwartz & Dixon 2014)

5.2.6 Truster mapper

Truster mapper saa tiedon halutusta liikkeestä kontrollerilta. Se laskee eri potkurilaitteille halutut työntövoimat ja antaa käskyt potkurille. Trust mapper osaa myös ottaa huomioon tilanteet, joissa jokin potkurilaitteista on vikaantunut. Niissä tilanteissa se kykenee laskemaan käskyt vain toiminnassa olevien potkurilaitteiden mukaan.

5.3 Arduino

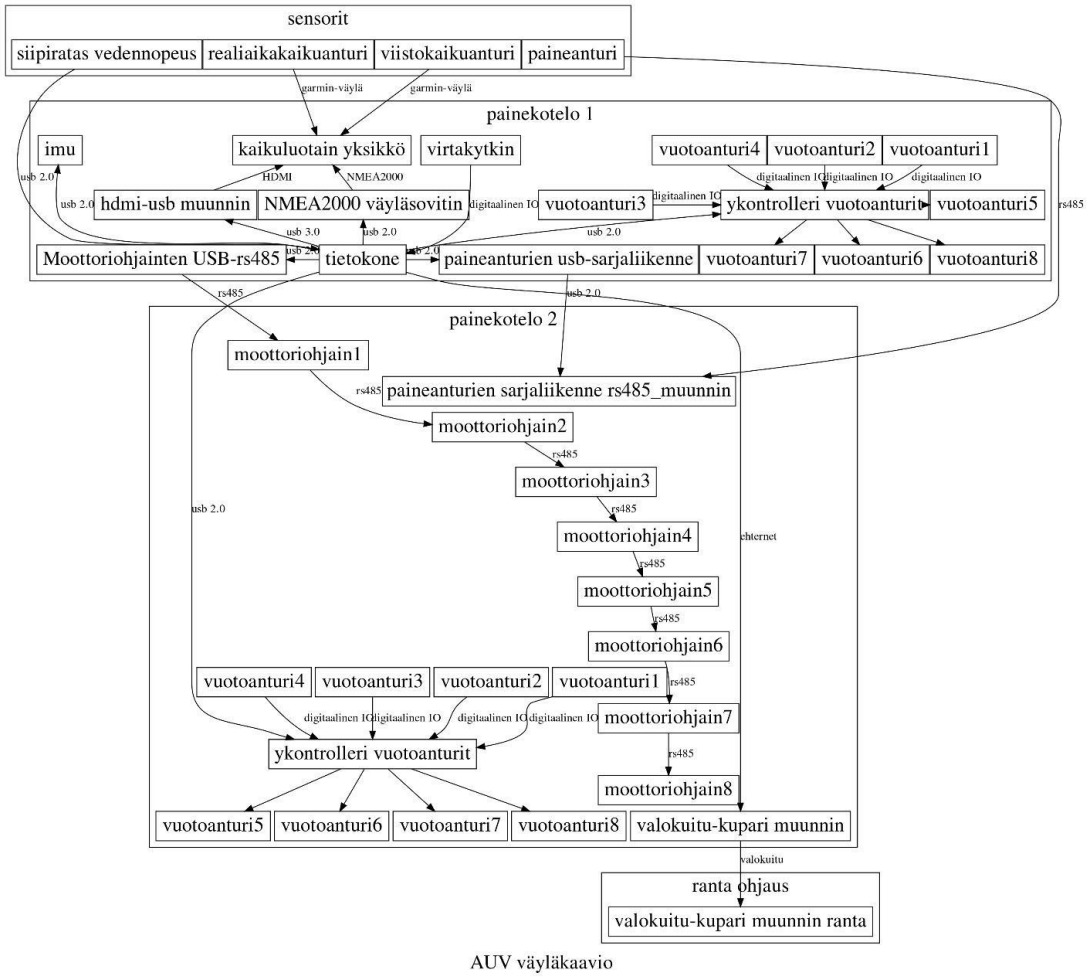
Arduino-ekosysteemi on helpossa lähestyttävyydessään ja laajassa laitetuessaan parantanut huomattavasti kehittäjien ja harrastelijoiden mahdollisuuksia. Vaikka arduino-ohjelmointi ei aina olekaan laskennallisesti tehokkainta mahdollista, ovat nopeammat arduino-yhteensopivat mikrokontrollerit, kuten teensy-perhe, niin nopeita, että harvassa tilanteessa on tarvetta alemman tason koodin kirjoittamiselle. Yksinkertaisille antureille, kuten käyttämällemme MS5803-paineanturille, atmel mega 328p on edelleen riittävän nopea.

Arduino-laitteen yhdistämisessä ROS-järjestelmään on käytössä kaksi tapaa, joista molempia käytettiin. Ensimmäinen tapa on käyttää rosserial-kirjastoa. Tällöin ROS-solmu kirjoitetaan mikrokontrolleriin ja isäntäkoneesta yksinkertaisesti käynnistetään geneerinen serial_node, eli sarjaliikennesolmu halutulla portilla. Tässä hyödynnettiin staattisia symbolisia porttiosoitteita. Myös sarjaliikenteen nopeus pitää määritellä. Huomioitavaa on, että se on määriteltävä myös silloin, kun se ei vaikuta liikenteen nopeuteen. Esimerkkinä tilanne, jossa käytetään teensy 3.6 -kontrolleria, joka toimii joka tapauksessa ”full speed usb” -nopeudella.

Toinen tapa on tehdä tavanomaisempi arduino-ohjelma mikrokontrollerille, ja jollain itse valitulla protokollalla siirtää data isäntäkoneelle. Isäntäkoneella voidaan tehdä vielä raskaammat laskutoimitukset datalla ja vasta tämän jälkeen julkaista käsitelty data ROS-topikkiin. Tässä toimintatavassa on hankalampaa se, että on mietittävä itse siirtoprotokollaa. Hyötynä tietysti on suurempi joustavuus.

6 VÄYLÄARKKITEHTUURI

Laitteeseen tulee useita eri väylärakenteita. Alla kuva rakennetun laitteen väylärakenteista (kuva 7).



Kuva 7. Rakennetun laitteen väyläkaavio

6.1 USB-väylä

USB-väylä on tietokoneen käytetyin väylä liitettäessä laitteita tietokoneeseen. Siitä on modernissa tietokoneessa käytössä monia eri versioita. Tässä laitteessa käytössä on USB 2.0- ja USB 3.0 -väylät. Tässä laitteessa emolevyn USB 3.0 -väylään on kytketty yksinomaan USB-HDMI -kaappauskortti, jolla saadaan kaikuluotainkuva koneelle. Tällä on pyritty siihen, ettei väylä ole varattuna, kun kuvaa on saatavilla, ja että video saadaan mahdollisimman reaaliaikaisesti koneelle. 1080p-kuvan siirto HDMI-kaapelissa tarvitsee kaistaa 3.2 Gigabittiä sekunnissa (HDMI 2019). Tämä on melko lähellä USB3.0-portin teoreettista maksimisiirtokapasiteettia, 5 Gigabittiä sekunnissa

(USB3.0 2019). Muut usb-laitteet on kytketty USB 2.0-väylään. Niiden kaistavaatimukset ovat huomattavan paljon pienempiä.

6.2 RS485-väylä

RS485 on edelleen hyvin käytetty differentiaali-signaalilla toimiva sarjaliikenneväylä. Väylä on hyvin häiriöitä kestävä ja sallii pitkiäkin siirtoetäisyyksiä, jopa sadoista metreistä muutamaan kilometriin. Väylän siirtonopeus on yleensä joitain satoja kilobittejä sekunnissa. Joillain toteutuksilla voidaan saavuttaa muutaman megabitin nopeuksia. (RS-485 2019).

