Alex Huikko

GPS-autopilotti veneeseen

Opinnäytetyö

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

2023

Kaakkois-Suomen
ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Insinööri (AMK)
Tekijä/Tekijät	Alex Huikko
Työn nimi	GPS-autopilotti veneeseen
Toimeksiantaja	Teemu Manninen
Vuosi	2023
Sivut	39 sivua, liitteitä 1 sivua
Työn ohjaaja(t)	Teemu Manninen

TIIVISTELMÄ

Navigointilaitteet ja autopilotit veneille ovat yleensä kalliita. Tavallinen veneilijä käyttäisi mielellään autopilottia, jos se olisi esteettömämpi ja halvempi sekä yhteensopiva useiden eri ohjainlaitteiden kanssa.

Tässä opinnäytetyössä suunniteltiin ja toteutettiin itseohjautuva veneen autopilottijärjestelmä nimeltä AutoSailor. Työn tavoitteena oli kehittää autopilottijärjestelmä veneeseen, joka olisi helppokäyttöinen, edullinen ja vaatisi vain vähän osia. Järjestelmän tulisi olla helposti asennettavissa ja kytkettävissä useisiin eri ohjainlaitteisiin.

Opinnäytetyö koostuu kahdesta pääjärjestelmästä: käyttäjälle näkyvästä verkkopohjaisesta käyttöliittymästä eli etupuolesta (Front-End) ja käyttäjälle näkymättömästä takapuolesta, jossa navigointi tapahtuu (Back-End). Kommunikointi päätelaitteiden välillä toimii MQTT-viestinnällä lähiverkossa.

Työ aloitettiin suunnittelemalla autopilottijärjestelmälle verkkopohjainen käyttöliittymä, joka on alusta- ja kohderiippumaton. Järjestelmän etupuoli tarjoaa käyttäjälle helppokäyttöisen verkkopohjaisesta käyttöliittymän, jossa käyttäjä voi asettaa, tallentaa ja manipuloida erilaisia navigaatioreittejä, joita pitkin järjestelmä pyrkii navigoimaan. Järjestelmän takapuoli prosessoi käyttöliittymässä määritetyn navigaatiodatan ja ohjaa ohjainlaitetta kohti määritettyjä reittipisteitä. Opinnäytetyön ohella kehitettiin erilaisia järjestelmiä työn käyttöliittymän helppokäyttöisyyden ja esteettisyyden parantamiseksi.

Tämän opinnäytetyön perusteella voidaan todeta, että edullisen ja toimivan autopilottijärjestelmän toteutus on täysin mahdollista käyttämällä saatavilla olevia teknologioita.

Asiasanat: AutoSailor, autopilotti järjestelmä, verkkopohjainen käyttöliittymä, MQTT viestintä, helppokäyttöinen

South-Eastern Finland
University of Applied Sciences

Degree title	Bachelor of Engineering
Author (authors)	Alex Huikko
Thesis title	GPS autopilot for a boat
Commissioned by	Teemu Manninen
Time	2023
Pages	39 pages, 1 pages of appendices
Supervisor	Teemu Manninen

ABSTRACT

Navigation equipment and autopilots for boats are usually expensive. The average boater would prefer to use an autopilot if it were more accessible, cheaper, and compatible with various control devices.

In this thesis, a self-guiding autopilot system for boats called AutoSailor was designed and implemented. The aim of the work was to develop an autopilot system for boats that would be easy to use, affordable, and require only a few components. The system should be easy to install and connect to various control devices.

The thesis consists of two main systems: the user-visible web-based user interface, or frontend, and the user-invisible backend, where navigation takes place. Communication between the devices operates using MQTT messaging on a local network.

The work began by designing a platform- and target-independent web-based user interface for the autopilot system. The frontend of the system provides the user with an easy-to-use web-based interface, where users can set, save, and manipulate various navigation routes that the system navigates through.

The backend processes the navigation data defined in the user interface and directs the control device towards the specified waypoints. In addition to the thesis, various systems were developed to improve the ease of use and aesthetics of the work's user interface.

Based on this thesis, it can be concluded that the implementation of an affordable and functional autopilot system is entirely possible using available technologies.

Keywords: AutoSailor, autopilot system, web-based user interface, MQTT messaging, easy to use

SISÄLLYS

SELITYSLUETTELO

AutoSailor
Opinnäytetyöni sovelluksen nimike, helppokäyttöinen autopilottiratkaisu veneeseen.

