Juha Vainio

Vallitsevien sääolosuhteiden tallennus reittiliikenteessä

Opinnäytetyö

Merenkulun koulutusohjelma

Huhtikuu 2016

KYAMK
University of Applied Sciences

KYAMK
University of Applied Sciences

Tekijä/Tekijät	Tutkinto	Aika
Juha Vainio	Merenkulkualan insinööri	Huhtikuu 2016

Opinnäytetyön nimi

Vallitsevien sääolosuhteiden tallennus reittiliikenteessä

47 sivua
18 liitesivua

Toimeksiantaja

Oy Viking Line Ab

Ohjaaja

Lehtori Ari Helle

Tiivistelmä

Tämän insinöörityön tarkoituksena oli kehittää automaattinen säätiloja rekisteröivä laite yksinkertaistamaan m/s Gabriellan matkaraportin täyttöä. Kerättävät tiedot koskevat ennen kaikkea matkalla vallinneita keskimääräisiä tuulen suunta- ja nopeustietoja. Myös aallokkotietoa oli tarkoitus havainnoida. Se ei yksiselitteisesti ole ainoastaan aallokon korkeustieto, vaan koostuu muistakin optimaaliseen ajonopeuteen vaikuttavista tekijöistä. Saatuja säätietoja käytetään osaltaan mm. ajotekniikan ja polttoainetalouden keskinäisten vaikutus suhteiden arvioimiseen. Tuulen suuntaa ja nopeutta koskevat muuttujat määritettiin primääritavoitteiksi ja aallokkoa koskeva sekundääritavoitteeksi sen huomattavan kompleksisuutensa vuoksi.

Työn laitealustaksi valittiin Arduino-mikrokontrolleri sen edullisuuden, helpon saatavuuden ja runsaan oheismoduulitarjonnan ansiosta. Lisäksi esimerkkiohjelmistoja ja ohjelmakirjastoja tarvittaville lisämoduuleille oli saatavilla niin runsaasti, että usko työn tavoitteiden täyttämiseen oli riittävä.

Työ on tyypiltään kokeellinen tutkimus, jossa rakennettu laite kerää säätietoa itsenäisesti laivan järjestelmän rinnalla ja jatko kehittää automaattisesti analyysin kerätystä tiedosta. Kerättävää tietoa voidaan vertailla laivan järjestelmän antamaan tietoon, mutta analyysin tuloksia voidaan verrata vain teoreettisiin ja koejärjestelyin aikaansaatuihin tietoihin.

Työn primääritavoitteet saavutettiin ja sekundäärisestä saatiin rohkaisevia tuloksia kehitystyön jatkamiseksi.

Asiasanat

energiatehokkuus, automaatio, Arduino, sulautetut järjestelmät

KYAMK
University of Applied Sciences

Author (authors)	Degree	Time
Juha Vainio	Bachelor of Marine Technology	April 2016

Thesis Title

Automated Recording of Weather Conditions in Regular Ferry Service

47 pages
18 pages of appendices

Commissioned by

Oy Viking Line Ab

Supervisor

Ari Helle, Senior Lecturer

Abstract

The purpose of this thesis was to develop an automated device for the registration of weather conditions which were required in the m/s Gabriella´s journey report. The conditions in question were both the mean wind speed and direction. Also, a wave analysis was to be created including not only the wave height but also other factors influencing the optimal driving speed as well. Weather information data will be used for example in the process of evaluating the dependencies between the steering techniques and fuel economy. The mean wind speed and direction were considered primary objectives and due to its complexity, the wave analysis a secondary objective.

As a basis for the study, an Arduino microcontroller was chosen due to its excellent affordability, availability and broad selection of peripheral device modules. Also, the number of example codes and code libraries was so plentiful that it was assumed that the study would be able to meet the given objectives.

This thesis is an experimental study where the device being built for the study collects weather data independently alongside the system of the ship and develops an automated analysis of the data. This data can be compared with the system data of the ship, but the results of the analysis can be compared only with theoretical results and the data gathered in an experimental set-up situation.

As a result, the primary objectives were achieved and some promising data was gathered for the secondary objective to support the continuation of the development process.

Keywords
energy efficiency, automation, Arduino, embedded systems

SISÄLLYS

1 JOHDANTO 8

2 INTERNATIONAL MARITIME ORGANIZATION IMO 10

2.1 Ship Energy Efficiency Management Plan (SEEMP) 10

2.2 SEEMP:n tarkoitus 10

2.3 SEEMP-prosessin osat 11

2.3.1 Suunnittelu 11

2.3.2 Käyttöönotto 12

2.3.3 Seuranta 12

2.3.4 Arviointi ja kehitys 12

2.3.5 Tulosten julkistaminen ja raportointi 12

2.3.6 Käytännön metodit 13

2.4 Apuohjelmia SEEMP kehitystyön tueksi 14

2.4.1 EMMA ™ Advisory Suite 14

2.4.2 Sensorit 17

2.4.3 Sea State 20

3 AUTOMAATTINEN SÄÄOLOSUHTEIDEN TALLENNUS 22

3.1 Arduino 22

3.2 Komponenttien valinta 23

3.2.1 Suunniteltu laitekokoonpano. 23

3.2.2 Laitealusta Arduino Uno R3 24

3.2.3 Tuulen suunta- ja nopeusanturit 25

3.2.4 Arduino ethernet-shield 26

3.2.5 Kiihtyvyysanturi 27

3.2.6 GPS - anturi 27

3.2.7 LCD - näyttö 28

3.3 Moduulien käyttämät protokollat 30

3.3.1 USB - sarjaliikenneprotokolla 30

3.3.2 I2C - protokolla 30
3.3.3 SPI - protokolla 31
3.3.4 Laitteen ohjelmointi 31
3.4 Komponenttien konseptitestaus 32
3.4.1 LCD - näyttö 32
3.4.2 Gyro / kiihtyvyysanturi 32
3.4.3 GPS - moduuli 33
3.4.4 Ethernet / mikro SD / Web-serveri 33
3.5 Lopullinen ohjelman toiminta lyhyesti 34
3.5.1 Tuulen suunta 34
3.5.2 Tuulen nopeuden mittaus 35
3.5.3 True arvojen laskeminen 36
3.6 Lopullinen testaus 37
3.6.1 Anemometri ja tuuli-indeksi 37
3.6.2 True - arvot 37
3.6.3 Gyro - moduuli 38
4 JOHTOPÄÄTÖKSET 40
5 JATKOKEHITYS 41
5.1.1 Tuulen suunta ja voimakkuus 41
5.1.2 MPU-6050 kiihtyvyysanturi ja Sea State 43
LÄHTEET 44
LIITTEET
Liite 1. True-arvojen mittapöytäkirjat
Liite 2. Käyttöohje ruotsiksi
Liite 3. Ohjelmakoodi

LYHENTEET

CS	Chip Select
CSV	Comma Separated Values
EEIO	Energy Efficiency Operational Indicator
EMS	Emergency Schedules
GPS	Global Positioning System
HiD	Human interface Device
IDE	Integrated Development Environment
IEEC	International Energy Efficiency Certificate
IO	Input Output
IMO	International Maritime Organization
I2C / IIC	Inter-Integrated Circuit, sarjaliikenneväylä
LCD	Liquid Crystal Display
MARPOL	IMO:n ympäristönsuojelua koskeva sopimus
MEPC	IMO: Marine Environment Protection Committee
MEMS	Micro-Electro-Mechanical Systems
MISO	Master Input Slave Output (SPI)
MOSI	Master Output Slave Input (SPI)
NMEA	National Marine Electronics Association
SCK	Serial Clock (SPI)
SCL	Serial Clock (I2C / IIC)
SDA	Serial Data (I2C / IIC)
SEEMP	Ship Energy Efficiency Management Plan
SPI	Serial Peripheral Interface, sarjaliikenneväylä
SS	Slave Select (SPI)

STP	Shielded twisted pair
USB	Universal Serial Bus, sarjaliikenneväylä

1 JOHDANTO

Koko 2000 luvun ovat energiakysymykset merenkulussa olleet merkittävänä puheenaiheena ja teknisen kehityksen ohjaajana pyrittäessä kehittämään meriliikennettä kustannustehokkaaseen suuntaan. Vuosikymmenen vaihteessa etenkin alusten ympäristövaikutuksiin perustuvat kiristykset päästörajoituksiin sekä hintalapun muodostaminen mm. CO2 päästöille, saivat kansainvälisen merenkulkujärjestön IMO:n kehittämään ohjeistuksen alusten energiatehokkuuden seurantaan ja kehittämiseen. Vuoden 2013 alusta tämä Ship Energy Efficiency Management Plan tuli pakolliseksi kaikille aluksille. Suunnitelma luo yhtenevät puitteet menetelmille ja toiminnoille, joilla varustamot voivat tehostaa omia toimintojaan sekä optimoida ja kehittää alustensa käyttöä ja tekniikkaa ympäristön ja kestävän kehityksen vaatimukset huomioiden. Suunnitelma ei ota kantaa menetelmiin, joilla tietoa kerätään, eikä odota saatujen tulosten aina kehittyvän odotettuun suuntaan. Suunnitelman tarkoituksena ja fokuksena on selvittää aluksen kulloinenkin energiatase ja että toimintoja pyritään tulosten pohjalta kehittämään. Sanktioitua on vain tämän seurannan ja kehitystyön laiminlyönti.

Uusissa aluksissa, joissa automaatio paljolti ohjaa ja säätää toimintoja, on myös raportoinnin automatisointi kehittynyt jopa reaaliaikaiseksi. Vanhemmissa laivoissa tietoa rekisteröidään paljon vielä käsin, kynän ja paperin välityksellä.

Tämän insinöörityön ensisijaisena tarkoituksena oli kehittää automaattinen säätiloja rekisteröivä laite m/s Gabriellan matkaraportin kyseisiä tietoja koskevaan osioon. Kerättävät tiedot koskivat ennen kaikkea matkalla vallinneita keskimääräisiä tuulen suunta- ja nopeustietoja. Myös aallokkotietoa oli tarkoitus havainnoida. Se ei yksiselitteisesti ole ainoastaan aallokon korkeus, vaan koostuu muistakin ajonopeuteen vaikuttavista tekijöistä. Saatuja säätietoja käytetään osaltaan mm. ajotekniikan ja polttoainetalouden keskinäisten vaikutus-suhteiden arvioimiseen. Tuulen suuntaa ja nopeutta koskevat muuttujat määritettiin primääritavoitteiksi ja aallokkoa koskeva sekundääritavoitteeksi sen huomattavan kompleksisuutensa vuoksi.

Työn laitealustaksi valittiin Arduino-mikrokontrolleri sen edullisuuden, helpon saatavuuden ja runsaan oheismoduulitarjonnan vuoksi. Lisäksi esimerkkioh-

jelmistoja ja ohjelmakirjastoja tarvittaville lisämoduuleille oli saatavilla niin runsaasti, että usko työn tavoitteiden täyttämiselle oli riittävä.

Työn alussa perehdytään kansainvälisiin sopimuksiin, joihin varustamojen ja sen alusten energia tasetta seuraavat järjestelmät pohjautuvat. Seuraavaksi työssä tutustutaan uuden laivan automaatiojärjestelmään, joka toteuttaa sopimusten mukaista tiedon keruuta automaattisesti sekä osallistuu aktiivisesti käyttöparametrien optimointiin. Seuraavaksi tutustutaan muutamiin tämän automaatiojärjestelmän vaatimiin anturityyppeihin, niiden toimintaperiaatteisiin, sekä automaatiossa esiintyvien sulautettujen järjestelmien tiedonsiirtotapoihin. Loppuosassa työtä selvitetään rakennetun laitteen ohjelman toiminnallista osaa, komponenttitestauksen tuloksia sekä lopullisen tuotantolaitteen toimintaa ja tuloksia. Myös jatkokehitystä pohditaan saatujen tulosten pohjalta. Lisäksi arvioidaan harraste/opetuskäyttöön tarkoitettujen mikrokontrollereiden käyttökelpoisuutta projektityyppisissä kokeiluissa.

Lähdemateriaalina on käytetty kansainvälisiä sopimuksia, järjestelmävalmistajien esityksiä ja esitteitä. Myös aikakausjulkaisuja, web-julkaisuja ja muita opinnäytetöitä on käytetty teoriaosuuden tukena. Ohjelmointi osuuden tietolähteinä on käytetty alan kirjallisuutta, alan keskusteluforumeita ja omaa ammattitaitoa tietojen käsittelyssä ja ohjelmoinnissa.

2 INTERNATIONAL MARITIME ORGANIZATION IMO

Kansainvälinen merenkulkujärjestö International Maritime Organization eli lyhyesti IMO, on kansainvälinen, Yhdistyneiden Kansakuntien (YK) alainen jäsenvaltioidensa muodostama järjestö, jonka tehtävänä on luoda kattava säännöstö turvalliselle ja turvatulle, sekä luontoa ja työskentely-ympäristöä suojelevalle merenkulkualan toimintakulttuurille. On sovittu, että säännöstön tulee olla toimintaa tehostava, tasapuolinen kaikille sekä jäsenvaltioittensa hyväksymä, että käyttöönottama. (IMO 2016a.)

2.1 Ship Energy Efficiency Management Plan (SEEMP)

IMO julkisti 2009 ohjeen laivojen energiatehokkuutta seuraavan ja kehittävän työkalun Ship Energy Efficiency Management Plan (SEEMP) kehittämisestä (IMO 2011).

Vuonna 2012 työkalu oli vielä vapaaehtoinen, mutta tuli pakolliseksi jo vuoden 2013 tammikuun 1. päivänä osana MARPOL Annex VI:tta, uutena kappaleena: “Chapter 4, Regulations on energy efficiency of ships” ja toimii osana International Energy Efficiency Certificate (IEEC) sertifikaattia. Vuoden 2013 alusta lähtien tuli SEEMP löytyä kaikilta yli 400GT aluksilta kansainvälisessä liikenteessä, sekä uusilta että vanhoilta. (IMO 2012.)

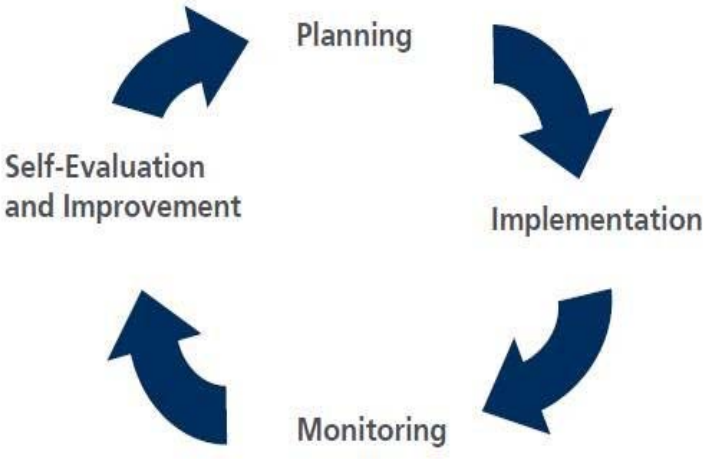
2.2 SEEMP:n tarkoitus

SEEMP-suunnitelma tulee olla osa varustamon toimintakulttuuria ja jokaiselle varustamon alukselle tulee laatia oma suunnitelmansa (IMO 2011).

SEEMP-suunnitelma tulisi nähdä työkaluna, jolla pyritään tehostamaan ei pelkästään aluksen polttoainetaloutta, vaan energia- ja käyttötehokkuutta kokonaisuutena. Näin toimien pyritään vaikuttamaan vähentäen aluksen luomaan ympäristökuormitukseen. SEEMP on aktiivinen dokumentti, jossa toiminnan suunnittelu, käyttöönotto, seuranta ja tulosten analysointi sekä jatkokehitystoimen seuraavat toisiaan keskeytymättä. SEEMP-suunnitelma voi muodostaa osan aluksen turvallisuusjohtamisjärjestelmää Safety Management System SMS. (About Shipping 2012.)

2.3 SEEMP-prosessin osat

Prosessi koostuu neljästä pakollisesta ja yhdestä vapaaehtoisesta osasta, joita ovat suunnittelu, käyttöönotto, seuranta, arviointi ja kehitys, sekä vapaaehtoisena osana tulosten julkistaminen ja raportointi. Kuva 1 kuvaa prosessin jatkuvaa itseään toistavaa kehityskulkua. (About Shipping 2012.)



Kuva 1 SEEMP prosessiin vaiheet (About Shipping 2012.)

2.3.1 Suunnittelu

Lähtökohtana toiminnan kehittämiselle on käydä läpi olemassa olevat käytännöt ja arvioida niiden toimivuus ja tehokkuus sekä energian, että resurssien järkevän käytön kannalta. Käytäntöjen arviointi jakaantuu osiin aluksittain, varustamon osalta, henkilöresurssien osalta ja tavoitteiden asetuksissa.

- Aluskohtaisia ovat mm. ajonopeuden optimointi, sääolojen huomioon otto reittisuunnittelussa, rungon huolto, koneisto-operaatioiden arviointi
- Varustamokohtaisia ovat informaation kulku laivojen ja satamien kanssa, lastauksen ja rahtauksen suunnittelu ja aikataulutus kokonaistoiminnan tehostamiseksi jne.
- Henkilöresurssien osalta henkilöstön tietoisuuden ja koulutuksen kehitys toiminnan kehittämiseksi, sekä toimet asetettujen tavoitteiden saavuttamiseksi
- Tavoitteiden asetus on vapaaehtoista, mutta toimii kannustimena järjestelmän jatkuvaan kehittämiseen

(About Shipping 2012.)

2.3.2 Käyttöönotto

Suunnittelun tuloksena kehitetään menetelmiä suunniteltujen tehtävien suorittamiseksi ja havaittujen puutteiden korjaamiseksi sekä nimetään vastuutahot kullekin toimelle. Tehtäväkuvaukset ja suoritetut toimenpiteet tulee dokumentoida kirjallisesti. (About Shipping 2012.)

2.3.3 Seuranta

Toiminnan tulosten seurantaa on vaikea arvioida muutoin kuin kvantitatiivisilla eli määrällisillä menetelmillä ja se tulisikin tehdä hyväksi havaituilla, mieluimmin kansainvälisiä standardeja käyttävillä mittareilla (About Shipping 2012).