Laitteessa on käytössä kaksi erillistä RS485-väylää. Moottoriohjaukset ovat kaikki yhden väylän päässä. Väylä on differentiaali-muuntimien jälkeen liitetty TTL–RS232 -muuntimen avulla tietokoneen sarjaliikenneporttiin. Näin toimittiin pienemmän vasteajan saavuttamiseksi verrattuna ratkaisuun, jossa väylä olisi kytketty USB–TTL-adapterilla USB-porttiin.

Toinen RS485-väylä on paineantureilla. Tähän ratkaisuun päädyttiin, koska paineanturit ovat painerungon ulkopuolella. Läpivientien määrää halutaan rajoittaa. Tulevaisuudessa saattaa olla tarvetta useammille antureille. Silloin kaikki anturit voidaan liittää yhteen rs485-väylään ja tuoda yhden läpiviennin kautta painerunkoon. Toinen syy ratkaisuun oli parempi häiriönkesto.

6.3 NMEA2000-väylä

NMEA2000 on venepuolen väylä, joka fyysisesti perustuu CAN-väylään (Jukka Penttinen 2017, 13). Kyseessä on kaikuluotainyksikön väylä, josta saatavat nopeus-, syvyys-, paikka- ja lämpötilatiedot on toteutettu NMEA2000-väylällä, kuten moderneissa veneilykäyttöön tarkoitetuissa kaikuluotaimissa yleensäkin. Ohjelmistokirjastona NMEA2000-väylälle käytettiin Timo Lappalaisen avoimeen lähdekoodiin perustuvaa kirjastoa, jota hyväksi käyttäen ohjelmoitiin Ros-solmu. Teensy 3.6 toimii väylämuuntimena, ja on itse liitettynä USB-väylällä tietokoneeseen. (Lappalainen 2019)

6.4 Ethernet

Ethernet on käytössä laitteen ollessa ROV-tilassa eli kauko-ohjauksessa. Päätietokoneella on Gigabitin rj45-portti, josta cat 6 -kaapelilla siirrytään alempaan painerunkoon, jonka laipassa on rj45-kuitumuunnin (TP-Link 2019). Kuidulla siirretään data rannalle, jossa taas kuitu muutetaan kupari-Ethernetiksi, helpottamaan liittämistä kannettavalle tietokoneelle. Kuitu valittiin siirtomediaksi paremman häiriökeston vuoksi. Hinnaltaankin kestävä valokaapeli on halvempaa kuin teollisuuskäyttöön tarkoitetut cat6-kaapelit. Tosin valokuidun päättämisestä tulee kustannuksia, joita ei Ethernet-kaapelin kohdalla synny. Ethernetiin on mahdollista tehdä itse liittimet.

6.5 I2C-väylä

I2C-väylä on tässä laitteessa käytössä paineanturin ja siihen kiinteästi liitetyn mikrokontrollerin välillä. Periaatteessa väylän spesifikaation mukaan anturin olisi voinut saada toimimaan suoraan I2C-väylällä anturilta painerungon sisään, kun maksimi kapasitanssi väylälle on 400 pF (telos Systementwicklung n.d.) ja esimerkiksi cat 6 -kaapeli on 46 pF /metri (extron n.d.) ja pituutta kaapelille olisi tullut 2-3 metriä, mutta toimintavarmuuden vuoksi väylä muutetaan mikrokontrollerilla rs485-väyläksi heti anturikotelolla. Lähistöllä on kuitenkin paljon moottorikaapeleita sekä moottoreita tuottamassa häiriöitä.

7 PAIKKA- JA NOPEUSTIEDON HAASTEET

Paikka- ja nopeustiedon saaminen pinnan alla eroaa suurilta osin vastaavien tietojen hankinnasta ilmassa tai maalla, vaikka joitain samanlaisuuksiakin on. Suurin ero on GPS:n ja vastaavien globaalien paikannusjärjestelmien puuttuminen. Radiosignaalit eivät etene vedessä pintakerroksia syvemmälle, joten näitä ei voida käyttää.

7.1 Erilaiset vedenalaiset paikannusmenetelmät

Vedenalaiseen paikannukseen ja nopeustietoon on useita erilaisia lähestymistapoja. Alla esitellään niistä yleisimpiä.

7.1.1 Paikalliset kiinteät äänen kolmiomittaukseen perustuvat järjestelmät

Joillekin alueille, lähinnä öljyntuotantoon, on rakennettu vedenalaisia paikallisia äänipohjaisia navigointijärjestelmiä. Nämä toimivat hyvin pienillä alueilla ja ovat usein suljettuja järjestelmiä. Vastaavaa globaalia paikannusta, jossa ei ole kumuloituvaa virhettä, ei veden alla ole.

7.1.2 Slam-algoritmit

Slam-algoritmeilla tarkoitetaan samanaikaisesti kartoittavia ja paikantavia algoritmeja. Maalla nämä ovat useimmiten käytössä sisätiloissa, mobiilirobotteja käytettäessä. Joissain määrin niitä käytetään myös ulkotiloissa.

Monet perinteiset slam-algoritmit perustuvat kulmien tunnistukseen, ja sopivat siten parhaiten ihmisen muokkaamiin ympäristöihin, joissa on paljon keinotekoisia muotoja, tai muotoja ylipäätään. Vedenpohja taas voi olla hyvinkin monotoninen.

Vedenalaisiin toteutuksista on jonkin verran tieteellisiä julkaisuja, kuten avoimeen lähdekoodiin perustuva dolphin slam (Silveira, Guth, Drews-Jr, Ballester, Machado, Moraes, Nelson & Botelho. 2015). Tässäkin kaikuluotain on hyvin erityyppinen kuin,

mitä tässä rakenteilla olevassa laitteessa käytössä oleva, joten tämäkään ei ainakaan sellaisenaan sovellu käytettäväksi tässä projektissa.

Tässä projektissa tutkittiin olemassa olevia slam-algoritmeja, mutta ainakaan vielä ei ole löytynyt tähän käyttöön suoraan soveltuvaa. Parhaaseen tulokseen päästäisiin käyttämällä esimerkiksi Deep Slam-tyyppisiä algoritmejä, jossa slam-järjestelmän front end on korvattu neuroverkolla. (DeTone 2017).

7.1.3 Nopeustiedon laskeminen kaikuluotainkuvasta

Yksi vedenalainen paikannusmenetelmä on nopeustiedon laskeminen kaikuluotainkuvasta. Tämän heikkoutena on sen riippuvuus pohjatyypistä. Pohjassa ei ole jatkuvasti sellaista muotoa, josta nopeutta voisi laskea. Joillain pohjatyypeillä taas nopeus on saatavissa lähes koko ajan. Toinen haaste tässä toteutuksessa on, ettei vielä ainakaan ole löytynyt tietoa siitä, että kukaan olisi yrittänyt vastaavaa.