MQTT (MQ Telemetry Transport)
MQTT (MQ Telemetry Transport): MQTT on protokolla, joka on suunniteltu kommunikoimaan Internet of Things (IoT) -laitteiden välillä.

Front-End
Verkkosivuston graafinen käyttöliittymä, johon käyttäjä pääse käsiksi. Toimii rajapintana AutoSailor sovellukselle.

Back-End
Käyttäjälle näkymätön ohjelmisto, vastuussa tietojen tallentamisesta ja käsittelystä.

EmojionEngine
Projektille rakentamani verkkosivupohjainen interaktiomoottori, joka lisää verkkosivun interaktiivisuutta.

GPS
Global Positioning System, satelliittipaikannusjärjestelmä.

Raspberry Pi
Yhden piirilevyn tietokone.

GPIO
Lyhenne sanasta General Purpose I/O, Raspberry Pi:n tapauksessa pinni joka voidaan ohjelmoida signaalin sisään tai ulostuloksi.

JavaScript
Verkkoympäristössä käytettävä ohjelmointikieli.

JSON

Yksinkertainen ja kevyt avoimen standardin tiedostomuoto tiedonvälitykseen ja tallennukseen.

I2C

Yksinkertainen kaksisuuntainen ohjaus- ja tiedonsiirtoväylä

Python

Korkean tason yleiskäyttöinen ohjelmointikieli.

Arduino

Avoimeen lähdekoodiin perustuva mikro-ohjain.

Sarjaporttiyhteys

Sarjaporttiyhteys (serial port connection) on tietokoneen ja ulkoisen laitteen välillä oleva sarjamuotoinen kommunikointikanava, jota käytetään siirtämään dataa kahden laitteen välillä.

UI

UI lyhenne sanoista User Interface, eli käyttöliittymä.

1 JOHDANTO

Veneiden navigointilaitteet ja autopilotit ovat kalliita, joten sain ohjaajalta idean rakentaa omatoimisesti autopilottijärjestelmä veneeseen. Aihe oli puoleensavetävä sen moninaisuuden takia, siinä elektroniikka sekä ohjelmointi sulautuvat hyvin yhteen.

Opinnäytetyön tavoitteena on rakentaa edullinen, helppokäyttöinen, modulaarinen sekä ennen kaikkea alustariippumaton prototyyppi autopilottiratkaisusta veneeseen. Nimesin sovellukseni AutoSailoriksi, se tulee sanoista Automatic ja Sailor (automaattinen ja merimies).

AutoSailor ohjaa venettä käyttäjän asettamien navigaatioreittien mukaisesti ja näyttää verkkopohjaisessa käyttöliittymässä käyttäjälle reaaliaikaista dataa. AutoSailorin Front-End ja Back-End keskustelevat MQTT-protokollan välityksellä jakaen tiedon JSON-tiedostomuodossa. Pienet JSON-datapaketit mahdollistavat nopean ja kaksisuuntaisen tiedon jakamisen molempien päätteiden välillä.

Kun käyttäjä suorittaa komennon verkkosivulla, sivu viestittää signaalilla tiedon Back-End:iin käyttäjän haluamasta komennosta. Kaikki tieto, mitä käyttäjän näkemässä käyttöliittymässä nähdään, heijastuu jokaiselle laitteelle reaaliaikaisesti riippumatta siitä, millä laitteella sivua käytetään. Tieto päivittyy dynaamisesti Front-Endin ja Back-Endin välillä.

Autopilotin pääyksikkönä toimii Raspberry Pi, joka jakaa lähiverkkoon edellä mainitun alustariippumattoman selainpohjaisen käyttöliittymän. Käyttöliittymän kautta käyttäjä pystyy näkemään autopilotin tiedot sekä määrittämään navigaatioreitin.

2 AUTOSAILORIN TOTEUTUS

Lyhyesti AutoSailor on ”tee-se-itse”-autopilottipaketti. Idea AutoSailorille syntyi halusta luoda helppokäyttöinen, alustariippumaton ja nopeasti asennettava veneenohjainsovellus, joka tarjoaa käyttäjille mahdollisuuden veneen autonomiseen ohjaukseen. Helppokäyttöisyydellä tarkoitan sovelluksen intuitiivista ja selkeää käyttöliittymää. Alustariippumattomuus tarkoittaa, että sovellus toimii useilla eri laitteilla ja alustoilla, kuten älypuhelimilla tai tableteilla.