IMO:n ohje MEPC.1/Circ.683 suosittaa käyttämään seurantaan sen asettamaa toista työkalua: Energy Efficiency Operation Indicator (EEIO). Siinä luodaan vertailuluvut energiatehokkuudesta käyttämällä CO2 päästöjen määrää verrattuna lastitonniin ja kuljettuun merimailiin (g CO2 / tn.nm). Tarkemmin käyttöä ja laskentatapaa selvitetään IMO dokumentissa MEPC.1/Circ.684. (IMO 2015b.)

2.3.4 Arviointi ja kehitys

Viimeisenä pakollisena vaiheena kierrossa on arviointimenetelmien kehittäminen ja saatujen tulosten taltiointi seuraavaa kehityskierrosta varten. Tarkoituksena ei ole vain saada selville, missä määrin toiminnan tehostaminen on vaikuttanut suotuisasti energiataseeseen vaan myös kertoa, kuinka tarkoituksenmukainen luotu seuranta- ja kehitysjärjestelmä on ja kuinka sitä voitaisiin edelleen kehittää. Kukin menetelmä tulee arvioida itsessään säännöllisesti ja saatuja tuloksia käyttää kunkin aluksen kehitystä arvioitaessa. (About Shipping 2012.)

2.3.5 Tulosten julkistaminen ja raportointi

Vaikkakin vapaaehtoista, voi tulosten julkistaminen olla eduksi varustamolle julkisuuskuvan ekologisuuden luomisessa ja käytännössä näkyä helpotuksina mm. satamamaksuissa. Myös ekologisuuttaan korostava kulutustavarateollisuus nähnee potentiaalia ”vihreiden” arvojensa esiintuonnissa koskien koko tuotantoketjuaan. (About Shipping 2012.)

2.3.6 Käytännön metodit

Eräs keskeisimpiä metodeja on polttoaineen kulutuksen optimointi mm. reittisuunnitelulla ja ajonopeuden säädöllä otettaessa muuttuvat sääolosuhteet huomioon. Myös satamakäyntien aikataulutus tarpeettomien odotusten välttämiseksi tulee huomioida. Tuotettua tehoa tarkkaillaan sekä akselitehona että laivan kokonaistehonkulutuksena, jossa pyritään optimoimaan myös hukkalämmön käyttö mahdollisuuksien mukaan. Trimmin vaikutuksia ja ballastin määrää seurataan kulutetun akselitehon ja saavutetun ajonopeuden suhteen. Pohja- ja potkuripintoihin kasvava merieliöstö lisää suuresti kitkaa ja aiheuttaa tehohäviöitä. Koska työsuunnitelma on aina laivakohtainen, tulee se tehdä yksilöllisesti kohta kohdalta. IMO antaa asiassa viitteellisen esimerkin siitä, kuinka lista voitaisiin kuvan 2 mukaisesti kohta kohdalta käydä läpi. Kuvan 2 taulukossa otsaketietojen jälkeen on vasemmalla aiheluokka, sitten suunniteltu toimenpide ja oikeassa laidassa nimetty vastuuhenkilö sekä lisätietoja. (IMO 2012.)

SHIP EFFICIENCY ENERGY MANAGEMENT PLAN

Name of Vessel:		GT:	
Vessel Type:		Capacity:	

Date of Development:		Developed by:	
Implementation Period:	From: Until:	Implemented by:	
Planned Date of Next Evaluation:			

1 MEASURES

Energy Efficiency Measures	**Implementation** (including the starting date)	**Responsible Personnel**
Weather Routeing	<Example> Contracted with [Service providers] to use their weather routeing system and start using on-trial basis as of 1 July 2012.	<Example> The master is responsible for selecting the optimum route based on the information provided by [Service providers].
Speed Optimization	While the design speed (85% MCR) is 19.0 kt, the maximum speed is set at 17.0 kt as of 1 July 2012.	The master is responsible for keeping the ship's speed. The log-book entry should be checked every day.

Kuva 2 IMO:n SEEMP ohjeen esimerkki (IMO 2012)

2.4 Apuohjelmia SEEMP kehitystyön tueksi

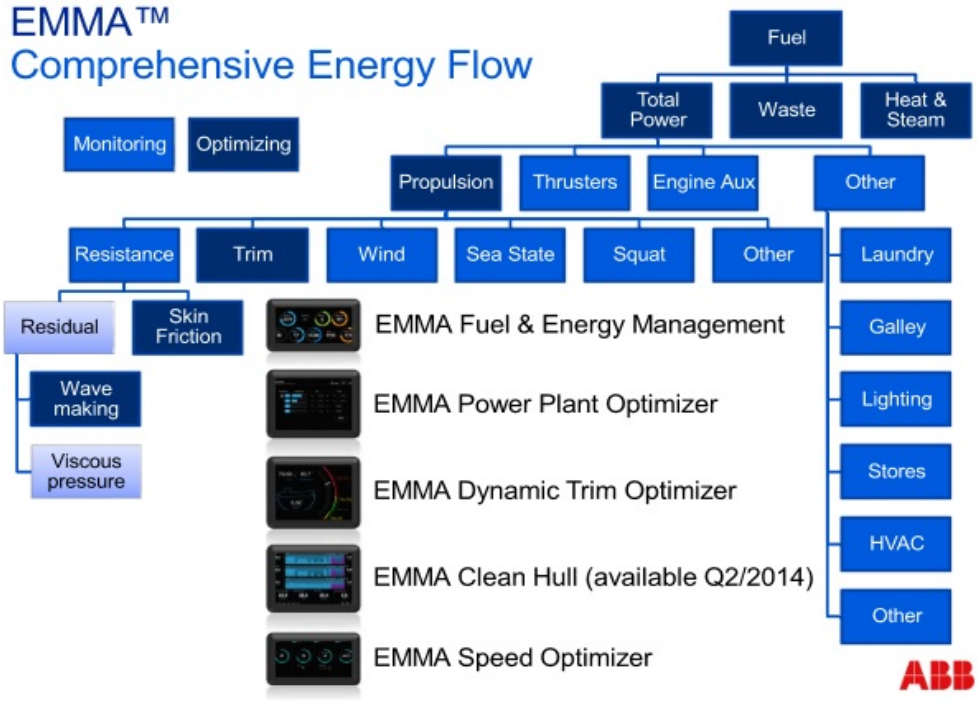
Laiva-automaation kehitys on ollut suurta 1960 luvulta lähtien ja tiedonsiirtomäärät ja nopeudet ovat kasvaneet kiloista gigabitteihin. Tietomäärien kasvaessa on tallennustarve alati kasvanut, tarvittavat käyttöliittymät parantuneet ja langatonta etähallintaakin on kehitetty. (Lehto 2014.) Kun automaatio auttaa hienosäätämään ajo parametreja ja alusten energiatasetta kokonaisuutena, on se muassaan tuonut myös automaattisten raportointijärjestelmien tarjonnan helpottamaan tiedon hankintaa ja analysointia. Järjestelmiä on lukuisia eri osa-alueille ja myös energiataseen seurantaan on kehitetty omat työkalunsa, jotka toiminnoiltaan seurailevat IMO:n SEEMP:ssä asettamia vaatimuksia. Kotimaista tarjontaa edustavat esim. Eniram Oy:n Eniram Platform (Eniram 2016) tai ABB:n Emma Advisory Suite (Ignatius. 2014). Koska työn tilaajan uusin laiva m/s Grace hyödyntää ABB Emma Advisory Suitea, katsotaan sen toimintoja hiukan läpi (Ignatius 2012).

2.4.1 EMMA ™ Advisory Suite

Emma™ Advisory Suite on ABB:n kehittämä energianhallinta- ja seurantajärjestelmä ja toimii aktiivisena osana optimaalisten ajoparametrien etsimisessä ja kohdalleen säätämisessä. Ohjelmisto on modulaarinen, ts. asiakas voi halutessaan ottaa käyttöön vain perusosat järjestelmästä ja laajentaa toimintoja myöhemmin paremmin tarpeita kattaviksi. Keskeisimpiä perusosia ovat Fuel & Energy Management, Powerplant Optimizer ja Dynamic Trim Optimizer. Järjestelmä on lisäksi laajennettavissa vielä Clean Hull ja Speed Optimizer moduuleilla. (Ignatius 2014.)

Kuvassa kolme ovat keskellä kaikki Emma™ Advisory Suite-ohjelmistokomponentit. Laatikoissa ovat aihealuettain vaalean sinisellä kohteet joita voidaan monitoroida, sekä tumman sinisellä kohteet joiden toimintaa voidaan järjestelmässä optimida automaattisesti. (Ignatius 2014.)

Koska ABB toimitti m/s Gracen sähköpotkurikäyttöjärjestelmän generaattoreineen, pääsähkökojeistoineen sekä taajuusmuuttajineen sekä komponentteja ankkuri- ja kiinnitysvinssisovelluksiin taajuusmuuttajineen, tuli EMMA ™ Advisory Suitesta aktiivinen osa kokonaisuuden hallintaa (ABB 2014 b.)



Kuva 3 Emma Suite moduulit (Ignatius 2014)

Ohjelmisto toimii tärkeänä työkaluna aluksen energiansäästöohjelmassa, se mm. kertoo koko aluksen energiankulutuksen. Saatu tieto tallennetaan ja analysoidaan optimaalisimman energian käyttötavan löytämiseksi. Myös tuulen, merenkäynnin, trimmin ja syväyksen vaikutusta energiankulutukseen seurataan tarkasti. (ABB 2014 b.)

Kuva 4 m/s Grace:n EMMA Onboard Tracker näyttö konevalvomossa

Kuvassa 4 on konehuoneessa sijaitsevan EMMA Onboard Trackerin näyttö, jossa keskeisimpiä ajo parametreja voidaan seurata tosiaikaisesti.

ABB:n kokonaistoimitus mahdollisti EMMA ™ Advisory Suiten integroinnin laivan järjestelmiin niin keskeisesti, että kaikkien järjestelmien tietoa voidaan tehokkaasti kerätä ja analysoida reaaliaikaisesti, mikä auttaa seuraamaan tehtyjen säätötoimien vaikutusten arviointia ja muutosta välittömästi. (Rosenqvist 2016.)

Kuva 5 m/s Grace:n EMMA Dynamic Trim Optimizer näyttö sillalla.

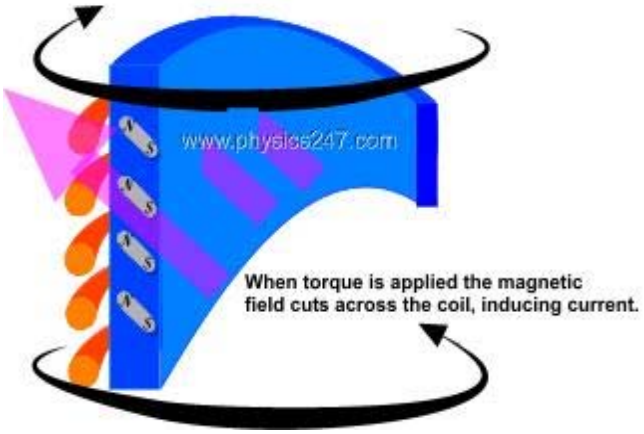
Järjestelmä on myös itseoppiva, jo kuukauden käytön jälkeen alkoi EMMA ™ Advisory Suite antaa korjausehdotuksia mm. trimmin suhteen, jotka osoittautuivatkin taloudellisesti merkittäviksi (Anderson 2015). Kuva 5 kuvastaa hyvin selkeästi optimaalisen trimmin vihreänä alueena aluksen keulan puoleiselle asteikolle kuvattuna.

Järjestelmätoimitukseen kuuluu lukuisia antureita, joista keskeisimpien toimintaperiaatteita on syytä käydä hiukan läpi (ABB 2016a).

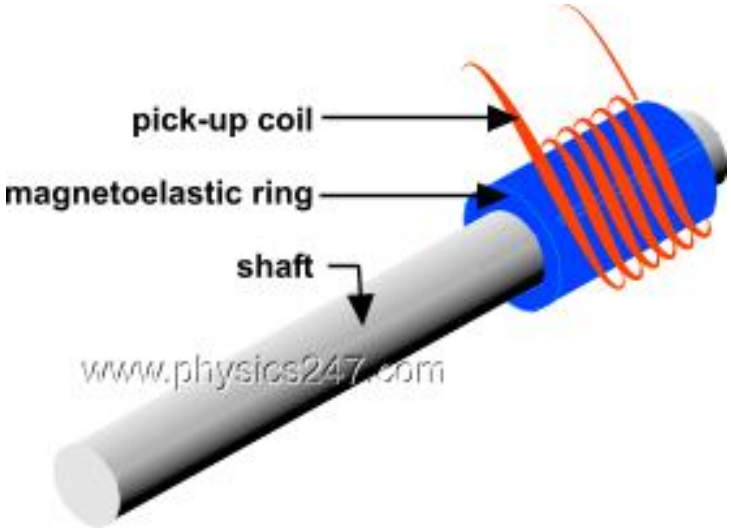
2.4.2 Sensorit

2.4.2.1 Torductor - vääntömomenttimittari

Vääntömomenttimittari aluksen potkuriakselissa kertoo suoraan potkuriin välitettävän propulsiotehon suuruuden. Mittaustapoja on erilaisia, m/s Gracessa on magnetoelastiset sensorit, jotka käyttävät hyväkseen Villary-ilmiötä (ABB 2016b). Kuva 6 kuvaa, kuinka elastisen massan magneettiset ominaisuudet muuttuvat massaa venytettäessä, puristettaessa, kierrettäessä tai taivutettaessa. Magneettikentän suunnan- ja voimakkuuden muutos massassa kuvassa 7 indusoituu sensorin sisältämään kelaan sähköjännitteeksi ja on siten muunnettavissa numeeriseen muotoon. (Physics 24/7 2016.)



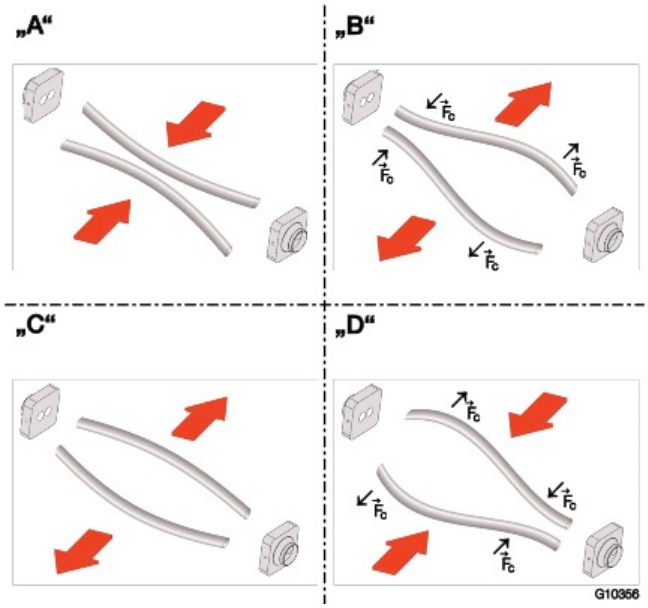
Kuva 7 Villary-ilmiöön perustuvan anturin toiminta (Physics 24/7 2016).



Kuva 6 Akselilla sijaitseva anturi keloineen (Physics 24/7 2016).

2.4.2.2 CoriolisMaster eli massavirtamittari

CoriolisMaster massavirtamittari toimii Coriolis-ilmiöön perustuen. Mittarissa yksi tai kaksi U-mutkalle taivutettua putkea saatetaan harmoniseen värähtelytilaan U mutkan keskeltä. Kuvan 8 osa A ja C kuvaavat, kuinka putkien taipuma on symmetrinen A<->C, kun ainetta ei virtaa putkissa.#Aineen virratessa putkien läpi, aiheuttaa aineen massa viiveen putkien värähtelyliikkeeseen U-mutkan alku- ja loppupään välille eli värähtely on epäsymmetristä osissa B<->D. Tämän epäsymmetrisyys mittaamalla saadaan selville vaihe-ero alku- ja loppupään värähtelyssä, amplitudimuutos sekä taajuuden muutos. Näin anturitiedoilla saadaan ratkaistua aineen massavirta, tilavuusvirta, tiheys, lämpötila sekä konsentraatio. (ABB 2014.)

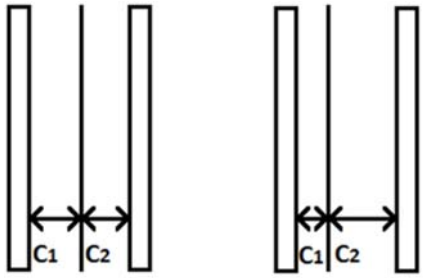


Kuva 8 Coriolis massavirtamittarin toimintaperiaate vaiheittain (ABB 2014).

Käytännössä on havaittu, että jo pelkän massavirtamittarin jälkiasennus auttaa parantamaan aluksen energiankulutuksen seurantaa ja kehitystä suuresti, kun kulutusmuutokset polttoainevirrassa saadaan selville reaaliaikaisesti. Näin tehdyt säätötoimenpiteet antavat heti tarkkaa palautetta muutosten vaikutuksista. Mm. XPRS:lle asennettiin Emerson Micro Motion - massavirtausmittauslaitteistot, joiden avulla oli mahdollista saada kulutustietoa laivan pääkoneiden, apukoneiden sekä höyrykattiloiden kuluttamasta polttoaineen määrästä reaaliaikaisesti. (Vuorinen 2014.)