7.1.4 Inertiamittaus

Imu eli inertian mittausyksikkö on yleisesti käytetty anturi. Näiden hinnat alkavat muutamista euroista ja kalleimmat ovat kymmeniä tuhansia euroja. Näissä lyhyen ajan tieto on yleensä käyttökelpoista, mutta anturivirheet kumuloituvat nopeasti niin, ettei pelkästään näiden varaan voida luottaa. Tilanne on hyvin erityyppinen ilmassa ja veden alla ainakin lineaarinopeudessa ja siitä lasketussa paikkatiedossa. (laskuesimerkki) Tilannetta voidaan johonkin pisteeseen asti helpottaa erilaisilla filtteröintiratkaisuilla kuten ekf eli laajennettu kalman-suodin. Joissain tapauksissa virhe voidaan poistaa lähes kokonaan (aalto esimerkki). Nämäkin tosin luottavat siihen, että jossain kohtaa voidaan luotettavasti todeta, että ollaan pysähdyksissä. Tätä ei voida luotettavasti todeta pinnan alla, vaikka potkurit olisivat sammutettuina. Merivirrat kuljettavat kuitenkin usein laitetta johonkin suuntaan.

7.1.5 Siipiratasvedennopeusanturi

Siipiratas mittaa ainoastaan veden virtausta, ja näin ollen saadaan parhaimmillaankin tällä tavalla laskettua pelkästään nopeus veden suhteen. Tämä voi tietyissä tilanteissa korreloida hyvinkin maanopeuden kanssa esimerkiksi pienissä järvissä, mutta paikoissa joissa vesi virtaa, ei näin saatu nopeus ole luotettava. Anturi on myös helposti vikaantuva. Tätä ominaisuutta voidaan tosin parantaa asentamalla useampi anturi. Siipiratas on hyödyllinen ainoana nopeusanturina lähinnä testaus- ja opetusvaiheessa.

7.1.6 Dopler-nopeus-loki

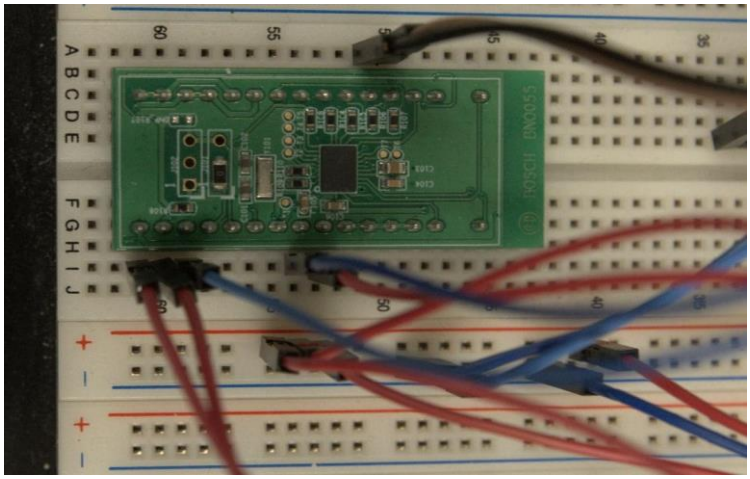
DVL eli dopler-nopeus -loki lähettää kahta tai neljää ultraäänipulssia pohjaan. Äänien palatessa voidaan niiden dopler-siirtymästä laskea nopeus, kahden äänen tapauksessa yhdessä ulottuvuudessa ja neljällä kahdessa ulottuvuudessa. Tämä on kaupallisissa kalliimman luokan vedenalaisissa laitteissa standarditapa tehdä nopeusmittaus. Ongelmana tässä on lähinnä hinta. Esimerkiksi floridalaisten käyttämä DVL maksaa noin 20.000 € eli se on tämän projektin budjettin ulottumattomissa. Pitkillä etäisyyksillä myös tässä nopeuden mittaustavassa on ongelmana virheen kumuloituminen. Laadukkaalla laitteella tämän projektin tavoittelemissa toimintasäteissä ongelma on kuitenkin melko rajallinen.

7.2 Projektin käyttöön valitut tavat

Tässä projektissa tavoitteena on kompensoida eri tapojen heikkouksia käyttämällä yhtä aikaa useampaa eri tapaa ja tekemällä näistä sensorifuusio käyttäen EKF- eli laajennettua kalman-suodinta. Siinä IMU:n kiihtyvyysanturi tuottaa hetkellisesti tarkkaa tietoa, joka voidaan kalibroida kaikuluotainkuvasta lasketulla nopeudella aina kun se on riitettävän luotettavana saatavilla. Pinnalla ollessa saadaan paikka ja nopeus GPS tiedosta.

7.2.1 IMU

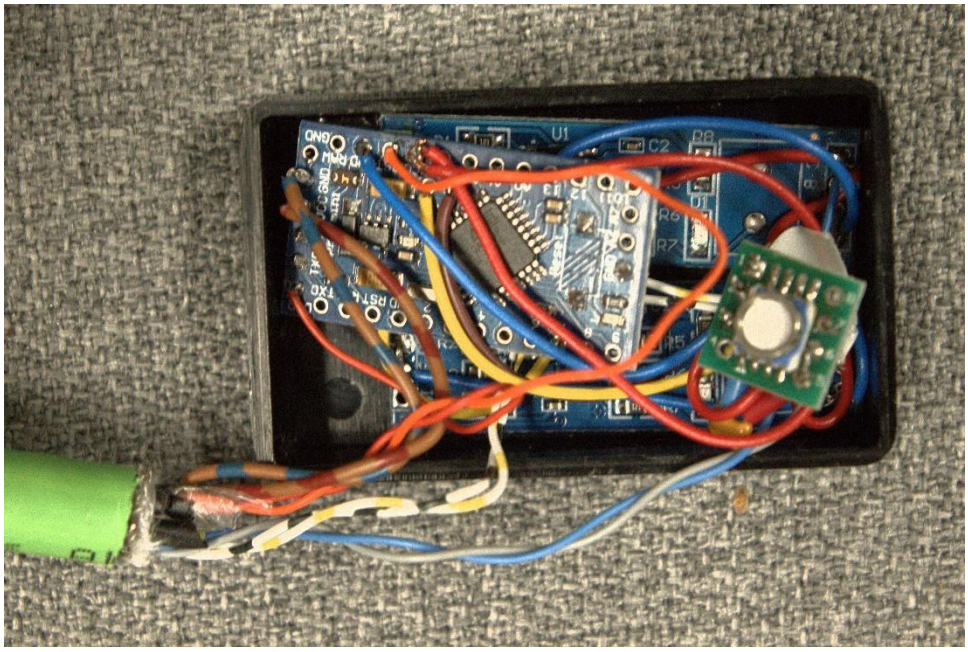
Käytetään Bosch:in valmistamaa BNO055 absoluuttiorientaatioanturia. Anturilla on monia hyviä ominaisuuksia kuten sisäinen mikrokontrolleri, jolla anturitiedot fuusioidaan ja saadaan ulos valmiiksi kompensoitua ja vakautettua dataa. Kyseiselle anturille löytyy myös valmis ROS node avoimena lähdekoodina. Anturista löytyy I2C-, SPI- ja sarjaliikennekommunikaatiotavat, joista tämän projektin käyttöön valikoitui sarjaliikenne, käyttäen sarjaliikenne–usb-muunninta. Valinta johtui lähinnä valmiina löytyneestä ROS-nodesta, joka oli kirjoitettu käyttäen sarjaliikennekommunikaatiota. Anturitiedon kumuloituvia virheitä, sekä ROS-liitäntää testattiin koekytkentälevylle tehdyllä kytkennällä (kuva 8). (Bosch-sensortec 2019)



Kuva 8. Bosch-inertiamittausanturi koekytkentälevyllä

8 SYVYYSTIETO

Syvyystieto veden alla voidaan jakaa kahteen osaan, etäisyyteen pohjasta ja etäisyyteen pinnasta. Etäisyys pintaan saadaan yksinkertaisesti painemittauksella. Anturiksi valittiin MS580314BA01-00. Anturin valmistajan datalahden mukaan tarkkuuden pitäisi pidemmälläkin aikavälillä säilyä 30cm:n sisällä (TE connectivity 2017). Tämä on jo sinällään riittävää, mutta tarkkuutta voidaan vielä lisätä useammalla anturilla. Harkinnassa on myös käyttää useampaa anturia ja laskea niiden eroista asentoa. Tällä ei tietysti päästä suureen tarkkuuteen, mutta saatavaa tietoa voidaan karkeasti verrata IMU-tietoon. Jos näissä esiintyy suurta heittoa, laite ymmärtää, että jokin on pielessä, sammuttaa itsensä ja palaa pintaan. Paineanturi (kuva 9) koostuu painesensorista, arduino pro ministä sekä rs485-muuntimista.