Seuraavaksi käyn läpi, kuinka aloitin projektin luomisen, mitä ongelmia tuli ratkaista ja miten projektin toteutus muuttui.

Pääkohdat, jotka halusin ratkaista projektissa:

1. Siinä tulisi olla helppokäyttöinen käyttöliittymä.
2. Se olisi helposti asennettavissa.
3. Sen tulisi olla joustava ja toimisi käytännössä kaikissa veneissä. Tämä riippuu täysin veneen ajurimoduulista.

Seuraavaksi kerron, miten aloitin AutoSailor-projektin rakentamisen. Prototyyppivaiheessa kävin läpi mahdolliset toteuttamismenetelmät, jotka olivat itselle jonkin verran tuttuja entuudestaan. On syytä huomioida, että projektin toteuttamismenetelmä, tavat, joilla lähestyin eri ongelmakohtia, muuttuivat myöhemmin projektin edetessä pidemmälle.

Ensimmäisen kohdan ratkaisu alkoi ajatuksesta, että verkkopohjainen käyttöliittymä jaettaisiin lähiverkkoon esimerkiksi kannettavan tietokoneen kautta. Tietokone toimisi sekä verkkosivun palvelimena sekä MQTT-Brokerina.

Ajatus siitä, että käyttöliittymään pääsisi käsiksi millä tahansa laitteella, kiehtoi. Se tekisi koko järjestelmästä hyvin yksinkertaisen ilman ylimääräisiä näyttöjä, sillä kaikillahan on jo yksi sellainen taskussa.

Verkkosivu mahdollistaisi myös sen, että käyttäjä voisi piirtää navigointireitit, näkisi veneen paikan, suunnan ja nopeustiedot sekä pystyisi ohjaaman ve-

neen autopilotin parametrejä nettisivulta käsin. Voisin myös hyödyntää valmiita JavaScript-kirjastoja, jolla saisin käyttäjäkokemuksesta helpon sekä sitä olisi mukava katsella.

Kommunikaatio päätelaitteiden välillä tapahtuisi lähiverkossa MQTT-viestein, ja verkkosivun avannut laite voisi lähettää ja vastaanottaa viestejä, jotka palvelin prosessoisi navigaatiodataksi ja lähettäisi ne eteenpäin ajurimoduuliin. Kokeilin edellä mainittua ratkaisua ensimmäisenä, mutta pian huomasin sen ongelmakohdat.

Seuraavaksi tuli ratkaista se, kuinka saisin navigaatiotiedot, kuten GPS-datan ja suuntatiedon. Kuvittelin, että GPS-datan olisin saanut elegantisti puhelimen kautta, mikäli käyttäjä olisi avannut verkkosivun puhelimella, sillä nykyisin jokaisella on GPS-paikannin omassa taskussa. Pöytä- tai kannettavalla tietokoneella tarkkaa GPS-dataa ei voi saada ilman ulkoisia komponentteja, jotka ovat tähän tarkoitukseen tehty. Nopeasti kävi ilmi, että puhelimen GPS-tietojen pyytäminen verkkosivun kautta vaatisi turvallisen yhteyden verkkosivulle, eli verkkosivulla täytyi olla SSL-sertifikaatti.

Lyhyesti SSL-varmenne on digitaalinen varmenne, joka todentaa verkkosivuston ja mahdollistaa salatun yhteyden. SSL tulee sanoista ”Secure Sockets Layer”, joka tarkoittaa suojausprotokollaa, joka luo salatun yhteyden verkkopalvelimen ja selaimen välille. (Kaspersky 2023.) Toisin sanoen: SSL suojaa Internet-yhteydet ja estää rikollisia lukemasta tai muokkaamasta kahden järjestelmän välillä siirrettäviä tietoja.

SSL-sertifikaattiongelma söi aikaa. Olisi ollut mahdollista jakaa sovellus Web-Hotellin kautta, jolloin se olisi sisältänyt SSL-varmenteen. Mutta koska halusin tehdä projektin ilman ulkoisia tekijöitä, parempi toteuttamismenetelmä oli löydettävä.