2.4.2.3 Attitude sensor eli asentosensori

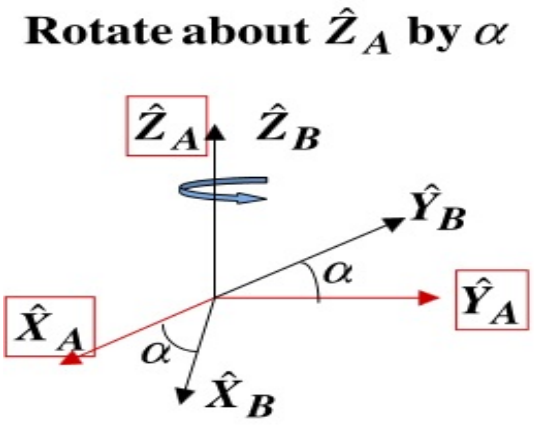
Asentosensoreita tarvitaan dynaamisen trimmin mittaamiseen ja aluksen koosta riippuen niitä tarvitaan kahdesta kolmeen kappaletta (ABB 2016a). Rakennetekniikoita antureille on useita erilaisia, kuten pietsosähköiset ja kapasitiiviset anturit (Pursiainen 2011).



Kuva 9 Kapasitiivisen MEMS tekniikalla tehdyn anturin periaate (Pursiainen 2011).

Tässä työssä käytetty moduuli on ns. MEMS tyyppinen, eli Micro-Electro-Mechanical Systems mikrovalmistustekniikka. MEMS tekniikalla on mahdollista valmistaa pienoiskoossa olevia sekä mekaanisia että sähkömekaanisia elementtejä, kuten moottoreita, aktuaattoreita ja kiihtyvyysantureita. (MNX 2016.)

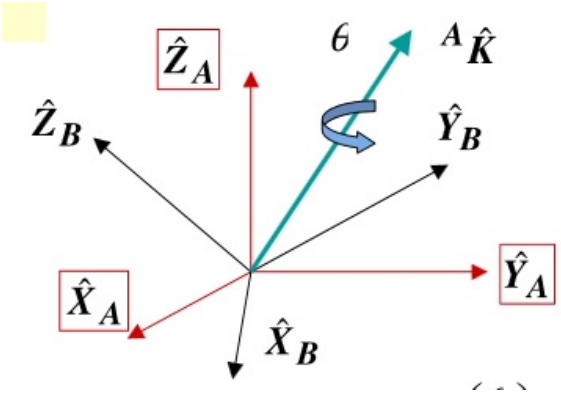
Kuvassa 9 esimerkiksi ovat kapasitiiviset levyt, joiden välinen kapasitanssi muuttuu kahden levyn välissä joustavasti kiinnitetyn levyn asennon muuttuessa maan vetovoiman tai muun kiihtyvyysvoiman vaikutuksesta (Pursiainen 2011).



Kuva 10 Eukleen Z-akselin kiertokulma kuvattuna X ja Y-akselin kulmamuutoksina. (de Weck 2001).

Eukleen kulmat kuvaavat kiertoa yhden akselin suhteen kahden muun akselin kiertokulmia tarkastelemalla, kuten kuvassa 10, jossa Z akselin kierto kuvataan X ja Y akselien alpha kulmamuutoksilla (de Weck 2001).

Kaikista kolmesta akselista saaduista suuntavektoreista saadaan neljäs summavektori ja kiertoa kuvaava skalaarifunktio yhteisnimeltään kvaternion, kuten kuvassa 11 oleva sininen summavektori kiertymineen näyttää (de Weck 2001).



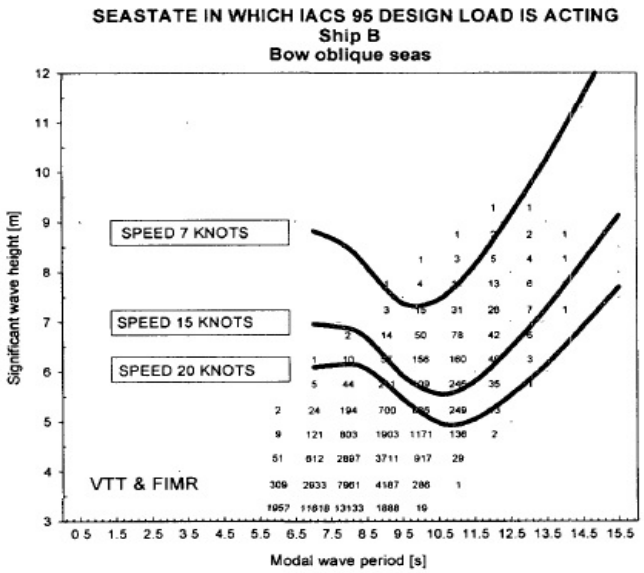
Kuva 11 Kvaternion kuvaa yhdellä suuntavektorilla ja kulmanopeutiedolla kaikkien kolmen eukleen vektorin tilaa (de Weck 2001).

Kvaternion on siis kierron kuvaus, kun tunnetaan kiertokulma ja kiertoakselin määräävä yksikkö vektori (Polvinen 2012).

Asentomuutosten automaattinen laskeminen tietokoneilla nopeutuu ja helpottuu suuresti kvaternionien avulla (de Weck 2001).

2.4.3 Sea State

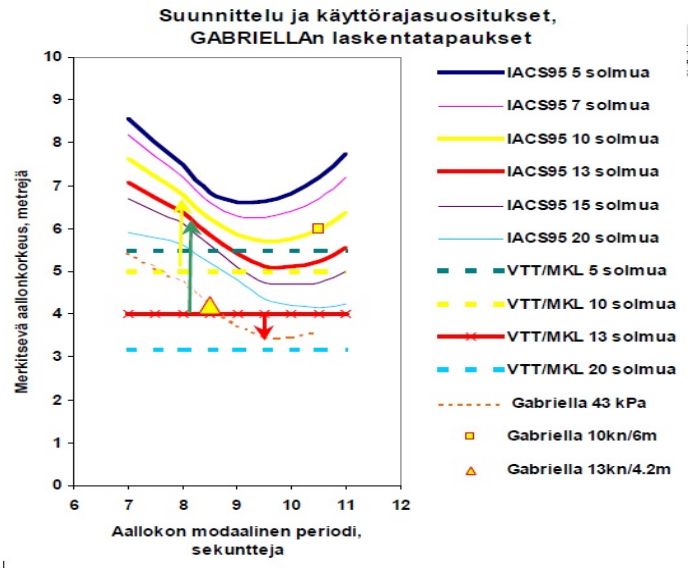
Asentomuutoksia ja kiihtyvyyksiä tarvitaan ajonopeuden optimointiin ja auttamaan operoimaan alusta sen suunniteltujen toimintaparametrien puitteissa. Uusille laivoille on SEEMPiin lisätty muuttuja Sea State, jota voidaan tarkastella ei vain aallokon korkeutena, vaan myös ns. modaalisena, ts. hallitsevana taajuutena eli kuinka usein aallon huiput seuraavat toisiaan. Tällä tiedolla on suuri merkitys keularakenteiden kestävyysvaatimuksiin. Kuva 12 osoittaa aallon korkeuden, aaltojen huippujen tiheyden ja aluksen nopeuden funktiona sallitut operointirajat keskikokoisille aluksille Itämeren liikenteessä, jotka rakenteiden kestävyys eri olosuhteissa sallii. Arvot ovat laivakohtaisia koska aluksen pituudella on hyvin suuri vaikutus. (Soininen, Karppinen, Kahma, Kukkanen, Rintala 2001.)



Kuva 12 IACS 95 mukaiset sallitut maksimirasitukset keskikokoiselle alukselle (Soininen, Karppinen, Kahma, Kukkanen, Rintala 2001)

Taulukko on laadittu Itämeren olosuhteisiin, jossa aallokon hallitseva (modaalinen) periodi on tyypillisesti 4 - 6 sekunnin luokkaa, mutta voi olla jopa 12 - 13 sekuntia. Syvässä vedessä vastaavilla periodiarvoilla on aallonpituudet huipusta huippuun noin 25 - 60 metriä, pisimpien periodien vastaten 225 - 260 metrin aallonpituuksia. (Ilmatieteen laitos 2016.)

Jos sitten tarkastellaan m/s Gabriellalle keulaportin vaurion yhteydessä laskettua taulukkoa kuvassa 13, voidaan havaita 9 - 11 sekunnin periodin olevan suurimman ajonopeuteen vaikuttavan tekijän merkitsevän aallonkorkeuden suhteen (Onnettomuustutkintakeskus 1998).



Kuva 13 VTT:n laskemat maksimikuormitusrajat m/s Gabriella:lle aallon korkeuden ja periodin suhteen (Onnettomuustutkimuskeskus 1998).

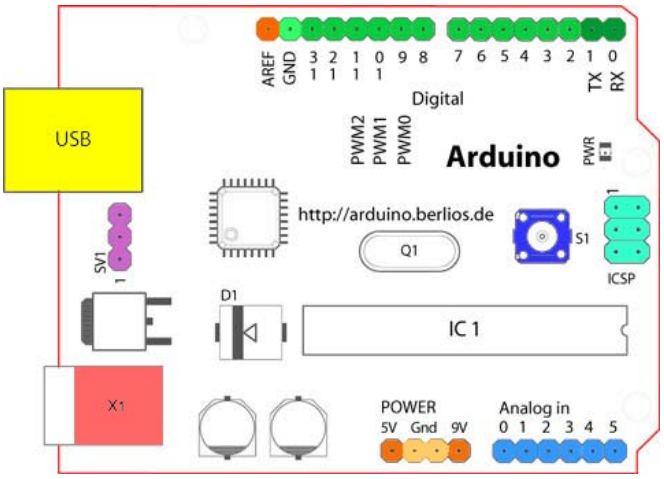
Aluksen Lpp:n ollessa 153 metriä, voitaneen arvella ”hankalan” aallon pituuden olevan n. 90 – 120 metriä kyseisellä periodilla. Aallokon periodinen vaihtelu vaikuttaa siten suuresti aluksen keulaan kohdistuvaan rasitukseen aallon korkeuden ja ajonopeuden lisäksi ja on näin ollen yksi tärkeä muuttuja seurattavaksi.

Tutkintaselostuksen mukaan aluksella liikennöintiin vaurion tapahtuma-aikaan olosuhteiden vaatimalla huolellisuudella. Keulaportin vaurio johtui portin piirustusten vastaisesta rakenteesta, joka sittemmin korjattiin. Sisaraluksissa ei vastaavaa virhettä keulaportin rakenteessa ole havaittu. (Onnettomuustutkintakeskus 1998).

3 AUTOMAATTINEN SÄÄOLOSUHTEIDEN TALLENNUS.

3.1 Arduino

Arduino on sekä avoimeen laitteistoon että avoimeen lähdekoodiin perustuva mikrokontrolleri, jossa ytimenä on 8-bittistä arkkitehtuuria käyttävä Atmel AVR -mikro-ohjain. Laitteistoa ohjelmoidaan C++:aan perustuvalla korkean tason ohjelmointikielellä. Arduino on lähtöisin Ivrea Interaction Design Instituutista Italiasta, jossa sitä käytettiin opetustarkoituksissa erilaisissa prototyyppiprojekteissa. Opiskelijoille, jotka eivät entuudestaan tunteneet kyseistä alaa, opetetaan näin ohjelmointia ja elektroniikkaa. Laitteen suosio kasvoi nopeasti ja avoimuudestaan johtuen on sekä rauta- että softapuolella runsaasti valikoimaa omien projektien kehittelyn pohjaksi. Laitealustan suosio perustuukin sen edullisuuteen, yhteensopivuuteen kehitys-ympäristönsä suhteen (Windows, Linux, Macintosh OSX) ja helppoon käyttöönottoon perustuen Arduinon C++:aan perustuvan korkeantason kielen käyttöön ja monipuolisiin ohjelmakirjastoihin. Taitojen karttuessa, on mahdollista liittää rautatason AVR-C-ohjelmakoodia osittain tai kokonaan tai ohjelmoida vaikka omia kirjastoja laitealustalle. Laitealustan kytkentä on julkaistu Creative Commons lisenssillä ja kuka tahansa voi rakentaa oman laitealustan niin halutessaan. (Arduino a.)



Kuva 14 Arduino Uno havainnekuva porteista ja liittimistä (Arduino 2016b).

Kuvassa 14 on Arduino Uno kortti, jossa digitaaliset portit ovat merkityt vihreällä värillä, analogiset portit sinisellä värillä, virtasyötöt 5V ja 3,3V oranssilla, USB liitin keltaisella ja ulkoinen virtaliitin X1 punaisella värillä (Arduino 2016b).

Teknisiä tietoja:

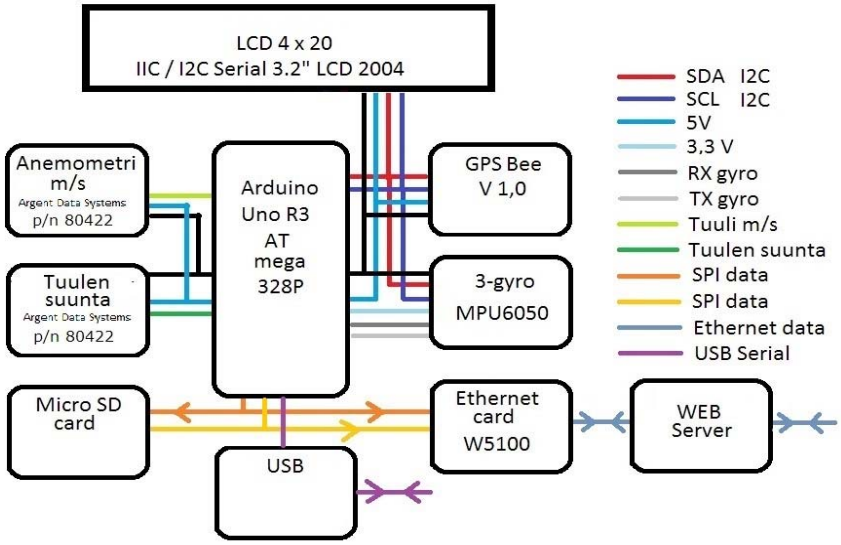
Prosessori	AT Mega 328P
Digitaalisia portteja	14
Analogisia portteja	6
Flash muisti	32kB
Kello	16Mhz

(Arduino 2016b)

3.2 Komponenttien valinta

3.2.1 Suunniteltu laitekokoonpano.

Kuvassa 15 on keskellä prosessorimoduuli ja ympärillä eri laitemoduulit ja niiden väliset virta- ja signaalijohtimet. Anemometri ja tuulensuuntaa mittaava anturi on kytketty n. 25 metriä pitkällä STP-parikaapelilla aluksen mastosta mittalaitteeseen radiohuoneessa. Muut komponentit ovat laitteen kotelon sisällä GPS-antennia lukuun ottamatta. Laitteen virtalähteenä toimivaa USB-laturia ei ole kuvattu tähän mukaan. Laite ei ole kiinni aluksen dataverkossa, vaan on täysin itsenäinen.



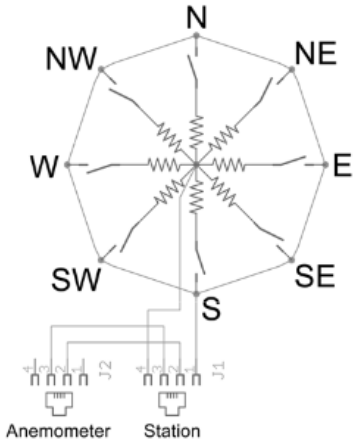
Kuva 15 Suunniteltu laitekokoonpano toiminnallisina lohkoina

3.2.2 Laitealusta Arduino Uno R3

Laitealustaksi valikoitui Arduino Uno R3 edullisuutensa, hyvän saatavuuden, runsaiden esimerkkiohjelmien ja laajan oheislaite- ja kirjastofunktioiden tuen vuoksi. Alustassa on valmiiksi 6 analogista ja 14 digitaalista kytkentäporttia, jotka ovat ohjelmallisesti valittavissa joko luku- tai kirjoitus (Input / Output) porteiksi. Laitteen muisti 32kB osoittautui myöhemmin liian pieneksi eräiden suunniteltujen toimintojen suorittamiseksi. Primääritavoitteisiin muisti silti riitti. Alusta sisältää lisäksi HiD-yhteensopivan USB-portin virransyöttöä varten. Sama liitäntä toimii myös laitteen ohjelmointiporttina ja sitä voidaan käyttää myös IO-kanavana ulkopuolisiin laitteisiin mittaustietoa syöttämään. Ohjelmointivaiheessa on debuggaus, eli muuttujien arvojen seuranta ja näin mahdollisten virheiden etsintä USB-portin välityksellä sarjaliikennemonitorin kautta enemmän kuin hyödyllinen apuneuvo. (Arduino 2016 c.)

3.2.3 Tuulen suunta- ja nopeusanturit

Tuulen suunta- ja nopeusanturit oli tarkoituksenmukaisinta valita olemassa olevasta sääasemasta, joista langallisten anturien vaatimuksen täytti osto hetkellä vain yksi malli, eli Weather Station WH1080. Anturit ovat toteutetut ns. reed-kytkimillä, joita anturin akseliin liitetty magneetti ohjaa auki/kiinni asentoihin. Tuulensuunta-anturissa kytkimiä on kahdeksan, yksi kullekin pää- ja välilmansuunnalle.

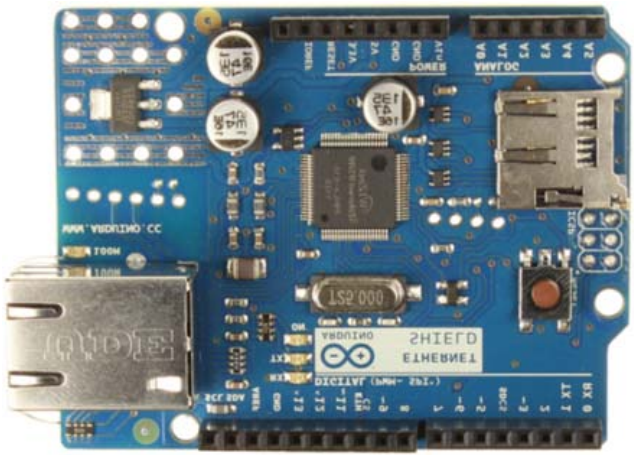


Kuva 15 Tuulensuunta-kytkimien kytkentäkaavio (Oberberger 2013).