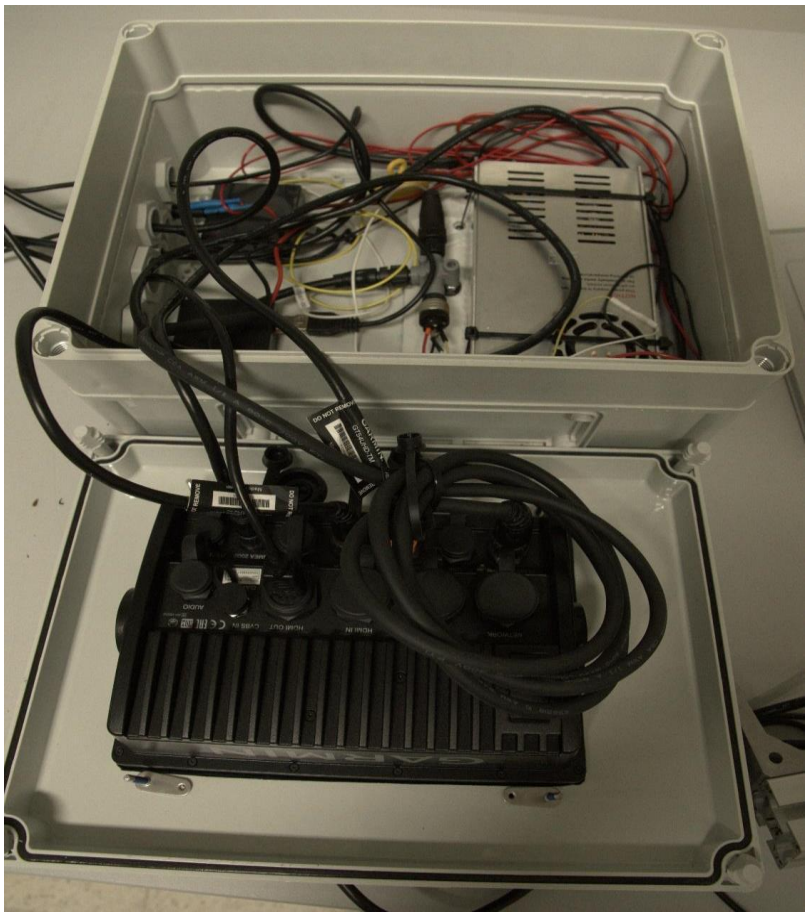


Kuva 9. Paineanturi ennen hartsivalua

Etäisyys pohjaan saadaan kaikuluotainyksikön tuottamana NMEA2000-väylän kautta. Tämän tarkkuudesta ei ole löytynyt tarkkaa tietoa, mutta mittakepillä datankeräysvaiheessa joitain kertoja testattuna, tarkkuus näyttäisi olevan myös joidenkin kymmenien senttimetrien tarkkuudessa. Erilaiset pohjanmuodot voivat toki tähän vaikuttaa.

9 DATANKERUU

Datan keruu toteutettiin kiinnittämällä anturit erilaisiin veneisiin ja anturitiedot tallennettiin kannettavalle tietokoneelle. Datankeruuoperaatiolla oli useita eri tavoitteita. Haluttiin opetella kalibroimaan ja käyttämään kaikuluotaimia sekä testata antureiden ja väylien ohjelmistoja. Testi käyttää Ros-solmuja ja tiedonkeruuta, sekä kerätä dataa neuroverkkojen opetukseen ja algoritmien kalibrointiin. Merkittävintä neuroverkkopohjaista nopeuslaskentaa ajatellen on, että pinnalla ajaessa voidaan taltioida GPS-nopeus, jota ei veden pinnan alla ole käytettävissä. Datankeräyslaitteistossa ROS-ohjelmisto pyöri kannettavalla tietokoneella, johon laitteet oli kytketty usb-väylien välityksellä. Laitteet on asennettu Fidoboxin koteloon. Kotelossa (kuva 10) vasemmalla virtalähde, oikealla ylhäällä NMEA2000-muunnin, oikealla alhaalla Inertia-mittaus, kannessa kaikuluotainyksikkö.



Kuva 10. Datankeräyslaitteiston kotelo kansi avattuna

Anturit kiinnitettiin veneeseen alumiiniprofiilista tehdyllä telineellä. Teline osoittautui hyvin toimivaksi ja muokkautuvaksi. Dataa kerättiin viidellä eri veneellä, joihin kaikkiin teline kiinnittyi pienellä säädöllä ruuvipuristimia hyväksi käyttäen alle puolessa

tunnissa. Tämän kaltaiseen ratkaisuun päädyttiin koska hankkeella ei ollut resursseja hankkia omaa venettä, ja kiinnitys piti tapahtua niin, ettei veneisiin tullut reikiä tai muita vaurioita. Kuvassa 11 on nähtävissä anturiteline Rauman pelastuslaitoksen veneeseen kiinnitettynä.

Kuva 11. Datankeräyslaitteisto veneeseen kiinnitettynä

Ensimmäisestä datankeruukokeilusta oppineena kirjoitettiin ROS launch –tiedosto, jolla kaikki eri ROS-solmut käynnistyivät yhdellä komennolla. Tässä yhteydessä selvisi myös udev-sääntöjen tärkeys. Tämä nopeutti järjestelmän käynnistymistä merkittävästi. Datankeräyksessä käytettiin rosbag-ohjelmaa, joka tallentaa haluttujen aiheiden viestit. Viestit sisältävät aikaleiman, jolloin tiedetään ajallisesti eri anturitietojen riippuvuus. Kaikuluotaimelta tallennettiin 5 eri kuvaa, reaaliaikakuvaa edestä ja veneen alta, sivukuvaa kahdelta eri kaikuanturilta sekä viistokaikukuva. Jokainen kuva on kooltaan 530 x 500 pikseliä. Kuvat tallennettiin rosbag-viesteinä lievästi pakattuna. Vaikka tämä hieman huonontaakin kuvanlaatua, säästää se tallennustilassa. Pakattunakin dataa kertyy noin 20 Gigatavua tunnissa.

10 NEUROVERKOT

Tässä opinnäytetyössä käydään pintapuolisesti läpi neuroverkkojen käyttöä projektissa. Tarkoituksena on kirjoittaa diplomityö enemmän tähän puoleen keskittyen. Pintapuolinen yleiskuva on kuitenkin hyödyksi. Neuroverkot olivat suuri osa projektia ajankäytöllisesti sekä myös kiinnostavin kohta. Työssä käytettiin Googlen tensorflow-kirjastoa.