Mikäli olisin toteuttanut projektin käyttämällä puhelinta sensoripakettina, navigoinnin aikana puhelinta ei tulisi myöskään liikuttaa, jos haluaisi siitä myös magnetometritiedot suuntatietoja varten. Tämä ei siis ollut oikea reitti, mutta ”väärän” lähestymistavan myötä opin verkkosivun rakentamista ja ongelmien ratkaisemista eri menetelmillä.

Lopulta valitsin Raspberry Pi:n käyttöön AutoSailor-projektissa, sillä se on edullinen ja tehokas tietokone. Se mahdollistaa, että koko navigointiyksikkö on käytännössä samassa paketissa. Lisäksi Raspberry Pi sisältää monia liitäntöjä erilaisten antureiden ja laitteiden kytkemiseen, kuten GPIO-pinnit ja I2C-väylän. Lisäksi Raspberry Pi:llä on mahdollista käyttää erilaisia käyttöjärjestelmiä, kuten Raspbian. Tämä käyttöjärjestelmä on suunniteltu erityisesti Raspberry Pi:lle ja sisältää valmiiksi monia tarvittavia ohjelmistoja. Raspberry Pi:llä on myös mahdollista käyttää Python-ohjelmointikieltä, joka on suosittu valinta monissa IoT-sovelluksissa.

3 AUTOSAILOR-JÄRJESTELMÄ

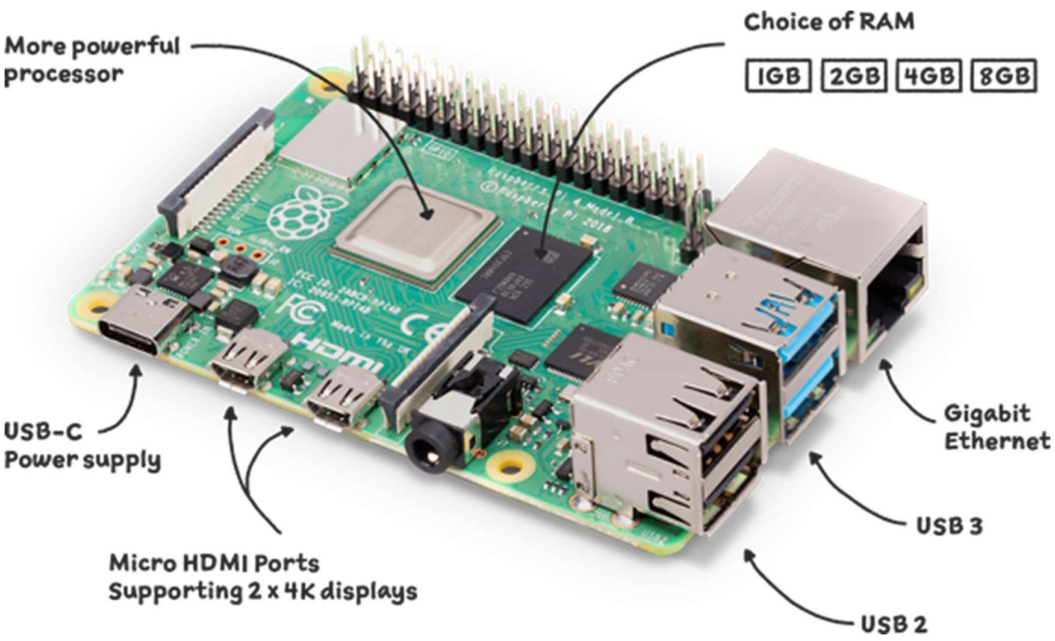
AutoSailor-sovelluksen prosessointiyksikkönä toimii Raspberry Pi, joka on yhden piirilevyn tietokone. Opinnäytetyön sovelluksessa tietokonetta käytetään eräänlaisena navigaattorina, joka jakaa lähiverkkoon navigaatiokäyttöliittymän. Raspberry Pi lukee ja prosessoi dataa ulkoisista komponenteista ja käyttää niitä hyväksi navigoinnissa.

Tietokoneen toiminta voidaan tässä tapauksessa jakaa kahteen eri osaan, Front-End sekä Back-End toimintaan.

- Lyhyesti Front-End-toiminnalla viitataan kaikkeen siihen toimintaan, mikä on käyttäjälle näkyvää.

- Toisin kuin Front-End-toimintaa, Back-End-toimintaa käyttäjä ei näe. Siellä kaikki ovat prosesseja, jotka ovat käyttäjälle näkymättömiä. Back-End käsittelee tiedon, jota Front-Endissä näytetään.