Toiset kahdeksan ilmansuuntaa on näiden lisäksi lähinnä vain teoriassa mahdollisuus lukea, kun kaksi reed-kytkintä on magneetin kiinni vetäminä yhtä aikaa. Käytännössä tätä ”välitilaa” oli vaihtelevissa määrin vaikea saavuttaa, mutta nämä välilmansuuntien välisuunnat ovat kuitenkin mukana mittaus- ja laskenta-aineistossa. Reed-kytkimien toinen napa on liitetty Arduinon analogiseen lukuporttiin jännitetason mittausta varten, kun taas reed-kytkimen toinen napa on kytketty jännitetason määräävän vastuksen kautta nollapotentiaaliin. Lukuporttiin on kytketty 10kΩ ns. ylösvetovastus 5 voltin käyttöjännitteestä syöttämään tarvittava lähtöjännite mittausta varten. Samoin on anemometrin reed-kytkin kytketty ylösvetovastuksen kanssa Arduinon digitaaliseen porttiin pulssien lukua varten. Ylösvetovastukset sijoitin mahdollisimman lähelle antureita kytkentälaatikkoon ylös mastoon n. 30 cm päähän antureista. Näin pyrin kompensoimaan pitkän n. 25 m. mastosta laitteelle johtavan signaalikaapelin mahdollisesti aiheuttamaa jännitehäviötä.

3.2.4 Arduino ethernet-shield

Laitelisäkortteja kutsutaan shieldeiksi, suora suomennos olisi suoja tai suojakilpi, joka ei varsinaisesti kuvaa korttien käyttötarkoitusta. Näitä lisäkortteja on lukuisia eri tarkoituksiin ja yhteistä niille on, että ne ovat liitettävissä liitinrimojensa välityksellä suoraan Arduino Uno yhteensopiviin kortteihin toisiinsa kiinni painamalla. Näin esim. kuvan 17 ethernet-lisäkortilla laajenee Arduino Uno yhdellä 10/100MB ethernet portilla, mikro-SD muistikorttipaikalla tulosten tallennukseen ja noutoon sekä mahdollisuudella web-serverin ajamiseen. (Arduino 2016d.)



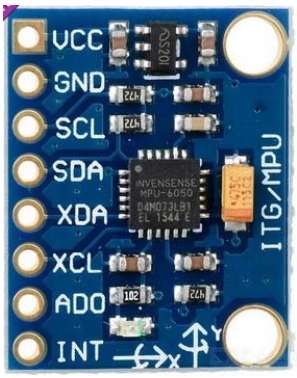
Kuva 16 Arduino Ethernet shield on kytkettävissä yhteen Arduino Uno- tai Megakortin kantaan. Oikeassa laidassa SD-korttipaikka. (Arduino 2016 d)

Kuvassa 18 alimpana on Arduino Uno, sitten ethernet-kortti ja päällimmäisenä oma kytkentäkortti, jossa GPS-anturi, kiihtyvyysanturi, ylösvetovastukset reedkytkimille, valodiodeille ja menu-kytkimelle, sekä komponenttien väliset virransyötöt ja signaalijohtimet.

Kuva 17 Kortit pinottuina päällekkäin koteloon.

3.2.5 Kiihtyvyysanturi

Kuvassa 19 on GY-521 gyro / kiihtyvyysanturimoduuli. Se käyttää Invensensin piiriä MPU-6050 ja I2C sarjaliikennekanavia (SDA = Serial Data, SCL = Serial Clock) tietoliikenteeseen. Lisäksi kytketään nollapotentiaali (maa) GND (Ground) liittimeen ja käyttöjännite Vcc liittimeen. Int liitin on keskeytyskanava (Interrupt) ja kytketään Arduino Unon digitaaliseen linjaliittimeen 2 tai 4. AD0 on osoiteliitin, jolla valitaan laitteelle yksi kolmesta osoitteesta: osoite #1 = maa, osoite #2 = kytkemättä, osoite #3 = käyttöjännite. XDA ja XCL nastat ovat moduulissa varattu ulkoisille lisäantureille kuten esim. magnetometri. (DX 2016a.)



Kuva 18 GY-521 Gyro anturissa ovat signaalit ja virtasyötöt selkeästi merkityt (DX 2016a).

3.2.6 GPS - anturi

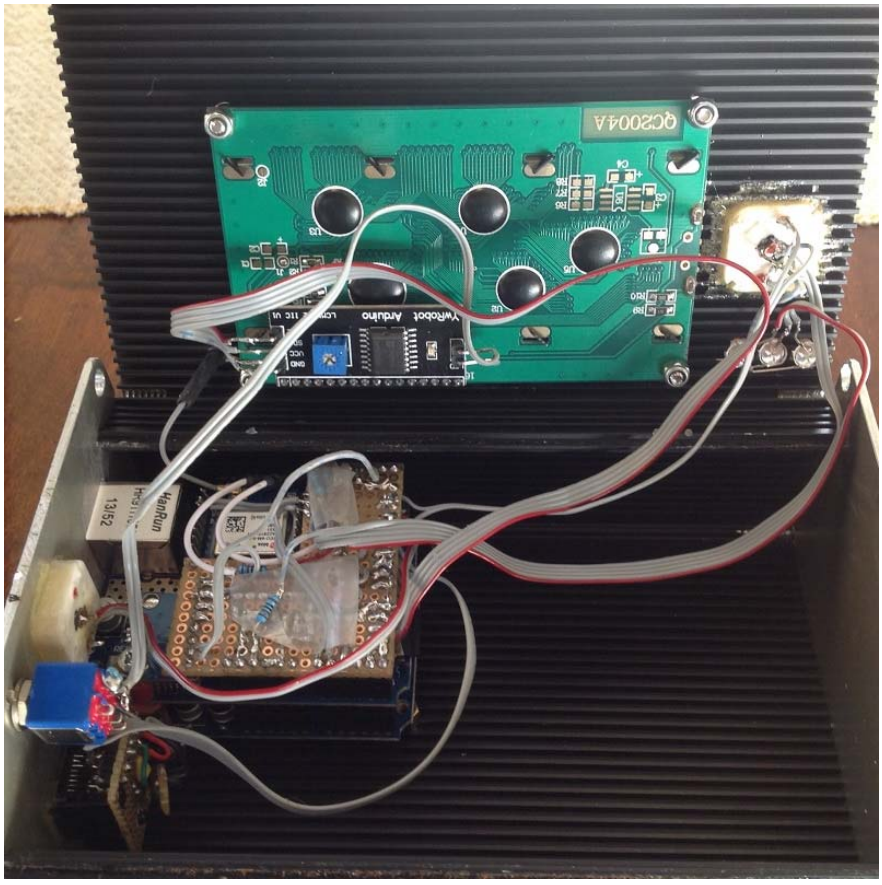
GPS-anturiksi valikoitui Electrofreaksin 50-kanavainen GPS-Bee Data Module Shield, koska sen erillinen antenni on sijoitettavissa kotelon ulkopuolelle ja mahdollisesti kaapelia jatkamalla myös kauemmaksikin itse näyttölaitteesta. Moduuli käyttää I2C sarjaliikennekanavaa kontrolliliikenteeseen, NMEA-lähetys- ja vastaanotto kanavat Tx/Rx ovat valittava Arduinosta erikseen. (DX 2016b.)

Kuva 19 ElektroFreaks:in GPS-anturi ja keraaminen antenni (DX 2016b).

3.2.7 LCD - näyttö

Kuvan 21 3,2 " LCD-näyttö on taustavalaistu, I2C sarjaliikennettä käyttävä moduuli, jossa on 20 merkkiä x 4 riviä = 80 merkkiä. Moduuli käyttää I2C sarjaliikennekanavaa tietoliikenteeseen. (DX 2016c.)

Kuva 20 4x20 merkin I2C LCD näyttö tarvitsee vain neljä johdinta (DX 2016c).



Kuva 22 Valmis laite sisältä johdotuksineen.

Kuten laitteen kuvasta 22 voi havaita, on sisällä vielä päivitysvaraa isommalle Arduino Mega 2560 kortille, joka vaihdetaan tulevaisuudessa. Kuvassa 23 on

laite käyttökuntoisena testiajossa. Keltainen LED-valo osoittaa että GPS yhteys on päällä, oikeassa alakulmassa näkyy satelliittien lukumäärä 6.

Kuva 23 Laite testiajossa

Kuvan 23 numerot viittaavat seuraaviin osiin ja toimintoihin:

1: sininen led-valo, vilkahtaa kun GPS-anturia luetaan.

2: punainen led-valo, vilkahtaa kun dataa kirjoitetaan tai luetaan tietueesta.

3: keltainen led-valo, vilkkuu, jos satelliitteja ei ole saatavilla. Palaa

yhtäjaksoisesti, kun luettavia satelliitteja on vähintään 3.

4: Menu kytkin, vaihtaa kytkintä painettaessa kuvan 23 ”ykkös” menun ”kakkos” menuksi, joka tulostaa näytölle edellisen päättyneen matkan TrueW arvot. Näyttö palaa 10 sekunnin jälkeen automaattisesti ”ykkös” menuun.

5 ja 6 LCD: päiväys ja kellonaika UTC

7 LCD: AppW, apparend wind, suhteellinen tuuli

8 LCD: TrueW, tosituuli, josta laivan liike laskettu pois

9 LCD: Logging / No logging, osoittaa, onko arvojen luku ja tallennus käynnissä.

10: Luettavissa olevien satelliittien määrä.

Liitteessä 1 on laitteen ruotsinkielinen käyttöopaste.

3.3 Moduulien käyttämät protokollat

3.3.1 USB - sarjaliikenneprotokolla

USB-portti on HiD-yhteensopiva, eli näkyy tietokoneelle Human Interface laitteena mahdollistaen vaikkapa hiiren/peliohjaimen rakentamisen Arduino laitealustalla. IDE-ohjelmointiympäristö käyttää USB-porttia Arduino alustan kanssa kommunikoimiseen ohjelmaa laitteeseen ladattaessa, sekä Serial Monitor sarjaliikenneohjelmaa interaktiiviseen kommunikointiin Arduino laitteen kanssa. Laitteessa ei itsessään näin tarvitse olla näyttöä tai toimintaa ohjaavia kytkimiä, laite voi kirjoittaa anturi- yms. tietoa laitteeseen kytkettyyn tietokoneeseen IDE Serial Monitor ohjelman kautta. Näppäimistötietoa voidaan syöttää Arduino laitteeseen samasta Serial Monitor käyttöliittymästä. Samoin on laitteen USB sarjaporttiin syöttämä tieto mahdollista ottaa vastaan millä tahansa sarjaporttiliikennettä lukevalla menetelmällä. Jos lähetettävä tieto on esim. CSV-protokollan mukainen (Comma Separated Value), osaavat useat taulukkolaskentaohjelmat lukea ja jatko käsitellä sitä. (Arduino 2016a.)

3.3.2 I2C - protokolla

I2C tai IIC toisella tavalla ilmaistuna on alun perin Philips Semiconductorsin kehittämä sarjaliikenneprotokolla, jossa laitteet ovat yhdistetty vain kahdella johtimella; SDA eli datalinja ja SCL eli kellolinja. Lisäksi laitteille tulee yhteinen maa ja kullekin käyttöjännitteensä mutta johtimia ei laitteiden välillä tarvita kuin kaksi. Tämä yksinkertaistaa kytkentäkorttien ja johdotusten suunnittelua ja Arduinon tapauksessa säästää luku/kirjoitusportteja vaikka ohjattavia laitteita olisikin useita. Arduinossa SDA linja on analogisessa liittimessä A5, sekä SCL linja liittimessä A4. Molemmat linjat ovat ns. open-drain tyyppisiä, eli Arduino pystyy asettamaan linjan loogiseen 0 tilaan, mutta looginen 1 vaatii ylösvetovastuksen käyttöjännitteestä. Riittää, että kytkentä tehdään kerran esim. isäntä laitteessa. (I2C Bus 2016a.)

Isäntälaitteita (Master) saa olla yksi ja siihen kytkettynä orjalaitteita (Slave) 7-bittisellä perusosoitteella maksimissaan 128 kpl. (Robot Electronics 2016)

Huomattava on, että saman tyyppisten laitteiden osoiteavaruus perusosoitteensa lisäksi on vain 2-bittinen, eli näin samaa laitetyyppiä voi olla vain kolme kytkettynä saman isäntälaitteen kanssa. Perusosoitteen vaihto tapahtuu esim. AD0-liittimellä kytkemällä se joko maahan 0-potentiaaliin (Gnd), käyttöjännit-

teeseen (Vcc) tai jättämällä se ”kellumaan”, eli kytkemättä sitä mihinkään. Näin sen jännitetaso jää jonnekin loogisen 0:n ja loogisen 1:sen välille. Protokollaa on kehitetty vuosien saatossa, nopeutta kasvatettu 100 kilobitistä 3,4 megabittiin sekunnissa. 10-bittinen osoiteavaruus laajentaa orjalaitteiden määrän 1008:aan. Myös mahdollisuus kytkeä useampia isäntälaitteita samaan linjaan on tehty, kunhan ne kaikki ovat ns. MultiMaster isäntälaitteita. (I2C Bus 2016b.)

3.3.3 SPI - protokolla

SPI (Serial Peripheral Interface Bus) käyttää tiedonsiirrossa neljää eri signaalijohdinta. Yksi on kellopulssi SCK (Serial Clock), joka tahdittaa kaikkien laitteiden tiedonsiirtoa ja ajoitusta. Toinen on MOSI (Master Output Slave Input), jolla isäntälaite lähettää dataa vastaanottavalle orjalaitteelle. Kolmas signaalijohdin on käänteinen edellisestä eli MISO (Master Input Slave Output), jolla isäntälaite vastaanottaa orjalaitteen lähettämää dataa. Neljäs on SS (Slave Select) tai CS (Chip select), jolla isäntälaite valitsee kulloisenkin orjalaitteen keskustelukumppanikseen. Tiedonsiirto I2C väylästä poiketen on siis Full Dublex tyyppistä, jossa laitteet voivat sekä lukea että kirjoittaa toisilleen yhtä aikaa. (Arduino 2016e.)

Arduinon ethernet-shield kortilla ethernet piiri ja mikro SD piiri käyttävät SPI protokollaa tiedonsiirtoon. Ethernet käyttää SS-signaalilinjana Arduino Unossa digitaalista liitintä 4 ja SD piiri digitaalista liitintä 10. (Arduino 2016e.)

3.3.4 Laitteen ohjelmointi

3.3.4.1 IDE - ohjelma

Ohjelmointiympäristönä on Arduinon sivuilta ladattava ohjelmisto IDE, joka on saatavilla Windows, Mac ja Linux versioina. Käytännön kannalta ei keskinäisiä eroja ole. Sama ohjelma käy kaikkien Arduino-prosessorikorttien ohjelmoimiseen. Käytössä ovat peruskirjastot, lisälaitekohtaisia kirjastoja erimerkkikoodeineen voi ladata käyttöön yksinkertaisella kirjastotyökalulla IDE-ohjelmassa tai asentamalla kirjastotiedostot manuaalisesti oikeaan paikkaan ohjelmistopuussa. Kirjasto on käyttävissä IDE-ohjelman uudelleenkäynnistyksen jälkeen. (Arduino 2016f.)

3.4 Komponenttien konseptitestaus

3.4.1 LCD - näyttö

LCD näyttö keskeisimpänä komponenttina laitteen toiminnan seuraamiseen ja tulosten tarkasteluun oli ensimmäinen kytketty ja testattu komponentti.

I2C LCD näytön käyttämiseksi tarvitaan **Wire.h** kirjasto I2C linjan käyttöön ja **LiquidCrystal_I2C.h** kirjasto I2C LCD moduulin käyttöön. Ohjelman alussa alustetaan LCD osoitteeseen 27 hex.

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

Setup() osassa käynnistetään LCD: **lcd.begin(20,4);** 20 merkkiä, 4riviä ja näin on näyttö käyttövalmis.

lcd.setCursor(3,0); siirtää kursorin ensimmäiselle riville (0), neljännen kirjaimen kohdalle rivin alusta.

lcd.print("Moro"); kirjoittaa tekstin Moro näyttöön.

lcd.clear(); pyyhkii koko näytön puhtaaksi.

3.4.2 Gyro / kiihtyvyysanturi

Pintaliitoskomponentein valmistetun GY-521 moduulin koko on n.20 x 30 mm. Käyttöjännite on valittavissa 3.3 ja 5 voltin välillä, lisäksi moduuli sisältää jänniteregulaattorin. Anturin virrankulutus on gyroskooppi käytössä tyypillisesti n. 3,6 mA, valmiustilassa 5 uA. Kiihtyvyysanturin mittausalueet ovat: ±2, ±4, ±8, ±16 g. Oletusarvona ±2 g. Gyroskoopin kulmanopeuksien lukualueet ovat ± 250, ±500, ±1000 ja ±2000 °/s, joista ± 250 °/s oletusarvoasetus. Piirin gyroskooppiosa antaa tietoa x-y-z akseleiden suhteen piirin asennosta, jolloin on mahdollista laskea mm. kaltevuuskulmat akseleiden suhteen sekä kulmanopeudet eli kuinka nopeasti piiri kiertyy jokin akselinsa suhteen. Koska dataa saadaan koko ajan sekä gyroskoopista ja kiihtyvyysantureilta 3-akselin suhteen, kykenee piiri tuottamaan liike-ja kiihtyvyys tietoa 3 x 3-matriisiin. Piiri liikeprosessori (DMP = Digital Motion Prosessor) laskee arvot 1024 tavun puskuriin (FIFO = First In First Out) ja ne saadaan ulos mm. Eukleen-kulmina ja kvaternioina. (Invensense 2014.)

Koska arvot vaihtelevat pienistäkin asennonmuutoksista johtuen, tulee laite asentaa loppusijoituspaikkaansa tukevasti ja ennen tuotantokäyttöönottoa kalibroida erillisellä ohjelmalla, joka etsii sopivat offset-arvot kyseiselle piirille ja asennuspaikalle. Offset arvoja käytetään omassa koodissa laitteen alustuksen yhteydessä. Esimerkiksi

```
mpu.setXGyroOffset(-3);

mpu.setYGyroOffset(-5);

mpu.setZGyroOffset(5532);

mpu.setZAccelOffset(1640);

mpu.setXAccelOffset(-2047);

mpu.setYAccelOffset(-152);
```

Anturin offset asetusten etsiminen ja perusdatan saaminen anturista kävi esimerkkiohjelmalla helposti (Rodenas 2015).