Työssä käytiin läpi merkittävä määrä muiden tuotoksia ja tieteellisiä julkaisuja aiheesta etsittäessä tähän käyttöön toimivia ratkaisuja. Ensimmäiseksi lähdin tutkimaan jotain reinforcement learning -tyyppistä algoritmia. Dronejen ohjaamiseen näistä löytyi ROS:n gazeboa ja tensorflow:ta hyödyntävä deep reinforcement learning drone control, jonka lähdekoodi on jaettu github-palvelussa. Tätä koodia tutkittiin ja kokeiltiin. Sitä ei ollut muutamaan vuoteen ylläpidetty ja se vaati jonkun verran muokkaamista ennen kuin kaiken sai käännettyä ja gazebo-simulaation käynnistettyä. Tämä vaihe oli hyvin opettavaista, vaikkei sitä olekaan tarkoitus ensimmäisessä vaiheessa ottaa käyttöön AUV-ohjauksessa. Tämän jälkeen alettiin tutkia tunnistamista viisto-kaikuluotainkuvasta ja nopeuden saamista kaikuluotainkuvasta. (Fischer 2018)

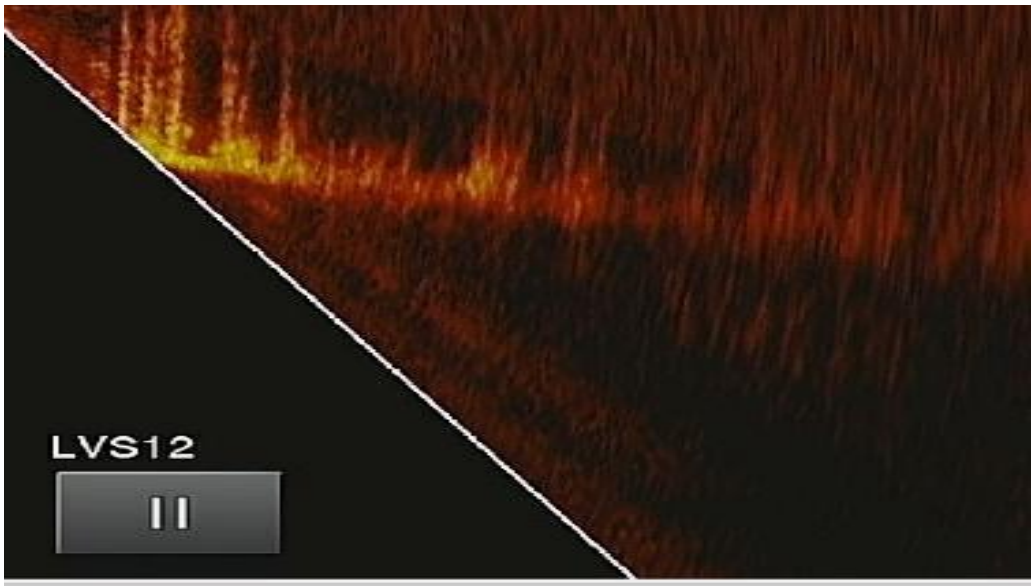
10.1 Nopeus kaikuluotainkuvasta

Yhtä perinteistä kameraperusteista visuaaliseen odometriaan perustuvaa algoritmiä rqt_svo:ta kokeiltiin, muttei se ainakaan sellaisenaan soveltunut käytettäväksi reaaliaikaisen eteenpäin luotaavan luotaimemme kanssa. Tulos ei näyttänyt parantuvan erilaisilla filttereillä kuten kandy-filtterillä. Kohinaa on liikaa. Tästä syystä alettiin tutkia neuroverkoilla toteutettua ratkaisua. (Forster, Pizzoli & Scaramuzza 2014)

Kirjallisuutta tutkittaessa ja esimerkkejä etsiessä löytyi udacityn projektin, jossa opetetaan autonomista autoa, johon liittyen githubista löytyi vapaata lähdekoodia. Kiinnostavaksi osoittautui tutkimuksen osa, jossa opetetaan neuroverkolla ohjaamaan autoa (kuva 12) niin, että se pysyy kaistaviivojen sisällä (Higgins 2019)

Vaikka tietynlaisesta kulmasta tarkasteltuna ongelma on hyvin erilainen kuin nopeuden tutkiminen kaikuluotainkuvasta (kuva 13). kuvat ovat hyvin erilaiset, ovat ne tavallaan myös hyvin samankaltaisia. Molemmissa tutkitaan sarjaa kuvia, joilla on ajallinen riippuvuus. Yksi neuroverkkojen suurimmista eduista on, että ne on helppo opettaa pintapuolisesti hyvinkin erilaisiin tehtäviin käyttäen täysin identtisiä neuroverkkoarkkitehtuureja. (udacity 2019)

Kuva 12. Ohjauspyörän asennon opetukseen käytettyä RGB kamera kuvaa (Cameron 2016)



Kuva 13. Eteenpäin suunnattu kaikuluotain

Udacity oli järjestänyt kilpailun, jonka finalistien työt olivat vapaassa jaossa githubissa. Niitä tutkimalla päädyttiin kokeilemaan komanda-ryhmän koodia, joka näytti

hyvin tehdyltä. Jossain määrin myös koodin sisältö oli selitetty. Tässäkään koodia ei ollut muutamaan vuoteen ylläpidetty, joten monet tensorflow-funktiot ovat muuttuneet. Pienellä ajankäytöllä ja tensorflown dokumentaatiota tutkimalla koodi saatiin kuitenkin kääntymään.

Seuraavaksi piti päättää, missä datamuodossa neuroverkkoa lähdetään opettamaan. Veneellä on kerättyä dataa, joka sisältää kaikuluotauskuvan sekä gps-nopeuden, molemmat aikaleimoilla rosbag-muodossa. Jos lähdetään opettamaan neuroverkkoa suoraan rosbagiä reaaliajassa ajamalla, se on todella hidasta. Silloin muistiin ladataan kuva kerrallaan samalla, kun viestejä julkaistaan.

Tässä lähdettiin samaan suuntaan kuin udacityn antaman esimerkin kanssa. Ross Wightman on tehnyt python-scriptin, joka ottaa rosbagista kuvat, numeroi ne ja laittaa kansioon, sekä tekee csv-tiedoston, jossa on ohjauspyörän asentotieto, aikaleima ja aikaleimaa vastaavien kuvatiedostojen nimet. (Wightman 2019)

Idea vaikutti huomattavasti paremmalta kuin rosbagistä suoraan opettaminen ja nopeus on kertaluokkia suurempi. Antti Halla auttoi tekemällä vastaavan python-ohjelman, joka ottaa datasta vastaavat tiedot, kaikuluotainkuvat, gps-nopeuden, syvyyden ja kiihtyvyysanturin tiedon. Tässä vaiheessa käyttöön ei kuitenkaan tule kuin gps-nopeus ja kaikuluotainkuva. Jatkossa myös muita tietoja voidaan hyödyntää.

Komandan koodiin tehtiin tarvittavat muutokset ja alettiin opettaa neuroverkkoa käyttäen sukelluslaitteen tietokonetta. Testejä tehtiin noin 20 minuutin rosbag-tallennuksilla. Jatkossa on tarkoitus liittää usempia rosbag-tiedostoja yhteen ja opettaa verkko käyttäen kaikkea kerättyä dataa kerralla. Rosbagien yhtymäkohdissa tulee virhettä opetusmateriaaliin, tätä ei ole tarkoitus huomioda, vaan katson virheellisten kuvien suhteen olevan olemattoman pieni, noin 4 / 30 000.