Raspberry Pi-4 ja sen ominaisuudet esitetty kuvassa 1.



Kuva 1. Raspberry Pi-4 (Raspberry Pi Foundation)

3.1 AutoSailor Front-End

AutoSailor-sovelluksen käyttöliittymän käyttäjäystävällisyyden ja monipuolisuuden lisäämiseksi suunnittelin sen toimimaan kaikissa yleisissä verkkoselaimissa, joita on saatavilla puhelimille, tableteille ja tietokoneille. Käyttöliittymä mahdollistaa veneen autopilotin helpon hallinnan lähiverkon kautta, mikä tekee AutoSailorista erittäin joustavan ja käyttäjäystävällisen.

Syy, miksi päädyin verkkopohjaiseen käyttöliittymään, on seuraava. Erillisten puhelinsovellusten ja tietokonesovellusten rakentaminen olisi ollut hyvin työlästä, sekä se olisi syönyt loputtomasti aikaa. Sitä ei olisi ollut tässä projektissa millään tavalla järkevä toteuttaa, eikä se olisi tarjonnut merkittäviä etuja. Siksi koenkin, että verkkopohjaiset käyttöliittymät ovat tulevaisuutta.

Raspberry Pi:n käyttö verkkosovelluksen jakamiseen mahdollistaa sen, että käyttäjä voi hallita autopilotin toimintaa lähiverkon kautta, ilman että tarvitsee asentaa sovellusta omalle laitteelleen. Tämä tekee AutoSailorista erittäin helppokäyttöisen ja joustavan. Verkkoselainpohjaisessa käyttöliittymässä on etuna sen laitteistoriippumattomuus, mikä tarkoittaa, että sovellusta on mahdollista käyttää millä tahansa laitteella, jolla voi käyttää verkkoselainta.

AutoSailor-sovelluksen käyttöliittymä tarjoaa reaaliaikaisen seurannan veneen sijainnista, nopeudesta ja suunnasta. Käyttäjä voi helposti asettaa autopilotin halutulle kurssille suoraan kartalta osoittamalla. Autopilotin herkkyyden säätäminen ja muiden asetusten muuttaminen on myös helppoa käyttöliittymän kautta.

Yhteenvetona voidaan todeta, että AutoSailor-verkkosovellus on käyttäjäystävällinen ja helppokäyttöinen veneen autopilotin ohjaussovellus. Sen käyttöliittymä on suunniteltu toimimaan kaikilla laitteilla, joissa on verkkoselain. Sovellus tarjoaa reaaliaikaisen seurannan veneen sijainnista ja mahdollistaa helpon autopilotin hallinnan lähiverkon kautta. Näiden ominaisuuksien ansiosta AutoSailor tekee käyttäjäkokemuksesta vaivattoman.

3.2 AutoSailor Back-End

AutoSailor-sovelluksessa ohjaus, navigointi ja tiedonsiirto tapahtuvat kaikki Back-Endin kautta. Käsitteenä Back-End on ohjelmisto tai palvelu, joka vastaanottaa, jäsentää ja käsittelee kaiken datan, ja se on käyttäjälle näkymätöntä toimintaa. AutoSailorin Back-End-palvelin sisältää esimerkiksi veneen sijaintiin, nopeuteen ja suuntaan liittyviä tietoja. Back-End ohjaa venettä Front-Endin määrittämän navigointireitin mukaisesti ja ohjaa sen kulkemaan pitkin reittiä tai asetettua päämäärää kohti.

Autosailorin Back-End on rakennettu Python-moduuleista, jotka toimivat yhtenä kokonaisuutena. Yksi esimerkki tällaisesta moduulista on ”Wayfinder”, joka on nimensä mukaisesti reitin etsijä. Moduuli auttaa navigoinnissa laskemalla reittipisteiden välisen matematiikan, kuten matkan ja kulman, sekä interpoloi virtuaalisen navigointipisteen, jota vene yrittää tavoittaa. Tämä korjaa tuulen aiheuttamaa suistumista ja auttaa veneen pysymään oikealla kurssilla.

Lisäksi Back-Endissä on Python-moduuleja, joita käytetään navigoinnin ja tiedonsiirron hallintaan. Esimerkiksi yksi tärkeä moduuli on ”ausanavi”, joka vastaanottaa GPS-signaaleja veneen GPS-moduulista sekä hakee kompassitiedon ulkoisesta Sense HAT-Moduulista ja muuntaa ne käyttökelpoiseksi navigointi-informaatioksi.