Jo konseptitestausvaiheessa ilmeni, ettei Arduino Uno R3:n muisti riitä anturin ajamiseen lopullisessa laitteessa.

3.4.3 GPS - moduuli

GPS moduuli käyttää TinyGPS++.h kirjastoa (Hart 2015) ja lähti helposti käyntiin Codebenderissä julkaistulla esimerkkiohjelmalla (Codebender 2015).

Koska työssä tarvittiin vain päivämäärä, aika ja kulkunopeus, karsittiin ”ylimääräinen” esimerkkikoodista pois ja otettiin omaan koodiin sellaisenaan.

3.4.4 Ethernet / mikro SD / Web-serveri

Verkkokortin käyttöönottoon löytyy Arduino forumista valmiit esimerkit, samoin kuin SD-kortin alustamiseen ja käyttöön. Web-server koodimalleja löytyy myös useita, joista osa käyttää SD korttia lähteenään. Esimerkkien avulla kävi konseptitestaus nopeasti ja helposti. (Starting electronics 2014.)

Tämäkin osuus lopputuotteesta jouduttiin karsimaan muistin riittämättömyyden vuoksi.

3.5 Lopullinen ohjelman toiminta lyhyesti

Ohjelmakoodi on kokonaisuudessaan liitteessä 3 ja seuraavassa viitataan sen rivinumeroihin.

Prosessien ajoaikavaatimukset vaihtelevat niiden luonteen mukaan jonkin verran. Tuulen suunnan ja GPS:n antaman nopeuden lukemiseen arveltiin riittävän n. 1 – 2 minuuttia, tuulen nopeuden määrittämiseen 1 – 2 sekuntia ja kiihtyvyysanturia tulisi lukea niin paljon kuin vain ehtii. Yksittäiset arvonvaihteluiden tasoittamiseksi käytetään koodissa keskiarvojakaumaa kaikkien arvojen määrittelyissä. Koska prosessori ei kykene moniajoon, täytyy eri antureiden lukuarvoja lukea kutakin vuorotellen talteen kunnes haluttu määrä näytteitä on saatu ja tämä summa sitten jaetaan näytteiden määrällä ja otetaan jatkokäsittelyyn. Antureita luetaan jatkuvana prosessina mutta niiden jatkokäsittely ja talteenotto tapahtuu vain laivan ollessa liikkeessä. Tämä arvojen jatkokäsittely ja talteenotto, eli loggaaminen, alkaa kun laivan keskinopeus ylittää 10 lukukierroksen aikana 2,9 solmua. Tieto saadaan GPS-moduulista keskiarvojakaumana. Yksi lukukierros on n. 1030 ms – 1050 ms. Joka sadannen kierroksen jälkeen lasketaan keskiarvot tuulen suunnalle sekä nopeudelle. Näistä lasketaan ns. True-arvot, joissa laivan liikkeen aiheuttama muutos tuulitietoon on kompensoitu pois. Saadut arvot talletetaan koko matkan kulkua taltioivaan tietueeseen matkan loputtua tapahtuvaa loppuyhteenvetoa varten. Tämä joka sadas kierros kestää 1250 ms.

Empiiristen kokemusten seurauksena lisättiin logituksen alkuun vielä viivettä n. 5 minuuttia, jotta Helsingistä lähtiessä laivan kulkusuunnan vaihtaminen satamassa saadaan tehtyä ennen varsinaisen matkaksi laskettavan ajan alkamista. Samoin todettiin matkan päättymisen vaativan viivettä Maarianhaminassa tapahtuvaan ympäri kääntymiseen satamaan saavuttaessa, joten toinen 5 minuuttia lisättiin myös matka-ajan loppuun. Näin saatu yhteensä 10 min. viive jakaantuu kahteen osaan ja aktualisoituu kokonaisuudessaan vain Maarianhaminassa pysähdyttäessä.

3.5.1 Tuulen suunta

Tuulen suuntatieto luetaan 16 alkiota käsittävään tietueeseen **w_DirArray{17}** funktiossa **void windDir()** riveillä 304 – 466. Tuulen suuntien indeksointi menee 0 - 15 alkaen pohjoisesta N (aluksen keula) ja kiertyen myötä päivään.

Tietueen 17:sta alkio indeksillä {16} on lukukertojen laskuri. Logituksen ollessa käynnissä, kasvatetaan tämän laskurin arvoa 100:aan samalla kun **w_DirArray{}** tietueen jotakin tuuli-indeksin osoittamaa 0 – 15 arvoa kasvatetaan sitäkin yhdellä. Jokaisella lukukierroksella luetaan myös tuulen nopeus m/s ja sijoitetaan se **w_SpdArray{}** tietueeseen tuuli-indeksin osoittamaan alkioon ja summataan olemassa olleen alkion arvon kanssa yhteen. Kun lukukertojen laskuri saavuttaa arvon 100 rivillä 427, etsitään **w_DirArray{}** tietueesta suurin tuulen suuntaindeksi muuttujaan **w_max** rivillä 429 ja **w_SpdArray{}** tietueesta luetaan edellä havaitun suurimman suuntaindeksin mukaan tuulen yhteenlaskettu nopeus ja jaetaan se suuntaindeksin osoittamalla lukukertojen määrällä rivillä 431. Näin on saatu 100 lukukerran keskiarvojakaumat tuulen suunnalle indeksilukuna 0 – 15 ja tuulen nopeudelle m/s kyseessä olevalle tuulen suunnalla.

Tässä vaiheessa kyseessä olevat arvot ovat ns. Apparent Wind, eli havaitut arvot suhteessa laivan runkoon ja sisältävät siten myös laivan kulkusuunnasta ja nopeudesta johtuvat komponentit. Jotta halutut True Wind arvot saataisiin selville, täytyy laivan kulun aiheuttama osuus laskea arvoista pois.

3.5.2 Tuulen nopeuden mittaus

Tuulen nopeuden mittaukseen löytyi valmis pulssilaskuri jonka arvo lukuaikaan suhteutettuna antaa tuulen nopeuden m/s, rivit 56 – 69. Koska laskenta tapahtuu kokonaan prosessorin rekistereissä, ei aikaa laskemiseen kulu pääohjelmassa loop() sinällään lainkaan vaan tapahtuu funktion **delay(1000);** määräämän yhden sekunnin viiveen ajan. Tätä aikaikkunaa, jonka muu koodi ”odottaa” laskentaa tapahtuvaksi, käytetään laitteen muiden prosessien ajamiseen.

```
/*
* HardwareCounting sketch
*
* uses pin 5 on 168/328, pin 47 on Mega
*/
unsigned int count;
unsigned int getCount()
{
TCCR1B= 0 ;                          // Gate Off / Counter Tn stopped
count = TCNT1;
TCNT1 = 0;
bitSet(TCCR1B ,CS12); // Counter Clock source is external pin
bitSet(TCCR1B ,CS11); // Clock on rising edge
```

```
bitSet(TCCR1B ,CS10); // you can clear this bit for falling edge
return count;
}
void setup()
{
Serial.begin(9600);
digitalWrite(5, HIGH);     // hardware counter setup (see ATmega data sheet
for details)
TCCR1A=0;                   // reset timer/counter control register A
getCount();                 // this will start the clock
}
void loop()
{
delay(1000);
Serial.println(getCount());
}
```

(Margolis, 567-568)

3.5.3 True arvojen laskeminen

Funktiolle **void CountTrue()**, rivit 488 – 582, annetaan muuttujien **w_Ind** ja **w_Spd** osoitteet tähdellä * osoitettuna, sekä GPS:stä saatu kulkunopeus muuttujissa **gps_Spd** metreinä sekunnissa ja kulkusuunta muuttujassa **gps_Ind** asteina 0 - 359. Koska tässä vaiheessa **gps_Ind** on asteina ja **w_Ind** on 0 - 15 indeksinä, pitää GPS:n astetieto muuntaa tuuli-indeksitiedoksi funktiossa **int Deg2Ind()** riveillä 248 – 302. Näin saaduilla tiedoilla lasketaan tuulen summavektorille uusi suunta ja nopeus. Nämä arvot palautuvat **w_Ind** ja **w_Spd** muuttujissa funktiosta ja kirjoitetaan talteen tietueisiin **w_SortArray1{}**, rivillä 441 ja **w_SpdArray1{}** rivillä 442.Tietueeseen osoittava indeksi **w_old** viittaa alkuperäiseen tietueeseen ennen kuin tuuli-indeksille laskettiin uusi arvo, koska tuulitieto on kuitenkin vielä ”vanhan” indeksin osoittamassa paikassa. Näihin tietueisiin kerätään True-arvot koko matkan ajalta, matkan päätteeksi tulostetaan jälleen keskiarvojakauma suurimman tuuli-indeksin osoittaman alkion arvojen mukaan. Ensin etsitään suurin tuuli-indeksi **w_SortMax** rivillä 452 ja sen mukaan haetaan tuulen yhteenlaskettu nopeus tietueesta **w_SpdArray1{}** ja jaetaan se indeksin lukumäärällä **MaxSampleD** rivillä 454. Jotta tuulen nopeus, joka datatyypiltään on float eli neljän tavun liukuluku, voitaisiin jakaa lukukerroilla, joka on tyypiltään kahden tavun kokonaisluku eli integer, pitää jakaja muuntaa myös liukuluvuksi. Muistin vähyyden vuoksi ei valmiita matemaattisia kirjastofunktioita voitu tähän käyttää vaan muunto piti tehdä laskemalla funktiolla **void Int2Float()** riveillä 468 – 474.

Näin saadaan selvitettyä keskinopeus ja -suunta tuulelle, joka on ollut suunnaltaan vallitsevin matkan aikana. Nämä arvot säilyvät tarkasteltavina laitteen menussa kaksi aina seuraavan matkan päättymiseen saakka, jonka jälkeen ne yli kirjoitetaan uusimmilla tuulitiedoilla.

Koska kyseessä oli tekijälle ensimmäinen tämän tyyppinen laite ja ohjelmointityö, alkoi koodi rakentua pitkälti konseptitestauksen palasia yhdistelemällä. Koska prosessoreita on vain yksi, eikä silläkään moniajo ole mahdollista, oli eri laitemoduulien toiminnan yhteen lomittaminen ja prosessien ajoajan jakaminen haastavaa ja mielenkiintoista.

3.6 Lopullinen testaus

3.6.1 Anemometri ja tuuli-indeksi

Anemometri kalibroitiin sijoittamalla se A/C-koneen sisälle tasaiseen ilmavirtaan. A/C konetta ajettiin manuaalisesti, jotta voitiin varmistua muuttumattomista olosuhteista. Anturia ajettiin vuorotellen alkuperäisellä sääasemalla ja sitten rakennetulla laitteella. Saatuja arvoja verrattiin keskenään ja sopivaa kerrointa käyttäen saavutettiin yhtenevät näyttämät n. 7 m/s tuulelle. Ilmavirran nopeus oli suurin, mitä koneesta saatiin. Kuten kuvasta 24 voidaan nähdä, ovat laivan oma ja laitteen käyttämä anemometri mastossa lähellä toisiaan. Näin voidaan järjestelmää hienosäätää vielä asennuksen jälkeenkin vertailemalla saatuja arvoja laivan oman järjestelmän kanssa.

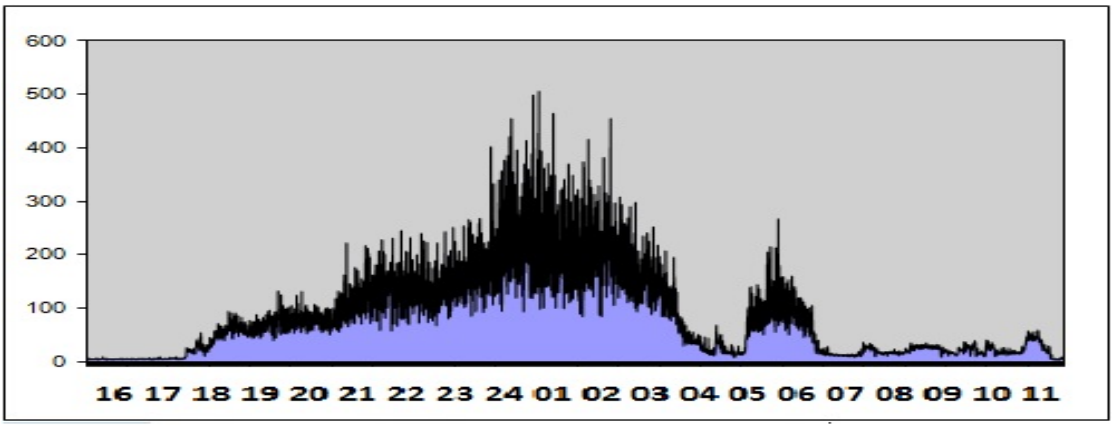
Kuva 21 Laivan ja laitteen anturit m/s Gabriellan mastossa

3.6.2 True - arvot

True-arvojen testaamiseen käytettiin ohjelmasta versiota, jossa muuttujien tuulen nopeus, aluksen nopeus ja kulkusuunta arvot syötettiin manuaalisesti ja saatuja True-arvoja tarkasteltiin tuuli-indeksin arvoa muuttamalla ja vertaamalla erikseen laskettuun arvoon. Keskeisiä asioita tarkastelussa olivat tuulen

nopeuden muutokset eri tulokulmilla suhteessa laivan runkoon, sekä tuulen suunnan vaihtelut vastaisesta myötäiseen verraten tuulen nopeuteen ja laivan nopeuteen. Esim. tuuli-indeksin näyttäessä vastaista tuulta, pitää huomioida lasketun tuulen nopeuden suhde laivan kulkunopeuteen. Jos tuulen nopeus on alempi kuin laivan kulkunopeus, on kyseinen tuuli kulkusuunnan myötäinen, vaikka tuuli-indeksi näyttääkin vastatuulta. Liitteessä 2 on koottuna saadut testitulokset, joista selviää laitteen laskemat tulokset ajoarvoja muutettaessa. Tuulen nopeudeksi on valittu 5 m/s kaikkiin testeihin. Ajosuunta ja nopeus muuttuvat taulukkokohtaisesti merkittyinä. Testattu tuuleen suuntanäyttämä taulukon vasemmassa sarakkeessa on näyttämä laivan rungon suhteen, eli N = keula, S = perä jne., mutta laskettu tuulen suunta sarake osoittaa tuulen tosisuunnat kartan ilmansuuntien mukaan.

3.6.3 Gyro - moduuli



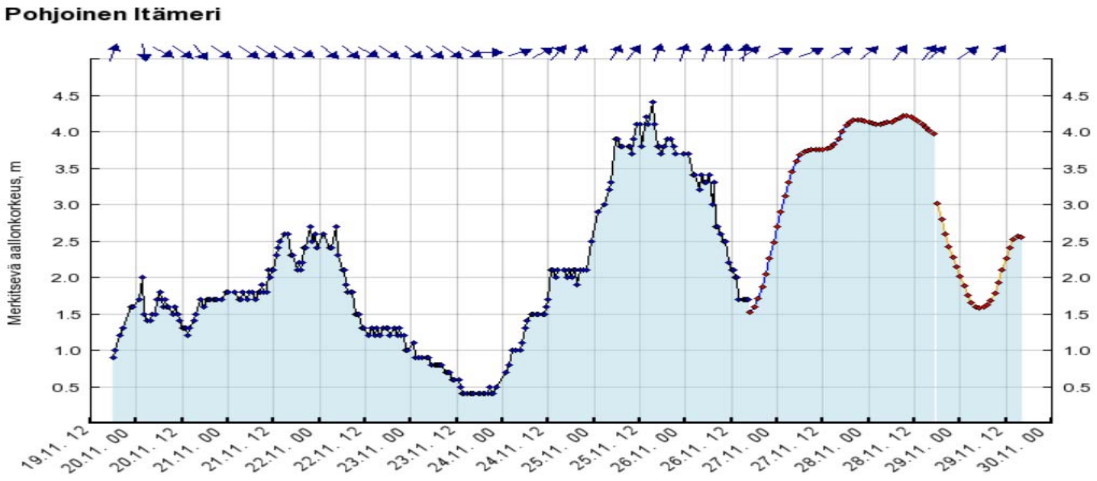
Kuva 22 Helsinki - Tukholma matkan kiihtyvyysarvojen kuvaaja 25.11.2015

Koska tässä tutkielmassa oli tarkoitus etsiä tapoja muuttuvan aallonkorkeuden vaikutusten arvioimiseen, tuntui mielekkäämmältä keskittyä kiihtyvyysanturin antamiin arvoihin asentoarvoja silti mitenkään ylenkatsomatta. Kuvassa 25 on 25.11.2015 alkaneen matkan taltioidut kiihtyvyydet Excel kuvaajaksi muokattuna. Lähtö Helsingistä on ollut n. 17:30, Pohjoisen Itämeren kautta on saavuttu Maarianhaminaan n. 04:20. Ahvenanmerelle on saavuttu n. 05:00 ja Tukholman saaristoon n. 06:45. Perillä Tukholmassa oltiin 11:00. Kuvaajan arvoja myötäilee kuvan 26 sääennuste, jossa odotettiin n. 3 metrin aallokkoa. Kuvassa 27, Ilmatieteenlaitoksen aallonkorkeusmittaus Pohjoisella Itämerellä 25.11.2015 päivälle osoittaa merkitsevän aalllonkorkeuden olleen 4,5 metriä.