10.2 Kohteiden tunnistus kaikuluotainkuvasta

Kohteen tunnistus kuvasta on yksi tutkituimmista neuroverkkojen käyttökohteista. Ongelma on hyvin ymmärretty ja monet pitävät sitä jo ratkaistuna. Eriäviäkin mielipiteitä

on ja esimerkiksi Geoffrey Hinton pitää kuvan tunnistuksessa yleisesti käytettyä tapaa ja erityisesti max pooling -lähestymistapaa vääränä. Hän uskoo kapseliverkko-tyyppisten ratkaisujen parantavan tunnistusta, koska kuvan osien avaruudelliset suhteet eivät katoa. Toistaiseksi nämä tavat ovat kuitenkin vielä hitaampia kuin perinteisemmät neuroverkkoratkaisut, joten näitä ei lähdetty kokeilemaan. (Misra 2019).

Ongelmaa lähestyttiin etsimällä hyvin toimivia tunnettuja ratkaisuja. Nopeasti löytyikin ROS-yhteensopiva paketti, joka tuntui toimivan suhteellisen helposti, ja kokeiluissa kannettavan tietokoneen webkamera tunnisti ihmisen ja muitakin kohteita hyvällä tarkkuudella. (Odabasi 2019)

Suurin ongelma nopeuden opettamisessa kaikuluotainkuvasta on opetusmateriaalin kerääminen. Siinä missä internetistä löytyy riittävä määrä normaaleja kuvia melkein minkä vain eläimen tai esineen tunnistuksen opetukseen, on kaikuluotainkuvia saatavilla hyvin rajallisesti. Hylkyjen kuvia viistokaikukuvana löytyy jonkun verran, muista kohteista vain hajanaisia yksittäisiä kuvia. Riippuen kohteen vaikeudesta ja neuroverkon arkkitehtuurista kuvia tarvitaan ainakin 300-1000 kappaletta tunnistettavaa kohdetta kohden. Eli tähänkin data on kerättävä itse.

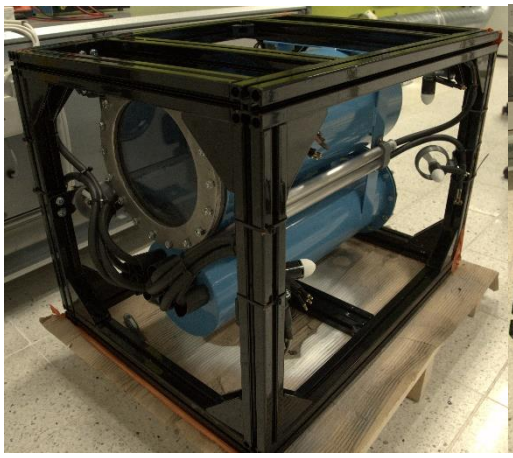
Keskusteluissa tekoälyasiantuntija Jussi Bergmanin kanssa datan hankinnan ongelmasta Bergman ehdotti, että olisiko jotain tapoja luoda keinotekoisesti riittävän hyvää opetus materiaalia. Usein käytetään erilaisia käännöksiä ja filttereitä, joilla saadaan lisättyä opetusmateriaalin määrää. Tämä on hyödyllistä, mutta edelleen mieltä pohditutti, voisiko olla muita keinoja, joita muut eivät ole vielä käyttäneet, tai jotka eivät muunlaisissa ongelmissa ole välttämättä hyödyllisiä. Pohdinnan tuloksena paras idea on käyttää taidetyylin muuttamiseen tarkoitettuja GAN-algoritmeja. Normaalisti näitä on käytetty muuttamaan esimerkiksi oma selfie-kuva niin, että se näyttää kuin se olisi Salvador Dalin maalaama taulu. Tavallaan se yhdistää artistisen tyylin ja lähdekuvan. Jos annetaan artistiseksi tyyliksi viistokaikukuva ja lähdekuvaksi vaikka ihminen makaamassa nurmikolla, saadaanko tästä kuva, joka vastaa viistokaikukuvaa vedenpohjasta, jossa makaa ihminen.

10.3 Esteen tunnistus

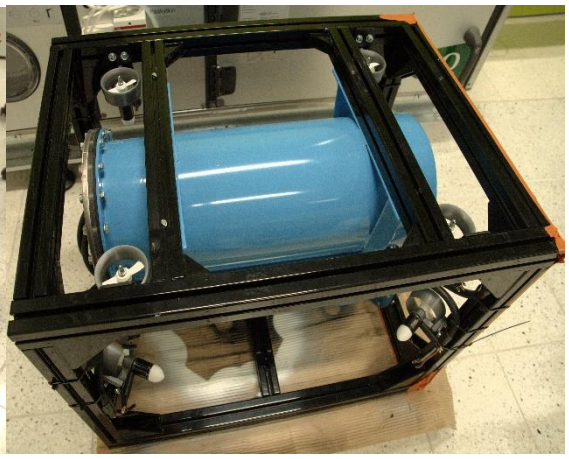
Eteen tulevat esteet pitää pystyä tunnistamaan, jotta ne voidaan kiertää. Vaikka herkimmät osat ovat suojattuna alumiinikehikon sisällä, on silti tärkeää havaita esteet jo ennen törmäystä. Periaatteessa tämä on suhteellisen helppoa eteenpäin suunnatusta reaaliaikakaikukuvasta. Yksinkertainen perinteinenkin konenäkö käyttäen esimerkiksi openCV-kirjastoa pitäisi olla riittävä. Ainut suurempi haaste tässä on kerätyn datan perusteella, että paikoin, jos kasvillisuutta on erittäin paljon kuvassa, voi se näyttää kiinteältä esteeltä. Tämän kaltaisten tilanteiden takia tämä toteutetaan myös todennäköisesti neuroverkolla, koska tällä tavalla toteutettuna tarkkuutta voidaan parantaa datamäärän lisääntyessä. Näiden tapauksien erottamiseen tarvitaan todennäköisesti suurehko esimerkkijoukko opetukseen.

11 TULOKSET

Vaikka aikataulusta johtuen itse laitteen sukellustestausta ei ehditty aloittaa ennen opinnäytetyön valmistumista, on hyvin paljon eri osa-alueita testattu. Testauksia on tehty fyysisesti tiiviyskokeilla, yksittäisten potkurien testausta vesialtaassa, ohjelmistoja datan keräyksessä, neuroverkkoja opettamalla ja ohjelmisto-osuuksia simulaatioita ajamalla. Tähän mennessä ei ole tullut eteen yhtään ylitsepääsemätöntä estettä, ja on odotettavissa, että laite tulee suoriutumaan myös sukellustestauksesta. Alla kuvia laitteesta (kuvat 14 ja 15).



Kuva 14. Laite edestä



Kuva 15. Laite ylhäältä

11.1 Tiiviys

Laitteen tiiviys haluttiin testata ennen varsinaista sukellusta, jotta mahdolliset vuodot eivät aiheuttaisi laiterikkoja. Painerunkojen testaus suoritettiin paineistamalla ne vedellä 5 barin paineeseen. Ensimmäisellä testauksella selvisi, että laippojen tiivistetyyppi ei ollut hyvin tarkoitukseen soveltuva ja se vaihdettiin kumiseen. Tämän lisäksi kiinnityspultteja lisättiin. Näiden muutostöiden jälkeen laipat olivat tiiviitä. Yhdessä hitsaussaumassa oli minimaalinen vuoto, joka vuosi noin pisaran minuutissa. Korjaushitsausten jälkeen tämäkin ongelma saatiin ratkaistua. Moottorikaapelien moottorinpuoleisten päätteiden tiiveys testattiin myös 5 bar paineella onnistuneesti.