Back-Endin moduulit ovat suunniteltu siten, että niitä voidaan käyttää yhdessä tai erikseen muiden järjestelmien kanssa, sekä moduuleja voidaan rakentaa jokaiseen tarpeeseen. Tämä mahdollistaa sen, että AutoSailoria voidaan haluttaessa laajentaa ja käyttää yhdessä muiden veneen elektronisten järjestelmien kanssa, kuten esimerkiksi sääaseman, puhelimen tai tabletin kanssa.

Back-Endin tärkein rooli AutoSailorin järjestelmässä on kuitenkin navigoinnin ohjaaminen. Kun käyttäjä on määrittänyt halutun navigointireitin Front-Endissä, Back-End ohjaa veneen kulkemaan pitkin reittiä ja ohjaa sitä kohti määränpäätä. Kokonaisuutena AutoSailorin Back-End on siis tärkein osa järjestelmää, sillä se mahdollistaa veneen navigoinnin.

Back-Endin rooli AutoSailorissa ei kuitenkaan rajoitu pelkästään navigoinnin ohjaukseen. Se vastaa myös tiedonsiirrosta ja kommunikoinnista muiden järjestelmien kanssa. Kokonaisuudessaan Back-End on keskeinen osa AutoSailorin järjestelmää, joka vastaa kaikista toiminnoista, jotka eivät ole suoraan käyttäjän nähtävissä tai käyttämässä navigointikäyttöliittymää.

Yhteenvetona Back-End vastaanottaa ja käsittelee kaiken datan, mukaan lukien veneen sijaintiin, nopeuteen ja suuntaan liittyvät tiedot. Back-Endin moduuleja käytetään navigoinnin ja tiedonsiirron hallintaan, esimerkiksi GPS-signaalien vastaanottoon ja käyttökelpoisen navigointi-informaation muuntamiseen. Back-End on keskeinen osa AutoSailorin järjestelmää, joka sitoo kaikki järjestelmät yhteen.

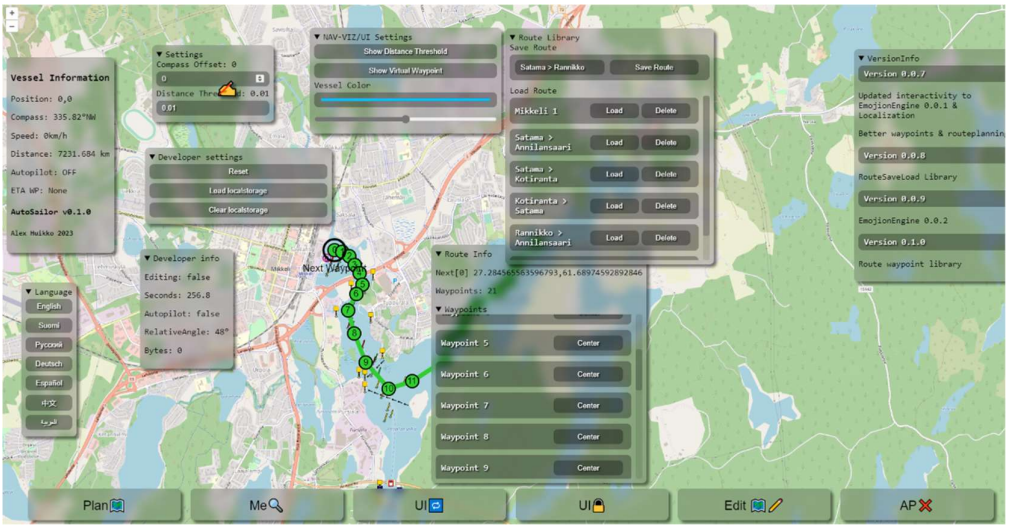
4 AUTOSAILORIN TOIMINTA

Tässä kappaleessa käydään läpi, kuinka AutoSailor toimii, mitä se pitää sisällään ja mitä teknologioita järjestelmä käyttää.

4.1 Verkkopohjainen käyttöliittymä

Aloitin opinnäytetyön projektivaiheen luomalla sovellukselle verkkosivun, joka mahdollistaa intuitiivisen käyttäjäkokemuksen. Verkkosivu on rakennettu käyttäen HTML-, CSS- ja JavaScript-, merkkaus- ja ohjelmointikieliä.