Datum	Tid Sve	Vind på 10m höjd (m/s) riktn	medel	byvind	Sign våghöjd (m)	Max våghöjd (m)	Ström (knop) riktn	hast
2015-11-25								
Helsingfors	16:30	-	-	-	-	-	-	-
Gråhara	16:53	-	-	-	-	-	-	-
	17:00	210	11	14	-	-	-	-
	18:00	210	13	16	2.3	3.3	360	0.1
Porkala	18:12	-	-	-	-	-	-	-
	19:00	200	13	15	2.2	3.3	350	0.1
	20:00	200	14	16	2.3	3.4	060	0.2
Russarö	20:24	-	-	-	-	-	-	-
	21:00	190	13	16	2.5	3.7	060	0.2
	22:00	190	14	17	2.8	4.2	050	0.2
Utö	23:00	180	16	19	3.2	5.0	030	0.1
2015-11-26								
	00:00	190	14	17	3.0	4.6	060	0.1
	01:00	190	7	9	-	-	-	-
	02:00	220	9	10	-	-	-	-
Nyhamn	02:28	-	-	-	-	-	-	-
	03:00	230	7	8	-	-	-	-
Mariehamn	03:20	-	-	-	-	-	-	-
	04:00	240	5	6	-	-	-	-
Söderarm	05:00	250	6	7	1.6	2.3	160	0.2
	06:00	250	5	5	-	-	-	-
	07:00	260	5	6	-	-	-	-
	08:00	260	6	7	1.1	1.5	190	0.1
	09:00	270	5	5	-	-	-	-
Stockholm	10:00	270	6	6	-	-	-	-

Kuva 23 Sääennuste Helsinki - Tukholma reitille 25.11.2015 (ms Gabriella 2015)

Kiihtyvyysanturin dataa on mahdollista käsitellä joko maan vetovoiman huomioon ottavana tai se pois laskettuna, eli yksinomaa kiihtyvyyden muutosta mittaavana suureena kaikkien kolmen akselin suhteen. Koska tällöin on maan vetovoimavaikutus laskettu pois datasta, häviää mukana myös anturin x-y-z suuntainen asentotieto. Jos tätä tarvitaan, tulee asento huomioida koodissa jollain muulla menetelmällä. Koska kyseessä oli konseptitestaus alukseen vaikuttavien kiihtyvyyksien havainnointiin, tyydyttiin koodissa vain laskemaan yhteen havaittujen kiihtyvyyksien itseisarvot 50 arvon keskiarvojakaumana. Korrelaatio havaittiin vertailemalla saatuja empiirisiä arvoja mm. sääennusteeseen ja havaittuun aallonkorkeusmittaukseen Ilmatieteenlaitokselta.



Kuva 24 Ilmatieteenlaitoksen aallonkorkeutta ilmaiseva kuvaaja, josta on nähtävissä, että merkitsevä aallonkokeus 25.11.2015 yöllä oli 4,5 m. (Ilmatieteenlaitos 2015)

4 JOHTOPÄÄTÖKSET

Kyseessä oli ensimmäinen tekijän mikrokontrollerin rakennus- ja ohjelmointityö ja tarjosi se siten haasteita monella tavalla. Jo pelkästään muuttujien arvojen tulostus LCD näytölle tuntui aluksi ihmeelliseltä saavutukselta. Ehkäpä innostusta riittikin juuri näiden pienten onnistumisten myötä, kun anturien luku alkoi sujua ja keskittyminen voitiin kohdistaa halutun laisen tiedon esille saamiseen suuremmitta vaikeuksitta. Toki paljon piti opiskella muiden koodia, jotta Arduino-kontrolleri alkoi taipua haluttuihin toimiin. Mahdollisuus käyttää korkean tason C++-kieltä mahdollisti aloittelijankin koodin ajamisen ja aiempi käsitys kontrollereiden vaikeasta ohjelmoinnista muuttui täysin. Yllättävää oli näinkin pienen kontrollerin monipuolisuus ehkä Unon muistin pienuutta lukuun ottamatta. Mutta sekin saadaan kohta korjattua ja sitten ei ole kuin mielikuvitus rajana, mitä mikrokontrollerilla voi tehdä.

Laite nykyisellä kokoonpanolla palvelee tarkoitustaan, johon se rakennettiin. Osa ihmisaistein suoritetusta tiedonkeruusta saatiin automaation piiriin. Tosin vieläkin pitää lopputieto laitteesta siirtää käsin matkaraporttiin, koska olemassa oleva raportointijärjestelmä ei tunne liittymäpintoja ulkopuolisiin tiedon syöttöjärjestelmiin.

Laitteen vakaudesta ja herkkyydestä ei tämän työn osalta voi tehdä kovin tarkkoja johtopäätöksiä, koska se vaatisi pitkällistä vertailua jonkin vastaavan laitteen rinnalla. Kiihtyvyysanturia testattaessa ilmestyi sen dataan satunnaisia suuria arvojen hypäyksenomaisia muutoksia, kun GPS-anturi ajettiin rinnalle käyntiin. Ongelmaa ei alettu tutkia sen tarkemmin koska tiedossa jo oli, että kiihtyvyysanturin kirjastojen suuri koko estää sen käytön Uno kortilla primääritavoite ohjelmiston kanssa. Mutta ongelma tullee ratkottavaksi Arduino Mega-kortin kanssa työskenneltäessä.

Joka tapauksessa tarjoaa tällainen harrastepohjainenkin mikro-kontrolleri oikein hyvän mahdollisuuden tutkia ja muokata erilaisten anturien antamaa dataa ja siten avartaa käyttäjänsä ymmärrystä alusten automaatiojärjestelmien toiminnasta pintaa syvemmältä.

5 JATKOKEHITYS

5.1.1 Tuulen suunta ja voimakkuus

Tuulen suunnan ja voiman vaikutuksia arvioitaessa voidaan tarkastella esimerkiksi kuvan 28 matkaa Helsingistä Maarianhaminaan ja katsoa, millaista tietoa matkasta olisi mahdollista generoida samalla lähtöaineistolla. Matka kestää 11 tuntia. Ajatellaan että tuulen suunta on koko ajan etelästä, tuuli voimistuu 0,5 m/s tunnissa koko matkan ajan alkaen 4m/s ja päättyen 9 m/s.

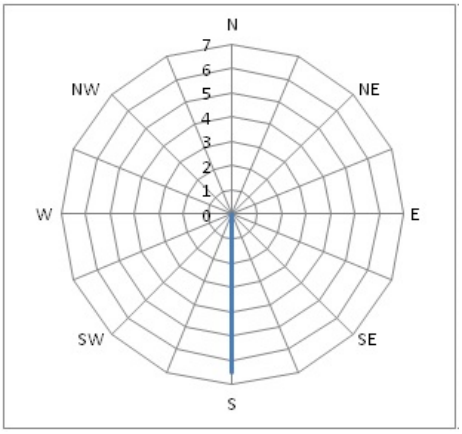
Matkan aikana rekisteröidään havaitut tuuliolosuhteet ja lasketaan niistä vallitseva tuulen suunta ja nopeus.

Mitattu keskituuli on näin ollen 6,5 m/s ja suunta S. Tämä tieto kirjoitetaan talteen matkaraporttiin matkan päätteeksi.

Kuva 25 Matkan kuvaaja välillä Helsinki - Maarianhamina (muokattu kartor.eniro.se 2016)

u

Kuvassa 26 kuvaajaksi muutettuna on data esitys varsin yksiselitteinen, kuten lienee tarkoituskin.



Kuva 26 Tuulen suunta ja nopeus yksiuloitteisena kuvaajana

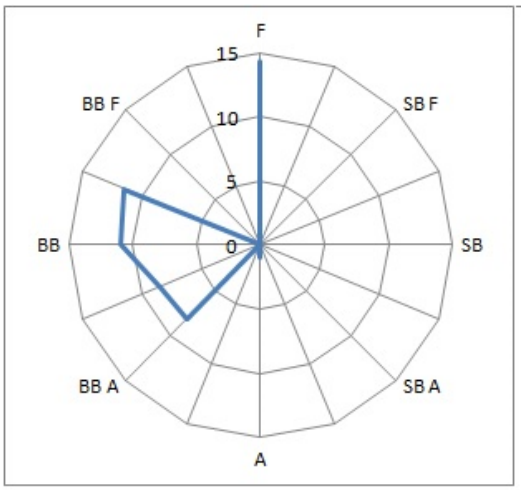
Kerättyä dataa havaitusta tuulesta laivan rungon suhteen ei käytetä sinällään hyväksi, mutta seuraavassa pohditaan edellä saadun aineiston analysointia siinä suhteessa, että olisiko data informatiivisempaa toisin esitettynä.

Lähtöarvot ovat samat kuin alussa mutta laivan liikkeiden aiheuttamat arvot edelleen mukana:

w_SpdArray[985,0,0,0,0,0,0,0,0,0,846,294,760,795,0,0]; sisältää yhteenlaskettuna tuulen kokonaismäärän m/s per havaittu tuulen suunta 0-15

w_sortArray[69,0,0,0,0,0,0,0,0,0,104,35,69,69,0,0]; sisältää lukukerrat per havaittu tuulen suunta 0-15.

Jakamalla indeksin osoittamat arvot tietueista keskenään, saadaan tuulen keskinopeudeksi esim. indeksillä [0] (= N) 985/69 = 14 m/s.

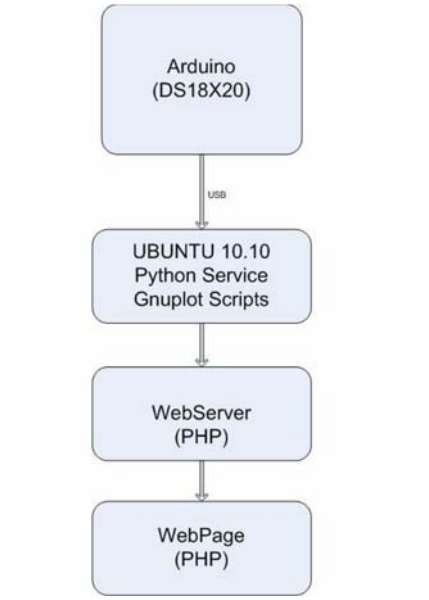


Kuva 27 Tuulen vaikutuksen kuvaus aluksen runkoon koko matkan ajalta.

Kuvaajaksi muutettuna saadaan kuvasta 30 selkeämpi käsitys tuulen suunnista laivan rungon suhteen. Huomattavaa on ero vasta- (F) ja myötätuulten (A) välillä, vaikka loppumatkasta myötätuuli onkin ollut etelästä 9 m/s. Kuvaajassa kasvaa vastatuulen 4 m/s osuus matkan alussa arvoon 14 m/s ja vastaavasti kompensoituu takatuuli 9 m/s yhdeksi metriksi sekunnissa vastaista laivan kulkiessa kyseiseen aikaan 10 m/s myötätuuleen. Samoin alkupuoliskoa matkan hallinnut sivuvastainen ja puolenvälin jälkeen vallinnut sivumyötäinen kuvautuvat selkeästi. Toisen suuntaisella matkalla olisi kuvaaja peilikuva edellisestä.

5.1.2 MPU-6050 kiihtyvyysanturi ja Sea State

Kiihtyvyysanturia aiotaan kokeilla lisää, kunhan prosessorikortti saadaan vaihdettua Arduino Mega-korttiin, jossa muistia on 256 kb Unon 32 kb sijaan. Kokeiltavaksi jää, kuinka aaltojen frekvenssi saadaan datasta selvitettyä. Mahdollisuuksia sen laskemiseen on monia. Lisäksi kytkettänee laite Linux-serveriin, jotta kiihtyvyysanturin merkittävästi suurempi datamäärä saataisiin siirretyksi talteen ja kuvatuksi helppotajuisempaan graafiseen muotoon esim. GnuPlot ohjelmalle (Väyrynen 2011).



Kuva 28 Lohkokaavio Arduinon kytkemisestä Linux-serveriin (Väyrynen 2011).

LÄHTEET

ABB. 2007. ABB elibrary. Torductor-S. https://library.e.abb.com/public/eef8846525950f25c1257380003623c6/3BSE022227R0101_-001.pdf [viitattu 1.3.2015]

ABB. 2014a. Coriolis Master. Operating instruction. https://library.e.abb.com/public/d65feba9d7ab7fd7c1257cdd0046180e/OI_FCB300_FCH300_EN_F.pdf [viitattu 1.3.2016]

ABB. 2014b. ABB power lehti 2/14. Energiatehokkuuden edelläkävijä - Case: Viking Grace & EMMA http://www.slideshare.net/ABBSuomi/abb-emma [viitattu 24.2.2016]

ABB. 2016a. Advisory Systems. http://new.abb.com/marine/systems-and-solutions/automation-and-advisory/advisory [viitattu 23.2.2016)

ABB. 2016b. ABB elibrary https://library.e.abb.com/public/334a5fe2aec2d1b2c1257a8a003bc786/ABB%20Generations_19%20Emma%20Ship%20Energy%20Manager.pdf [viitattu 24.2.2016]

About Shipping. 2012. What is the SEEMP http://about-ship.blogspot.fi/2012/05/what-is-seemp.html [viitattu 25.2.1016]

Andersson S. 2015. Päällikkö m/s Grace. Haastattelu 11.8.2015 m/s Gabriella.

Arduino. 2016a. Inroduction to the Arduino Board. https://www.arduino.cc/en/Reference/Board [viitattu 31.3.2016]

Arduino. 2016b. Introduction to the Arduino board. https://www.arduino.cc/en/Reference/Board [viitattu 31.3.2016]

Arduino. 2016c. What is Arduino? https://www.arduino.cc/en/Guide/Introduction [viitattu 17.12.2015]

Arduino. 2016d. Arduino Ethernet Shield https://www.arduino.cc/en/Main/ArduinoEthernetShield [viitattu 3.3.2016]

Arduino. 2016e. SPI Library https://www.arduino.cc/en/Reference/SPI [viitattu 5.3.2016]

Arduino. 2016f. Arduino Software (IDE) https://www.arduino.cc/en/Guide/Environment [viitattu 19.12.2015]

codebender. 2015. TinyGPS++: Full example https://codebender.cc/example/TinyGPS++/FullExample#FullExample.ino [viitattu 12.5.2015]

de Weck, Olivier L. 2001. Attitude Determination and Control (ADCS) http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-851-satellite-engineering-fall-2003/lecture-notes/l9_acs.pdf [viitattu 1.3.2016]

DX. 2016a. http://www.dx.com/fi/p/gy-521-mpu6050-3-axis-acceleration-gyroscope-6dof-module-blue-154602#.Vv0aso9OLIV [viitattu 22.2.2016]

DX. 2016b. http://www.dx.com/fi/p/elecfreaks-50-channel-gps-bee-data-mini-embedded-antenna-module-shield-board-for-arduino-381150#.Vtf7fNA0j7w [viitattu 22.2.2016]

DX. 2016c. http://www.dx.com/fi/p/arduino-iic-i2c-serial-3-2-lcd-2004-module-display-138611#.Vtf7LNA0j7w [Viitatu 22.2.2016]

Eniram. 2016. Products http://www.eniram.fi/products/ [viitattu 24.2.2016]

Hart M. 2015 GitHup, Inc. TinyGPS++ library files. https://github.com/mikalhart/TinyGPSPlus/tree/master/examples [viitattu 12.5.2015]

InvenSense. 2014. MPU-6050 Data Sheet http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/ [Viitattu 13.2.2016]

I2C Bus.2016a. I2C – What's that? http://www.i2c-bus.org/ [viitattu 5.3.1016]

I2C Bus. 2016b. MultiMaster http://www.i2c-bus.org/MultiMaster/ [viitattu 5.3.2016]

Ignatius J. 2012. Viking Grace Teknologia http://www.vikinggrace.com/abbs-emma-advisory-suite-ms-viking-grace/ [viitattu 26.2.2016]

Ignatius J. 2014. ABB's Marine Advisory System. http://www.thedigitalship.com/conferences/presentations/2014bergen/day1_ses/DSBergen_Jukka_Ignatius.pdf [viitattu 26.2.2016]

Ilmatieteen laitos. 2016. Aallokko Itämerellä http://ilmatieteenlaitos.fi/aallokko-itamerella [viitattu 22.3.2016]

IMO. 2011. MEPC.1/Circ.683. Form of a ship energy efficiency management plan (SEEMP) http://www.imo.org/en/MediaCentre/HotTopics/GHG/Documents/683%20SEEMP.pdf#search=MEPC%2E1%2FCirc%2E683 [Viitattu 22.2.2016]

IMO. 2012.MEPC 63/23, Annex 9. 2012 Guidelines for the development of a ship energy efficiency management plan (SEEMP) http://www.imo.org/en/KnowledgeCentre/IndexofIMOResolutions/Documents/MEPC%20-%20Marine%20Environment%20Protection/213(63).pdf#search=MEPC%2063%2F23 [viitattu 22.2.2016]

IMO. 2015a. MEPC.203(62), annex 19. Amendments to the annex of the protocol of 1997 to amend the international convention for the prevention of pollution from ships, 1973, as modified by the protocol of 1978 relating thereto http://www.imo.org/en/OurWork/Environment/PollutionPrevention/AirPollution/Documents/203(62).pdf#search=MEPC%2E203%2862%29 [viitattu 25.2.2016]

IMO 2015b. MEPC.1/Circ.684. Annex guidelines for voluntary use of the ship energy efficiency oprational indicator (EEOI) http://www.imo.org/en/OurWork/Environment/PollutionPrevention/AirPollution/Documents/Circ-684.pdf#search=MEPC%2E1%2FCirc%2E684 [viitattu 25.2.2016]

IMO. 2016a. Introduction to IMO http://www.imo.org/en/About/Pages/Default.aspx [viitattu 22.2.2016]

IMO 2016b. Energy Efficiency Measures http://www.imo.org/en/OurWork/Environment/PollutionPrevention/AirPollution/Pages/Technical-and-Operational-Measures.aspx [viitattu 25.2.2016]

Lehto P. 2014. Opinnäytetyö Laiva-automaation suunnittelu ja ohjeistus. https://www.theseus.fi/bitstream/handle/10024/70923/Pasi_Lehto.pdf?sequence=1 [viitattu 26.2.2016]

MNX. 2016. What is MEMS Technology? https://www.mems-exchange.org/MEMS/what-is.html [viitattu 31.3.2016]

Margolis Michael. 2011. Arduino Cookbook. O'Reilly. 567-568

Oberberger M. 2013. Arduino Weatherstation. http://maxoberberger.net/projects/arduino-weatherstation/ [viitattu 31.3.2016]