11.2 Potkurilaite

Potkurilaitteen ja moottoriohjaimen toiminta testattiin vesiastiassa. Tarkemmat työntövoimamittaukset on vielä tekemättä mutta tuntuma on, että työntövoimaa on riittävästi laitteen liikuttamiseen haluttua muutaman kilometrin tuntinopeutta. Moottoriohjaimen ja ROS-ohjelmiston yhteen liittäminen testattiin myös onnistuneesti.

11.3 Kaikuluotain

Kaikuluotainta testattiin datan keräyksessä. Datan siirto HDMI-USB-muuntimella tietokoneelle toimi luotettavasti, eikä ongelmia useidenkaan tuntien käytön jälkeen ilmennyt. Tähän liittyvä ROS-koodi tuli samalla myös testattua. Kaikuluotainten asetuksille etsittiin optimaaleja asetuksia niin, että kuva olisi mahdollisimman hyvää erilaisissa pohjaolosuhteissa, ja että kohteet tunnistuisivat mahdollisimman hyvin.

11.4 Nopeuden tunnistus kaikuluotainkuvasta

Kaikuluotainkuvasta lasketun nopeuden alustavat tulokset ovat rohkaisevia. Nopeuden virhe GPS-nopeuteen verrattuna on keskimäärin alle 0,2 km/h. Kasvavalla datamäärällä virheen uskotaan vielä pienenevän. Pitää myös huomioida, ettei GPS-nopeuskaan ole tarkka varsinkaan pienissä nopeuksissa. Joissain tilanteissa, joissa mitattu GPS-nopeus on virheellinen, voi laskettu nopeus olla lähempänä totuutta. Miten nämä tulokset korreloivat käytännön tilanteeseen, jossa uudenlaisia pohjan muotoja on lähes rajattomasti, jää nähtäväksi.

12 JATKOKEHITYS

Kehitysmahdollisuuksia on moneen eri suuntaan, niin ohjelmistopuolella kuin rautapuolellakin. Monessa mielessä mielenkiintoisimmat kohteet ovat jäljellä, kun kaikki perusasiat on enimmäkseen saatu toimimaan. Tarkoituksena onkin jatkaa ainakin toistaiseksi laitteen kehittämistä.

12.1 Dopler-nopeus-loki

Maalla nopeustutkat ovat lähes ilmaisia, ja itserakennettuna arduinosta ja anturista pystyy rakentamaan toimivan ratkaisun noin 10 €:n hinnalla. Vedessä käytettävä Doppler-nopeusloki maksaa käytettynäkin tuhansista kymmeniin tuhansiin euroihin, vaikka lopulta tekniikka on monessa mielessä hyvin samanlaista.

Taajuusmielessä ollaan vielä kertaluokkia hitaammissa signaaleissa, joiden käsittely on helpompaa ja halvempaa. Erilaisia piatzo-lähetinvastaanottimia on saatavilla muutamasta eurosta ylöspäin. 1 Mhz:n taajuudella toimivia tehokkaampiakin piatzo-lähetinvastaannottimia saa muutamilla kymmenillä euroilla. Kaupalliset ratkaisut kuten floridalaisten käyttämä, toimii tällä taajuudella. (Alibaba 2019)

Tässä on hyvä jatkotutkimuksen kohde. Onko tässä jotain teknisesti niin vaikeaa, ettei kehitys onnistu pienillä resursseilla, vai eikö kukaan ole koittanut tällaisesta näkökulmasta asiaan tarttua. Kaupallisestikin onnistuessaan tämän kaltaisella huomattavasti halvemmalla laitteella voisi olla markkinat olemassa. Maailmalla on paljon pienellä tutkimusrahalla toimivia instituutioita ja ryhmiä, jotka kamppailevat saman taloudellisen ongelman kanssa. Esimerkiksi läheskään kaikilla Robosub-kilpailuun osallistuvilla joukkueilla ei ole käytössä lainkaan DVL:llää. Jos hinta saadaan painettua korkeintaan muutamaan tuhanteen euroon, uskon että mielenkiintoa laitetta kohtaan löytyy.

12.2 Laajennetun kalman-suotimen korvaaminen neuroverkolla

Kalman-suotimet ovat olleet standardi tapa tehdä sensorifuusioita jo pitkään, ja monessa mielessä ne ovat siinä myös erittäin hyviä. Niiden kyky toimia muuttuvissa olosuhteissa on kuitenkin rajallinen. Kun dataa kertyy laitetesteistä enemmän, voisi kehittää neuroverkkokorvaajaa tälle. Neuroverkolle voi helpommin opettaa esimerkiksi tilanteita, jossa joku tietty anturi ei annakaan oikeaa tietoa tai ei anna tietoa lainkaan.

12.3 Gazebo-simulointi

Gazebo on varmastikin käytetyin simulaatioympäristö ros-järjestelmissä. Joskin monilla tahoilla on myös tehtäväkohtaisia simulaatioita. Floridalaisillakin on käytössä sekä gazebo- että itse kehitetty simulaatioympäristö. Simuloinnin arvoa ei pidä vähätellä, ja Floridalaisten gazebo-simulaatioita ajamalla oppii paljon heidän koodistaan ja ohjelmistoarkkitehtuurista, joka heillä on käytössä. Esimerkiksi drone-puolella simulaatiot ovat olleet niin hyviä, että kun on opetettu vahvistusoppimisen menetelmillä drone lentämään kamerakuvan perusteella simulaatiossa, se on osannut lentää suoraan myös oikeassa maailmassa (Hwangbo 2017). Joillain algoritmeilla kuten A3C-tyyppisillä vahvistusoppimisen algoritmeillä simulointi toimii paremmin myös siitä syystä, että oppiminen on helpompaa ja tasaisempaa kun neuroverkkoa opetetaan usealla kohteella samanaikaisesti. (Juliani 2016).

Myös esimerkiksi korkeamman tason autonomisten tehtävien testaus on huomattavasti helpompaa simulaatioympäristössä. Floridalaisten tekemästä simulaatiosta saadaan hyvä pohja myös tämän projektin mallille. Robotin simuloimiseksi tarvitaan robotin tiedot URDF-mallina. URDF-tiedostot voidaan tuottaa Solidworks 3d-mallinnus -ohjelmalla tehdystä mallista käyttäen apuna tähän tarkoitettua käännös ohjelmaa. (Brawner 2019).

Simulaatiota varten tarvitaan tiedot laitteen vedenvastuksesta, sekä moottorien tuottama työntö eri tehoarvoilla. Tästäkään syystä ei simuloinnin tekeminen ollut ensimmäisten tehtävien joukossa. Koeajodatan perusteella ne ovat kuitenkin laskettavissa.

Haastavinta tulee oletettavasti olemaan kaikuluotainten simulointi. Gazebossa on kyllä Lidar-tyyppisiä antureita mallinnettuna (Selby n.d.), joten tämä varmaan on hyvä lähtökohta.

Työ oli hyvin haastava ja opettavainen. Kokonaisuuden hallinta oli paikoitellen vaikeaa, ja piti tehdä vaikeita valintoja ajankäytön suhteen. Kaikkein eniten pidin neuroverkkojen tutkimisesta, ja oli todella mielenkiintoista nähdä modernien koneoppimismetodien vahvuus käytännön ongelmien ratkaisuissa. Ohjelmisto-osaaminen kasvoi merkittävästi niin neuroverkoista kuin myös ROS-järjestelmästä.