Alhaalla esitetyssä kuvassa 2 näkyy Autopilotin käyttöliittymä.



Kuva 2. AutoSailor UI

Verkkosivulla vasemmassa yläkulmassa on näkyvissä useita tärkeitä tietoja veneestä järjestyksessä.

- Koordinaatit
- Kompassidata, asteina
- Navigoinnin tila
- Odotettu aika seuraavalle reittipisteelle.

Käyttöliittymästä olen tehnyt täysin muokattavan siirtoikkunamallisen ratkaisun, joka mahdollistaa jokaisen näkyvän ikkunan liikuteltavuuden. Kun käyttäjä on liikuttanut ikkunan haluttuun kohtaan, positio tallennetaan verkkoselaimen muistiin.

”Developer Info” kertoo käyttäjälle normaalisti näkymättömiä tietoja, kuten editoidaanko reittiä juuri, onko autopilottistatus kytkettynä päälle ja kuinka pitkään sovellus on ollut päällä. Se kertoo myös mitä dataa autopilotti lähettää ajurilaitteelle. Nettisivu koostuu liikuteltavista palikoista, jotka pitävät sisällään navigointitietoja, joten käyttäjä voi itse rakentaa sivusta mieluisen.

Verkkosovellus tarjoaa käyttäjälle graafisen käyttöliittymän, jossa voi tarkastella veneen sijaintia kartalla ja navigointitietoja reitin varrella. Käyttäjä voi määrittää navigaatioreitin piirtämällä kartalle pisteitä. Verkkosivu lähettää piirretyn reitin Back-Endin muistiin ja samalla hakee uudestaan tuon äsken piirretyn reitin, tällöin se päivittyy jokaiselle auki olevalle verkkosivulle.

4.1.1 OpenLayers

Valitsin projektissa käytettäväksi karttatietorajapinnaksi OpenLayers JavaScript-kirjaston sen helppokäyttöisyyden ja ominaisuuksien vuoksi. Kirjastolla on laaja valikoima ominaisuuksia, kuten erilaisia karttatasojen esittämistapoja, interaktiivisia kontrolleja, kuten zoomaus ja kiertäminen, sekä työkaluja tietojen muokkaamiseen ja piirtämiseen. OpenLayers tarjoaa myös mahdollisuuden yhdistää karttatiedot erilaisiin tietolähteisiin, kuten paikkatietokantoihin ja tiedostoihin.

OpenLayersin käyttöönotto on suhteellisen helppoa, sillä se tarjoaa monia esimerkkejä ja dokumentaatiota sen käyttämisestä. OpenStreetMap piirtää maailmankartan, jonka päälle piirretään OpenSeaMap-merikartta. Merikartassa näkyy muun muassa, matalikot, satamat, poimutiedot, majakat sekä erilaiset viitat. Tässä projektissa se mahdollistaa navigaatiodatan luomisen interaktiivisesti Front-Endissä, sekä samassa kartassa voidaan näyttää Back-Endissä prosessoitu navigointidata käyttäjälle, kuten veneen sijainti, suunta ja kuljettava navigaatioreitti.

4.1.2 Lokalisointi

Päätin tehdä verkkosivun käyttöliittymän käytettäväksi eri kielillä, sillä se oli suhteellisen helppo toteuttaa. Käytännössä kaikki sanat, jota nettisivu näyttää, haetaan (kuva 3) Localization.js-tiedostosta.

```
export const localization_dictionary = {
    "en" : {
      "position" : "Position",
      "compass" : "Compass",
      "speed" : "Speed",
      "distance" : "Distance",
      "editing" : "Editing",
      "autopilot" : "Autopilot",
      "route_info" : "RouteInfo",
      "settings" : "Settings",
      "relative_angle" : "RelativeAngle",
      "waypoints" : "Waypoints",
      "seconds" : "Seconds",
      "vessel_info" : "Vessel Information",
      "waypoint_index" : "Next",
      "distance_threshold" : "Distance Threshold",
      "compass_offset" : "Compass Offset",
      "dev_info" : "Developer info",
      "dev_settings" : "Developer settings",
      "language" : "Language",
      "virtual_waypoint" : "Virtual Waypoint",
      "next_waypoint" : "Next Waypoint",
      "eta" : "Waypoint arrival",
      "kmh" : "km/h",
      "hour" : "h",
      "plan" : "Plan",
      "me" : "Me",
      "ui_reset" : "UI Reset",
      "ui_lock" : "UI",
      "show" : "Show",
      "hide" : "Hide",


    },
    "fi" : {
      "position" : "Sijainti",
      "compass" : "Kompassi",
      "speed" : "Nopeus",
      "editing" : "Editointi",
      "distance" : "Etäisyys",
      "autopilot" : "Autopilotti",
      "route_info" : "Reittitiedot",
      "settings" : "Asetukset",
      "relative_angle" : "Suhteellinen kulma",
      "waypoints" : "Reittipisteet",
      "seconds" : "Sekunnit",
```