Onnettomuustutkintakeskus.1998. Tutkintaselostus B2/1998 M. msGabriella, keulaportin vaurio Suomenlahdella 24.10.1998 http://turvallisuustutkinta.fi/material/attachments/otkes/tutkintaselostukset/fi/vesiliikenneonnettomuuksientutkinta/1998/b21998m_tutkintaselostus/b21998m_tutkintaselostus.pdf [viitattu 24.3.2016]

Polvinen J. 2012. Matematiikan pro-gradu tutkielma Kompleksiluvut ja kvaterniot kiertoina. Jyväskylän yliopisto. https://jyx.jyu.fi/dspace/bitstream/handle/123456789/40539/URN:NBN:fi:jyu-201212053307.pdf?sequence=1 [viitattu 1.3.2016]

Pursiainen H. 2011. Opinnäytetyö Kallistuman mittaaminen kiihtyvyysanturilla järjestelmän asennon säätämiseksi. Jyväskylän ammattikorkeakoulu. https://www.theseus.fi/handle/10024/32689 [viitattu 30.2.2016]

Physics 24/7.2016. Torque Sensors and Domain Theory http://www.physics247.com/physics-tutorial/torque-sensors.shtml [viitattu 30.2.1016]

Robot Electronics. 2016. Using the I2C Bus. http://www.robot-electronics.co.uk/i2c-tutorial [viitattu 5.3.2016]

Rodenas, Luis. s.a. Arduino Sketch to automatically calculate MPU6050 offsets http://www.i2cdevlib.com/forums/topic/96-arduino-sketch-to-automatically-calculate-mpu6050-offsets/ [viitattu 21.3.2016]

Rosenqvist M. 2016. Sähkömestari m/s Grace. Haastattelu 15.1.2016 m/sGrace

Starting Electronics. 2014. Arduino SD card web server https://startingelectronics.org/tutorials/arduino/ethernet-shield-web-server-tutorial/SD-card-web-server/ [viitattu 23.3.2016]

Soininen H., Karppinen T., Kahma K., Kukkanen T., Rintala S. 2001. Ro-Ro Passenger Ship Bow Door Structural Design. SNAME Transactions, Vol. 109, 2001, pp. 269-283 http://www.sname.org/HigherLogic/System/DownloadDocumentFile.ashx?DocumentFileKey=d0aa0ff5-e900-43d4-b347-030d87a6d502 [viitattu 21.3.2016]

Slideshare. 2014. ABB Power lehti 2/14. Energiatehokkuuden edelläkävijä. case: Viking Grace & EMMA. http://www.slideshare.net/ABBSuomi/abb-emma [viitattu 24.2.2016)

Vuorinen J. 2014. Opinnäytetyö Energiatehokkuuden tehostaminen Viking XPRS aluksella. YAMK Kymenlaakson ammattikorkeakoulu. https://publications.theseus.fi/bitstream/handle/10024/72090/vuorinen_juha.pdf?sequence=1 [viitattu 2.3.2016]

Väyrynen A. 2011. Opinnäytetyö Mikrokontrollerin integroiminen Linux-käyttöjärjestelmään. Oulunseudun ammattikorkeakoulu. https://www.theseus.fi/bitstream/handle/10024/26666/Vayrynen_Ari.pdf?sequence=1 [viitattu 31.3.2016]

LIITE 1

Weather Datalogger v 1.0 18.2.2016

Så här ser apparaten ut. Data samling börjar automatiskt efter GPS mottagaren for kontakt med åtminstone 3 satelliter (gul diod 3 lyser) och färjan har medelhastighet > ~3 knop. Blå diod 1 blinkar hela tiden och visar att GPS-mottagning pågår. Rör diod 2 lyser då å då när apparaten uppdaterar datarekord 8.

Huvud menu 1

Datum (5) och tidpunkt (6) visas hela tiden. OBS, tiden är UTC, Coordinated Universal Time.

Råd 7 visar Apparent Wind, vind hastighet och rigtning såsom dom påvärkar fartyg. Råd 8 visar kalkulerad värdena för vind rigtning och hastighet utan fartygets rörelse.

Råd 9 visar status om data samlas eller inte. Punkt 10 visar antal satelliter. 4 är Menu knapp. Den ändrar visning av skärmen. Efter ~10 sek. ändrar visningen tillbaka om man inte har tryckt knappen igen tidigare.

Här är andra skärmen efter knappen. Den visar statistiken av den sista resan tills nästa resa är gjord.

Huvud menu 2

Råd 1 visar medetal av vind hastighet för hela resan.

Råd 2 visar vind riktning, som har varit skenbar för det mest för resan.

Värdena hittas härifrån tills nästa resa är över och är överskriven med nya värdena. Knapp 3 lyser 50%.

LIITE 2

Mittapöytäkirja 17.2.2016

Ajosuunta	**0**	**astetta**	
Ajonopeus	**3**	**kn =**	**1,543333 m/s**

Suunta	Tuuli m/s	lask. suunta	lask. nopeus m/s
NW	5	NWW	3,17
W	5	SWW	5,23
SW	5	SSW	5,89
S	5	S	6,54
SE	5	SSE	5,89
E	5	SEE	5,23
NE	5	NEE	3,17
N	5	NNW	3,34

Ajosuunta	**0**	**astetta**	
Ajonopeus	**10**	**kn =**	**5,144444 m/s**

Suunta	Tuuli m/s	lask. suunta	lask. nopeus m/s
NW	5	S	2,3
W	5	SW	7,17
SW	5	SSW	9,36
S	5	S	10,14
SE	5	SSE	9,36
E	5	SE	7,17
NE	5	S	2,3
N	5	S	0,14

Ajosuunta	**0**	**astetta**	
Ajonopeus	**20**	**kn =**	**10,28889 m/s**
Suunta	Tuuli m/s	lask. suunta	lask. nopeus m/s
NW	5	SSW	6,6
W	5	SW	11,44
SW	5	S	14,43
S	5	S	15,29
SE	5	S	14,43
E	5	SE	11,44
NE	5	SSE	6,6
N	5	S	5,29
Ajosuunta	**45**	**astetta**	
Ajonopeus	**3**	**kn =**	**1,543333 m/s**
Suunta	Tuuli m/s	lask. suunta	lask. nopeus m/s
NW	5	NNW	3,17
W	5	NWW	5,23
SW	5	SWW	5,89
S	5	SW	6,54
SE	5	SSW	5,89
E	5	SSE	5,23
NE	5	SEE	3,17
N	5	NE	3,46

Mittapöytäkirja 17.2.2016

Ajosuunta	**45**	**astetta**	
Ajonopeus	**10**	**kn =**	**5,144444 m/s**

Suunta	Tuuli m/s	lask. suunta	lask. nopeus m/s
NW	5	SW	2,3
W	5	W	7,17
SW	5	SWW	9,36
S	5	SW	10,14
SE	5	SSW	9,36
E	5	S	7,17
NE	5	SW	2,3
N	5	SW	0,14

Ajosuunta	**135**	**astetta**	
Ajonopeus	**3**	**kn =**	**1,543333 m/s**

Suunta	Tuuli m/s	lask. suunta	lask. nopeus m/s
NW	5	NEE	3,17
W	5	NNE	5,23
SW	5	NNW	5,89
S	5	NW	6,54
SE	5	NWW	5,89
E	5	SWW	5,23
NE	5	SSW	3,17
N	5	SE	3,46

Ajosuunta	**135**	**astetta**	
Ajonopeus	**10**	**kn** =	**5,144444 m/s**
Suunta	Tuuli m/s	lask. suunta	lask. nopeus m/s
NW	5	NW	2,3
W	5	N	7,17
SW	5	NNW	9,36
S	5	NW	10,14
SE	5	NWW	9,36
E	5	W	7,17
NE	5	NW	2,3
N	5	NW	0,14
Ajosuunta	**225**	**astetta**	
Ajonopeus	**3**	**kn** =	**1,543333 m/s**
Suunta	Tuuli m/s	lask. suunta	lask. nopeus m/s
NW	5	SSE	3,17
W	5	SEE	5,23
SW	5	NEE	5,89
S	5	NE	6,54
SE	5	NNE	5,89
E	5	NNW	5,23
NE	5	NWW	3,17
N	5	SW	3,46

Mittapöytäkirja 17.2.2016

Ajosuunta	**225**	**astetta**	
Ajonopeus	**10**	**kn =**	**5,144444 m/s**

Suunta	Tuuli m/s	lask. suunta	lask. nopeus m/s
NW	5	NE	
W	5	E	2,3
SW	5	NEE	7,17
S	5	NE	9,36
SE	5	NNE	10,14
E	5	N	9,36
NE	5	NE	7,17
N	5	NE	2,3
			0,14

Ajosuunta	**315**	**astetta**	
Ajonopeus	**3**	**kn =**	**1,543333 m/s**

Suunta	Tuuli m/s	lask. suunta	lask. nopeus m/s
NW	5	SWW	3,17
W	5	SSW	5,23
SW	5	SSE	5,89
S	5	SE	6,54
SE	5	SEE	5,89
E	5	NEE	5,23
NE	5	NNE	3,17
N	5	NW	3,46

Ajosuunta	**315**	**astetta**	
Ajonopeus	**10**	**kn** =	**5,144444 m/s**
Suunta	Tuuli m/s	lask. suunta	lask. n opeus m/s
NW	5	SE	2,3
W	5	S	7,17
SW	5	SSE	9,36
S	5	SE	10,14
SE	5	SEE	9,36
E	5	E	7,17
NE	5	SE	2,3
N	5	SE	0,14