LÄHTEET

Aeberhard M. 2016. BMW automated driving with ros. Viitattu 12.9.2019 https://www.ros.org/news/2016/05/michael-aeberhard-bmw-automated-driving-with-ros-at-bmw.html

Alibaba. 2019. Ultrasonic sensor. Viitattu 14.9.2019 https://www.alibaba.com/product-detail/1MHz-Waterproof-Ultrasonic-Fuel-Level-Sensor_60313250783.html?spm=a2700.7724838.2017115.59.285837ef2q9wxq

Bosch-sensortec. 2019. IMU. Viitattu 6.8.2019 https://www.bosch-sensortec.com/bst/products/all_products/bno055

Brawner S. 2019. Urdf exporter. Viitattu 14.9.2019 http://wiki.ros.org/sw_urdf_exporter

Cameron O. 2016. Udacity steering challenge. Viitattu 27.11.2019 https://medium.com/udacity/teaching-a-machine-to-steer-a-car-d73217f2492c

CHRobotics. Understanding Euler angles. Viitattu 12.9.2019 http://www.chrobotics.com/library/understanding-euler-angles

DeTone, D. 2017. Deepslam, Toward Geometric Deep Slam. Viitattu 7.9.2019. https://arxiv.org/pdf/1707.07410.pdf

Extron. Cat6 cable. Viitattu 12.9.2019 https://www.extron.com/download/files/specs/UTP_CAT_6_cable_020402.pdf

Fischer N., Hughes D., Walters P., Schwartz E. & Dixon W. 2014. Nonlinear RISE-Based Control of an Autonomous Underwater Vehicle. Viitattu14.9.2019 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.661.4091&rep=rep1&type=pdf

Fischer T. Deep reinforcement learning drone control. Viitattu 14.9.2019 https://github.com/tobiasfshr/deep-reinforcement-learning-drone-control

Forster C., Pizzoli M. & Scaramuzza D. SVO: Fast semi-direct monocular visual odometry. 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 15-22. doi: 10.1109/ICRA.2014.6906584

HDMI. 2019. Wikimedia foundation. Viitattu 6.8.2019 https://en.wikipedia.org/wiki/HDMI

Higgins, M. 2019. Steering models. Viitattu 4.9.2019 https://github.com/udacity/self-driving-car/tree/master/steering-models

Hwangbo J. 2017. Control of a Quadrotor with Reinforcement Learning. Viitattu 14.9.2019 https://arxiv.org/abs/1707.05110

Juliani A. 2016. Asynchronous Actor-Critic Agents (A3C). Viitattu 14.9.2019 https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2

Kalman filter. 2019. Wikimedia foundation. Viitattu 7.9.2019 https://en.wikipedia.org/wiki/Kalman_filter

Kalman suodin. 2019. Wikimedia foundation. Viitattu 7.9.2019 https://fi.wikipedia.org/wiki/Kalman-suodin

Lappalainen, T. NMEA2000. Viitattu 6.8.2019 https://github.com/ttlappalainen/NMEA2000

Machine Intelligence Laboratory University of Florida. 2019. Subjugator source code. Viitattu 4.9.2019 https://github.com/uf-mil/SubjuGator

Meanwell. 2019. DPU-3200. Viitattu 12.9.2019 https://www.meanwell.com/Upload/PDF/DPU-3200/DPU-3200-SPEC.PDF

Misra A. Capsule networks. Viitattu 13.9.2019 https://towardsdatascience.com/capsule-networks-the-new-deep-learning-network-bd917e6818e8

Moore, T. 2018. Robot localization. Viitattu 7.9.2019 http://wiki.ros.org/robot_localization

Moore, T. 2015. Robot localization issues. Viitattu 7.9.2019 https://github.com/cra-ros-pkg/robot_localization/issues/235

Moreno V. & Pigazo A. 2009. Kalman filter Recent Advances and Applications. ISBN: 978-953-307-000-1

National Marine Electronics Association. 2009. NMEA2000. Viitattu 12.9.2019 https://www.nmea.org/Assets/20090423%20rtcm%20white%20paper%20nmea%202000.pdf

National Marine Electronics Association. 2002. NMEA0183. Viitattu 12.9.2019 http://www.plaisance-pratique.com/IMG/pdf/NMEA0183-2.pdf

Odabasi C. 2019. Object detection on ROS. Viitattu 14.9.2019 https://github.com/cagbal/ros_people_object_detection_tensorflow

Pasmark. 2019. Prosessori vertailu. Viitattu 12.9.2019 https://www.cpubenchmark.net/high_end_cpus.html

Penttinen, J. 2017. NMEA 2000 retromittari. AMK-opinnäytetyö. Vaasan ammattikorkeakoulu. Viitattu 6.8.2019 https://www.theseus.fi/bitstream/handle/10024/123736/Penttinen_Jukka.pdf?sequence=1&isAllowed=y

Robonation. 2019. Robosub. Viitattu 05.10.2019 https://robonation.org/programs/robosub/

RS-485. 2019. Wikimedia foundation. Viitattu 6.8.2019 https://en.wikipedia.org/wiki/RS-485

Saarivirta M. 2018. Autonomiset sukeltavat robotit pähkinänkuoressa. AMK-opinnäytetyö. Satakunnan ammattikorkeakoulu. Viitattu 17.9.2019 http://urn.fi/URN:NBN:fi:amk-2018110216538

Selby W. Lidar Sensor in ROS Gazebo and RViz. Viitattu 14.9.2019 https://www.wilselby.com/2019/05/simulating-an-ouster-os-1-lidar-sensor-in-ros-gazebo-and-rviz/

Silveira L., Guth F., Drews-Jr P., Ballester P., Machado M., Moraes F., Nelson D. & Botelho S. 2015. An Open-source Bio-inspired Solution to Underwater SLAM. IFAC-PapersOnLine. 48. 10.1016/j.ifacol.2015.06.035.

Suhlman, N. 2019. Subjugator returns. Viitattu 12.9.2019 https://ci.mil.ufl.edu/blog/subjugator/team/2019/01/29/SubjuGator-2019.html

TE connectivity. 2017. MS5803-14BA. Viitattu 05.10.2019 https://www.mouser.fi/datasheet/2/418/NG_DS_MS5803-14BA_B3-1130132.pdf

Telos Systementwicklung. I2c maksimi kapasitanssi. Viitattu 6.8.2019 https://www.i2c-bus.org/i2c-primer/termination-versus-capacitance/

TP-LINK. 2019. Kuitumuunnin. Viitattu 6.8.2019 https://www.tp-link.com/us/business-networking/accessory/mc220l/#specifications

Udacity. 2019. Self driving car engineering nanodegree. Viitattu 4.9.2019 https://www.udacity.com/course/self-driving-car-engineer-nanodegree--nd013

University of Florida. 2017. Subjugator. Viitattu 4.9.2019 http://subjugator.org/

University of Florida. 2016. Subjugator. Viitattu 27.11.2019 http://subjugator.org/?page_id=2661

USB3.0. 2019. Wikimedia foundation. Viitattu 6.8.2019 https://en.wikipedia.org/wiki/USB_3.0

Volya D. Weekly update. Viitattu 12.9.2019 https://ci.mil.ufl.edu/blog/update/software/2018/06/07/software.html

Wightman, R. 2019. Udacity driving reader. Viitattu 4.9.2019 https://github.com/rwightman/udacity-driving-reader