LIITE 3

```
1 #include <LiquidCrystal_I2C.h>
2 #include <TinyGPS++.h>
3 #include <SoftwareSerial.h>
4 #include "Wire.h"
5 #include <SPI.h>
6
7 LiquidCrystal_I2C lcd(0x27, 20, 4);
8 static const int RXPin = A1, TXPin = A0; //NMEA data
9 static const int GPSBaud = 9600;
10 TinyGPSPlus gps; // The TinyGPS++ object
11 SoftwareSerial ss(RXPin, TXPin); // The serial connection to the GPS device
12
13 int offSet = 0;
14 int Yled = 6;
15 int Rled = 7;
16 int SW1led = 8;
17 int Bled = 9;
18 int sensor_1 = 5;
19 int sensor_2 = A2;
20 int menuSw = 1;
21 bool menu2_ON = false;
22 int rounds = 0;
23 int r100 = 10;
24 float top_speed = 0;
25 int logDelayCount = 0;
26 int countD = 0;
27 int sp_count = 0;
28 String w_dirInd[16] =
{"N","NNE","NE","NEE","E","SEE","SE","SSE","S","SSW","SW","SWW","W","NWW","NW","NN
W"};
29 float w_degInd[16] =
{0,22.5,45,67.5,90,112.5,135,157.5,180,202.5,225,247.5,270,292.5,315,337.5};
30 int w_dirArray[17]; // First 100 samples to Dir calculations
31 float w_SpdArray[16]; // First 100 samples to Spd calculations
32 int w_sortArray1[16]; // Collecting table of True wind directions
throughout the voyage
33 float w_SpdArray1[16]; // Collecting table of True wind speeds throughout
the voyage
34 int w_max;
35 int w_SortMax;
36 int GPS_s;
37 float w_Spd;
38 float Gspeed;
39 String TrueInd = "No data";
40 float TrueWind;
41 int w_sensVal = HIGH;
42 int w_sens = HIGH;
43 bool blinkState = true;
44 bool logOn = false;
45 bool logOk = true;
46 bool logDelay = true;
47 bool stopDelay = false;
48 bool Gdown = false;
49 bool menu_Clear = false;
50 int menu2_Clear;
51 bool Menu2_Clear = false;
52 unsigned long count;
53 float gps_Speed;
54 float gps_Course;
55
56 unsigned long getCount()
```

```
57 {
58 TCCR1B = 0;
59 count = TCNT1;
60 TCNT1 = 0;
61 bitSet(TCCR1B , CS12);
62 bitSet(TCCR1B , CS11);
63 bitSet(TCCR1B , CS10);
64 // Gate Off / Counter Tn stopped
65 // Counter Clock source is external pin
66 // Clock on rising edge
67 // you can clear this bit for falling edge
68 return count;
69 }
70 // ================================================================
71 // === INITIAL SETUP ===
72 // ================================================================
73
74 void setup() {
75 lcd.init(); // initialize the lcd
76 lcd.backlight();
77 lcd.clear();
78 pinMode(Bled, OUTPUT); // initialize ports
79 pinMode(Rled, OUTPUT);
80 pinMode(Yled, OUTPUT);
81 pinMode(SW1led, OUTPUT);
82 pinMode(sensor_1, INPUT);
83 pinMode(sensor_2, INPUT);
84 pinMode(menuSw, INPUT);
85 analogWrite(SW1led, 135);
86 analogWrite(Bled, 150);
87 Wire.begin();
88 TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
89 Serial.begin(115200); //initialize serial monitor
90 ss.begin(GPSBaud); //Initialize GPS
91 attachInterrupt(menuSw,handle_click,RISING); //Initialize the Menu button
interrupt
92 digitalWrite(5, HIGH);
93 TCCR1A = 0;
94 getCount();
95 lcd.setCursor(0, 0); lcd.print("DATALOGGER v1.1 2016");
96 lcd.setCursor(0, 1); lcd.print("Arduino open project");
97 lcd.setCursor(0, 2); lcd.print("Freeware copyright");
98 lcd.setCursor(0, 3); lcd.print("2016 Juha Vainio");
99 feedGPS(5000);lcd.clear();
100 lcd.setCursor(0, 1); lcd.print("AppW:");
101 lcd.setCursor(0, 2); lcd.print("TrueW:");
102 lcd.setCursor(9, 2); lcd.print("Calculating");
103 lcd.setCursor(0, 3); lcd.print("NO LOGGING");
104 lcd.setCursor(11, 3); lcd.print("Nr SAT");
105 }
106 // ================================================================
107 // === MAIN PROGRAM LOOP ===
108 // ================================================================
109
110 void loop() {
111             windSpeed();
112                         if (stopDelay) {
113                         stopWatch();
114                         }
115             if (menu2_ON) {
116                         menu_Clear = true;
117                         if (menu2_Clear <= 10) {
118                         menu2();
119                         menu2_Clear++;
```

```
120                             analogWrite(SW1led, 0);}
121                             else {
122                             menu2_ON = !menu2_ON;
123                             menu2_Clear = 0;}
124             } else {
125             if (menu_Clear) {
126                             lcd.clear();
127                             lcd.setCursor(0, 1); lcd.print("AppW:");
128                             lcd.setCursor(0, 2); lcd.print("TrueW:");
129                             lcd.setCursor(9, 2); lcd.print("Calculating");
130                             lcd.setCursor(0, 3); lcd.print("NO LOGGING");
131                             lcd.setCursor(11, 3); lcd.print("Nr SAT");
132                             menu_Clear = false;}
133             menu1();
134             analogWrite(SW1led, 255);
135             }
136 }
137
138 static void stopWatch() {
139 countD += 1;
140             if (countD == 2) {
141             stopDelay = false;
142             countD = 0;
143             Serial.println("logging restart allowed");
144             }
145 }
146
147 void menu1() {
148 Menu2_Clear = false;
149 printDate(gps.date);
150 printTime(gps.time);
151 lcd.setCursor(18, 3); lcd.print("   ");
152 lcd.setCursor(18, 3); lcd.print(GPS_s);
153 }
154
155 void menu2() {
156             if (!Menu2_Clear) {
157             lcd.clear();
158             printDate(gps.date);
159             printTime(gps.time);
160             lcd.setCursor(0, 1); lcd.print("Last journey stats: ");
161             lcd.setCursor(0, 2); lcd.print("True Wind: ");
162             lcd.setCursor(12, 2); lcd.print(TrueWind);
163             lcd.setCursor(0, 3); lcd.print("True Dir: ");
164             lcd.setCursor(12, 3); lcd.print(TrueInd);
165             }
166 Menu2_Clear = true;
167 }
168
169 void handle_click () {
170 static unsigned long last_interrupt_time = 0;
171 unsigned long interrupt_time = millis();
172             if (interrupt_time - last_interrupt_time > 200 ) {
173             menu2_ON = !menu2_ON;
174             }
175 last_interrupt_time = interrupt_time;
176 }
177
178 float windSpeed() {
179 static int w_count;
180 float timeS = millis();
181 GPS();
182             if (!menu2_ON){
183             menu1(); }
```

```
184             delay(1000);
185             float time2 = millis() - timeS;
186             w_Spd = ((((getCount() * 1000) / (time2))*0.17));
187             windDir();
188                             if (!menu2_ON) {
189                             lcd.setCursor(9, 1); lcd.print(w_Spd);}
190 }
191
192 int Deg2Ind(float Degrees)
193 {
194             if (Degrees >=349 || Degrees < 11) {
195             Degrees = w_degInd[0];
196             }
197             else if (Degrees >= 11 && Degrees < 34) {
198             Degrees = w_degInd[1];
199             }
200             else if ( Degrees >=34 && Degrees < 56) {
201             Degrees = w_degInd[2];
202             }
203             else if (Degrees >= 56 && Degrees < 79) {
204             Degrees = w_degInd[3];
205             }
206             else if (Degrees >= 79 && Degrees < 101) {
207             Degrees = w_degInd[4];
208             }
209             else if (Degrees >= 101 && Degrees < 124) {
210             Degrees = w_degInd[5];
211             }
212             else if (Degrees >= 124 && Degrees < 146) {
213             Degrees = w_degInd[6];
214             }
215             else if (Degrees >= 146 && Degrees < 169) {
216             Degrees = w_degInd[7];
217             }
218             else if (Degrees >= 169 && Degrees < 191) {
219             Degrees = w_degInd[8];
220             }
221             else if (Degrees >= 191 && Degrees < 214) {
222             Degrees = w_degInd[9];
223             }
224             else if (Degrees >= 214 && Degrees < 236) {
225             Degrees = w_degInd[10];
226             }
227             else if (Degrees >= 236 && Degrees < 259) {
228             Degrees = w_degInd[11];
229             }
230             else if (Degrees >= 259 && Degrees < 281) {
231             Degrees = w_degInd[12];
232             }
233             else if (Degrees >= 281 && Degrees < 304) {
234             Degrees = w_degInd[13];
235             }
236             else if (Degrees >= 304 && Degrees < 326) {
237             Degrees = w_degInd[14];
238             }
239             else if (Degrees >= 326 && Degrees < 349) {
240             Degrees = w_degInd[15];
241             }
242             else {
243             Serial.println("Value out of range! Can't be...");
244             }
245             return Degrees;
246 }
247
```

```
248 int deg2Int(int Degrees)
249 {
250             if (Degrees >=349 || Degrees < 11) {
251             Degrees=0;
252             }
253             else if (Degrees >= 11 && Degrees < 34) {
254             Degrees=1;
255             }
256             else if ( Degrees >=34 && Degrees < 56) {
257             Degrees=2;
258             }
259             else if (Degrees >= 56 && Degrees < 79) {
260             Degrees=3;
261             }
262             else if (Degrees >= 79 && Degrees < 101) {
263             Degrees=4;
264             }
264             else if (Degrees >= 101 && Degrees < 124) {
265             Degrees=5;
266             }
267             else if (Degrees >= 124 && Degrees < 146) {
268             Degrees=6;
269             }
270             else if (Degrees >= 146 && Degrees < 169) {
271             Degrees=7;
272             }
273             else if (Degrees >= 169 && Degrees < 191) {
274             Degrees=8;
275             }
276             else if (Degrees >= 191 && Degrees < 214) {
277             Degrees=9;
278             }
279             else if (Degrees >= 214 && Degrees < 236) {
280             Degrees=10;
281             }
282             else if (Degrees >= 236 && Degrees < 259) {
283             Degrees=11;
284             }
285             else if (Degrees >= 259 && Degrees < 281) {
286             Degrees=12;
287             }
288             else if (Degrees >= 281 && Degrees < 304) {
289             Degrees=13;
290             }
291             else if (Degrees >= 304 && Degrees < 326) {
292             Degrees=14;
293             }
294             else if (Degrees >= 326 && Degrees < 349) {
295             Degrees=15;
296             }
297             else {
298             Serial.println("Value out of range! Can't be...");
299             }
300             return Degrees;
301 }
302
303 void windDir()
304 {
305 String w_dir;
306 digitalWrite(Rled, false);
307 int w_value = (analogRead(sensor_2) + offSet);
308             if (w_value >= 775 && w_value < 880) {
309             w_dir = "N"; w_dirArray[0] += 1;
310             if (logOn) { w_SpdArray[0] += w_Spd;}
311
```

```
312             if (!menu2_ON) {
313             lcd.setCursor(15, 1); lcd.print("N ");}
314             } // 1
315             else if (w_value >= 350 && w_value < 435) {
316             if (!menu2_ON) {
317             lcd.setCursor(15, 1); lcd.print("NNE");}
318             w_dir = "NNE"; w_dirArray[1] += 1;
319             if (logOn) { w_SpdArray[1] += w_Spd;}
320             }
321             else if ( w_value >= 435 && w_value <540) {
322             if (!menu2_ON) {
323             w_dir = "NE"; w_dirArray[2] += 1;}
324             if (logOn) { w_SpdArray[2] += w_Spd;}
325             if (!menu2_ON) {
326             lcd.setCursor(15, 1); lcd.print("NE ");}
327             }
328             else if (w_value >= 75 && w_value < 89) {
329             w_dir = "NEE"; w_dirArray[3] += 1;
330             if (logOn) { w_SpdArray[3] += w_Spd;}
331             if (!menu2_ON) {
332             lcd.setCursor(15, 1); lcd.print("NEE");}
333             }
334             else if (w_value >= 89 && w_value < 115) {
335             w_dir = "E"; w_dirArray[4] += 1;
336             if (logOn) { w_SpdArray[4] += w_Spd;}
337             if (!menu2_ON) {
338             lcd.setCursor(15, 1); lcd.print("E ");}
339             }
340             else if (w_value >= 50 && w_value < 75) {
341             w_dir = "SEE"; w_dirArray[5] += 1;
342             if (logOn) { w_SpdArray[5] += w_Spd;}
343             if (!menu2_ON) {
344             lcd.setCursor(15, 1); lcd.print("SEE");}
345             }
346             else if (w_value >= 165 && w_value < 230) {
347             w_dir = "SE"; w_dirArray[6] += 1;
348             if (logOn) { w_SpdArray[6] += w_Spd;}
349             if (!menu2_ON) {
350             lcd.setCursor(15, 1); lcd.print("SE ");}
351             }
352             else if (w_value >= 115 && w_value < 165) {
353             w_dir = "SSE"; w_dirArray[7] += 1;
354             if (logOn) { w_SpdArray[7] += w_Spd;}
355             if (!menu2_ON) {
356             lcd.setCursor(15, 1); lcd.print("SSE");}
357             }
358             else if (w_value >= 260 && w_value < 350) {
359             w_dir = "S"; w_dirArray[8] += 1;
360             if (logOn) { w_SpdArray[8] += w_Spd;}
361             if (!menu2_ON) {
362             lcd.setCursor(15, 1); lcd.print("S ");}
363             }
364             else if (w_value >= 230 && w_value < 260) {
365             w_dir = "SSW"; w_dirArray[9] += 1;
366             if (logOn) { w_SpdArray[9] += w_Spd;}
367             if (!menu2_ON) {
368             lcd.setCursor(15, 1); lcd.print("SSW");}
369             }
370             else if (w_value >= 620 && w_value < 700) {
371             w_dir = "SW"; w_dirArray[10] += 1;
372             if (logOn) { w_SpdArray[10] += w_Spd;}
373             if (!menu2_ON) {
374             lcd.setCursor(15, 1); lcd.print("SW ");}
375             }
```

```
376             else if (w_value >= 540 && w_value <= 620) {
377             w_dir = "SWW"; w_dirArray[11] += 1;
378             if (logOn) { w_SpdArray[11] += w_Spd;}
379             if (!menu2_ON) {
380             lcd.setCursor(15, 1); lcd.print("SWW");}
381             }
382             else if (w_value >= 915 && w_value < 1015) {
383             w_dir = "W"; w_dirArray[12] += 1;
384             if (logOn) { w_SpdArray[12] += w_Spd;}
385             if (!menu2_ON) {
386             lcd.setCursor(15, 1); lcd.print("W ");}
387             }
388             else if (w_value >= 820 && w_value < 860) {
389             w_dir = "NWW"; w_dirArray[13] += 1;
390             if (logOn) { w_SpdArray[13] += w_Spd;}
391             if (!menu2_ON) {
392             lcd.setCursor(15, 1); lcd.print("NWW");}
393             }
394             else if (w_value >= 860 && w_value < 915) {
395             w_dir = "NW"; w_dirArray[14] += 1;
396             if (logOn) { w_SpdArray[14] += w_Spd;}
397             if (!menu2_ON) {
398             lcd.setCursor(15, 1); lcd.print("NW ");}
399             }
400             else if (w_value >= 700 && w_value < 775) {
401             w_dir = "NNW"; w_dirArray[15] += 1;
402             if (logOn) { w_SpdArray[15] += w_Spd;}
403             if (!menu2_ON) {
404             lcd.setCursor(15, 1); lcd.print("NNW");}
405             }
406             else {
407             Serial.println(w_value); w_dir = "N/A";if (!menu2_ON) {
408             lcd.setCursor(15, 1); lcd.print("N/A");}
409             digitalWrite(Rled, blinkState);}
410
411             if (!menu2_ON) {
412             lcd.setCursor(9, 1); lcd.print(" ");
413             lcd.setCursor(9, 1); lcd.print(w_Spd);}
414
415 if (logOn && !menu2_ON) {
416 lcd.setCursor(0, 3); lcd.print("LOGGING ON");}
417
418 if (logOn) {
419 w_dirArray[16] += 1;}
420 else {
421 w_dirArray[16] = 0;
422             for (int i = 0; i < 16; i++) {
423             w_SpdArray[i] = 0.0;
424             w_dirArray[i] = 0;}
425 }
426
427 if (w_dirArray[16] == 100) {
428 float maxSampleD = 0.0;
429 w_max = FindMax(w_dirArray,0, 15 );
430 maxSampleD = Int2Float(w_dirArray[w_max]);
431 w_SpdArray[w_max] = (w_SpdArray[w_max] / maxSampleD);
432 int w_old = w_max;
433 CountTrue(&w_max,&w_SpdArray[w_max],gps_Course,gps_Speed);
434
435 if (logOn && !menu2_ON) {
436 lcd.setCursor(9, 2); lcd.print(" ");
437 lcd.setCursor(9, 2); lcd.print(w_SpdArray[w_old]);
438 lcd.setCursor(15, 2); lcd.print(" ");
439 lcd.setCursor(15, 2); lcd.print(w_dirInd[w_max]);
```

```
440 }
441 w_sortArray1[w_max] += 1;
442 w_SpdArray1[w_max] += w_SpdArray[w_old];
443 for (int i = 0; i < 17; i++) {
444                 w_dirArray[i] = 0;
445                 w_SpdArray[i] = 0.0;
446                 bool (blinkState = !blinkState);
447                 digitalWrite(Rled, blinkState);}
448 }//if rounds2
449
450 if (!logOn && !logOk) {
451 float MaxSampleD = 0.0;
452 w_SortMax = FindMax(w_sortArray1, 0, 15);
453 MaxSampleD = Int2Float(w_sortArray1[w_SortMax]);
454 w_SpdArray1[w_SortMax] = (w_SpdArray1[w_SortMax] / MaxSampleD);
455 TrueWind = w_SpdArray1[w_SortMax];
456 TrueInd = w_dirInd[w_SortMax];
457 logOk = true;
458 for (int i = 0; i < 16; i++) {
459                 w_sortArray1[i] = 0;
460                 w_SpdArray1[i] = 0.0;}
461 stopDelay = true;
462 lcd.setCursor(0, 3); lcd.print("NO LOGGING");
463 Serial.print("True wind dir: ");Serial.print(TrueInd);
464 Serial.print(" True wind speed: ");Serial.println(TrueWind);
465 }
466 }//void windDir()
467
468 float Int2Float(int w_max)
469 {
470 float int2fl = 0.0;
471 for (int i=0; i<w_max; i++){
472                 int2fl += 1.0;}
473 return int2fl;
474 }
475
476 int FindMax(int Array[], int Start, int Stop){
477 int Loc,Max;
478 Max = Array[Start];
479 Loc = Start;
480 for (int i = Start + 1; i < Stop; i++){
481                 if (Array[i] > Max){
482                 Max = Array[i];
483                 Loc = i;}
484                 }
485 return Loc;
486 }
487
488 void CountTrue(int *w_Ind, float *w_Spd, int gps_Ind, float gps_Spd) {
489 float w_x,w_y,g_x,g_y,SinA;
490 int Int, gps_Int,b;
491 float a,c;
492 gps_Int = deg2Int(gps_Ind);
493 //Calculate wind directions N and S, x-vektor 0
494 if (*w_Ind == 0 || *w_Ind == 8) {
495                 if (*w_Ind == 0) {
496                                 if (*w_Spd < gps_Spd) {
497                                 *w_Spd = (gps_Spd - *w_Spd);
498                                 *w_Ind = gps_Int + 8;}
499                 else {
500                                 *w_Spd = (*w_Spd - gps_Spd);
501                                 *w_Ind = gps_Int;}
502                 } // N
503 else {
```

```
504             *w_Spd = (gps_Spd + *w_Spd);
505             *w_Ind = gps_Int + 8;}//S
506 }// 0||8
507
508 //Calculate wind directions E ja W, y-vector = gps_Spd
509 else if (*w_Ind == 4 || *w_Ind == 12) {
510             *w_Spd = sqrt((pow(*w_Spd,2) + pow(gps_Spd,2)));
511             SinA=((sin(gps_Spd / *w_Spd))*(180/3.14));
512             Int = deg2Int(SinA);
513             if (*w_Ind == 4) {
514                             *w_Ind = *w_Ind + Int + gps_Int;
515             } // E
516 else {
517             *w_Ind = *w_Ind - Int + gps_Int;}//W
518 }// 4||12
519
520 //calculate other directions than N
521 else if ((*w_Ind > 0 && *w_Ind < 4) || (*w_Ind <= 15 && *w_Ind > 12)) {
522             bool North = false;
523                             if (*w_Ind > 0 && *w_Ind < 4) {
524                             w_x = abs(cos(w_degInd[*w_Ind]) * *w_Spd);
525                             w_y = abs(sin(w_degInd[*w_Ind]) * *w_Spd);
526             }
527             else {
528                             w_x = abs(cos(360 - w_degInd[*w_Ind]) * *w_Spd);
529                             w_y = abs(sin(360 - w_degInd[*w_Ind]) * *w_Spd);
530             }
531                             if (w_y < gps_Spd) { //wind less than vessel speed = backwind
532                             w_y = (gps_Spd - w_y);
533                             }
534             else {
535                             w_y = (w_y - gps_Spd); // wind more than vessel speed
536                             North = true;
537             }
538             *w_Spd = sqrt((pow(w_x,2) + pow(w_y,2)));
539             SinA=((sin(gps_Spd / *w_Spd))*(180/3.14));
540             Int = deg2Int(SinA);
541 if (*w_Ind > 0 && *w_Ind < 4) {
542             if (North) {
543                             *w_Ind = ((*w_Ind + Int) + gps_Int);}
544             else {
545                             Int = 8 - Int;
546                             *w_Ind = ((*w_Ind + Int) + gps_Int);
547                             North = false;} //
548             }
549             else {
550                             if (North) {
551                             *w_Ind = ((*w_Ind - Int) + gps_Int);}
552                             else {
553                             Int = 8 + Int;
554                             *w_Ind = ((*w_Ind + Int) + gps_Int);
555                             North = false;} //
556             }//
557 }// 11-15 & 1-3
558
559 else if ((*w_Ind > 4 && *w_Ind < 8) || (*w_Ind >= 9 && *w_Ind < 12)) {
560             if (*w_Ind > 4 && *w_Ind < 8){
561                             w_x = abs(cos((w_degInd[*w_Ind]- 90)) * *w_Spd);
562                             w_y = abs(sin((w_degInd[*w_Ind] - 90)) * *w_Spd);}
563             else {
564                             w_x = abs(cos((270 - w_degInd[*w_Ind])) * *w_Spd);
565                             w_y = abs(sin((270 - w_degInd[*w_Ind])) * *w_Spd);}
566                             *w_Spd = sqrt((pow(w_x,2) + pow(w_y + gps_Spd,2)));
567                             SinA=((sin(gps_Spd / *w_Spd))*(180/3.14));
```

```
568                             Int = deg2Int(SinA);
569                             if (*w_Ind > 4 && *w_Ind < 8) {
570                                     *w_Ind = ((*w_Ind + Int) + gps_Int);} //
571                             else {
572                                     *w_Ind = ((*w_Ind - Int) + gps_Int);}//
573                             }// 5-7 & 9-11
574             else{
575             Serial.println("No True vectors calculated! Can't be... :( ");}
576 if (*w_Ind > 15) {
577             *w_Ind = *w_Ind - 16;}
578 else if (*w_Ind < 1) {
579             *w_Ind = 15 + *w_Ind;}
580 if (*w_Ind > 15) {
581             *w_Ind = *w_Ind - 16;}
582 }// void CountTrue()
583
584 static void GPS() {
585 GPS_s = gps.satellites.value();
586 if (GPS_s > 2 && GPS_s < 7) {
587             analogWrite(Yled, 100);feedGPS(20);}
588 else if (GPS_s >= 7) {
589             analogWrite(Yled, 200);feedGPS(20);}
590 else {
591             feedGPS(50);}
592
593 if (millis() > 5000 && gps.charsProcessed() < 10)
594 Serial.println(F("No GPS data received: check wiring"));
595 float curr_speed = float(gps.speed.mph());
596 if (curr_speed > 2.9) {
597             top_speed = top_speed + curr_speed;
598             sp_count++;
599 }
600 else {
601             top_speed = 0;
602             sp_count = 0;
603             logOn = false;
604             logDelay = true;
605 }
606
607 if (logDelay && !logOn) {
608             if (!stopDelay && (top_speed / sp_count > 2.9)) {
609             logDelayCount++;
610                             if (logDelayCount == 30) {
611                             Serial.println("logDelay over");
612                             logDelay = false;
613                             logDelayCount = 0;
614                             logOn = true;
615                             logOk = false;
616                             }
617             }
618 }
619
620 if (rounds < r100) {
621             rounds++;
622 }
623 else {
624             rounds = 0;
625             gps_Course = (gps.course.deg());
626             gps_Speed = (((top_speed / sp_count) * 1.852) / 3.6);
627             sp_count = 0;
628             top_speed = 0;
629             }
630 }//GPS()
631
```

```
632 static void feedGPS(unsigned long ms) { //Feeds GPS
633 unsigned long start = millis();
634 do {
635             digitalWrite(Bled, true);
636                         while (ss.available())
637                         gps.encode(ss.read());
638                         }
639             while (millis() - start < ms);
640             digitalWrite(Bled, false);
641 }//feedGPS(xx)
642
643 char printDate(TinyGPSDate &d)
644 {
645 char* Date;
646 if (!d.isValid())
647 {
648             Serial.println(F("Date not valid "));
649 }
650 else
651 {
652             char sz[32];
653             sprintf(sz, "%02d.%02d.%02d ", d.day(), d.month(), d.year());
654             lcd.setCursor(0, 0); lcd.print(sz);
655                         if (!logOn && !logOk) {
656                         Serial.print(sz);Serial.print(" ");}
657                         }
658             feedGPS(10);
659 }
660
661 static void printTime(TinyGPSTime &t)
662 {
663 int i;
664 if (!t.isValid())
665 {
666             Serial.println(F("Time not valid"));
667 }
668 else
669 {
670             char sz[32];
671             sprintf(sz, "%02d:%02d ", t.hour(), t.minute());
672             lcd.setCursor(11, 0); lcd.print(sz); lcd.setCursor(17, 0); lcd.print("UTC");
673                         if (!logOn && !logOk) {
674                         Serial.println(sz);}
675                         feedGPS(10);
676                         }
677 }//void
```