Kuva 3. Lokalisointikirjasto

Verkkosivun koodissa kirjastosta voidaan hakea sanoja, joita sivulla käytetään, ja sanakirjasto vaihtuu automaattisesti aina uusi kieli valittaessa. Tällä hetkellä verkkosivu tukee englantia, suomea, saksaa, espanjaa, venäjää, arabiaa sekä kiinaa (yksinkert.)

4.1.3 EmojionEngine

JavaScript-kirjaston nimi, EmojionEngine-on sanaleikki PS2-konsolissa käytettävästä EmotionEngine-suorittimesta. Kirjasto käyttää jokaisen alustan omia emojeja käyttöliittymässä.

Halusin tehdä verkkopohjaisesta käyttöliittymästä persoonallisen, ja mielestäni onnistuin tässä, sillä, EmojionEngine on silmäkarkkia käyttäjälle ja, niin kuin nimestä voi päätellä, se käyttää emojien visuaalista viehätysvoimaa tarjoamalla myös hauskan käyttäjäkokemuksen. Se lisää myös käyttöliittymän esteettistä arvoa.

Rakensin verkkosivulle ”EmojionEngine”-nimisen JavaScript-kirjaston. Se on eräänlainen interaktiomoottori, joka muuttaa käyttäjän kursorin ja verkkosivun käyttöliittymän elementtejä interaktiivisella tavalla. Moduuli on helppo lisätä nettisivulle, -se ottaa huomioon verkkosivulla esiintyvät elementit ja muuntaa kursorin erilaisilla tavoilla. Käyttäjän tarvitsee vain merkata halutut napit ”emojion-element”-luokalla, joka tekee napeista interaktiivisen tuntuiset. Erilaiset kursorit määritetään EmojionEnginen sisäisessä cursors-moduulissa, joka sisältää useita valmiita kursorimuotoja. Käyttäjä voi myös lisätä omia kursorimuotoja cursors-moduuliin, jotka on helppo hakea pääohjelmassa.

EmojionEngine toimii automaattisesti, mutta käyttäjä voi kutsua luokan metodeja manuaalisesti. Moottorin voi kytkeä päälle sekä pois vaihtamalla ”emojionEngineEnabled”- arvoa true:n ja false:n välillä.

```
import EmojionEngine from './js/modules/emojionengine/emojionengine.js';

var emojionEngineEnabled = true;
var emojionEngine = null;

if(emojionEngineEnabled)
{
  const eeOptions = {
    cursorSize : "30px",
    color : "#BADA55",
  }
  emojionEngine = new EmojionEngine(eeOptions);
}else{
  document.documentElement.style.setProperty('--cursor_state', "visible");
}
```

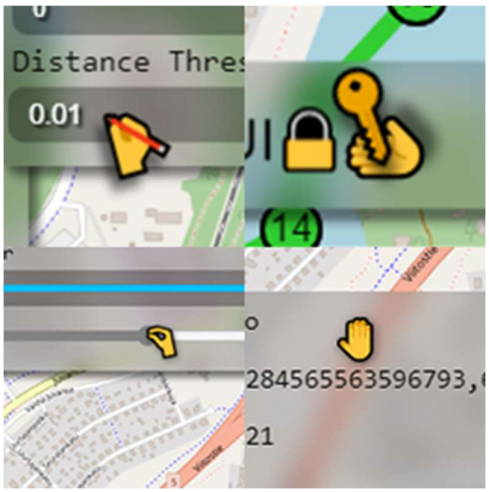
Kuva 4. EmojionEnginen alustus verkkosivulle

Kuvassa 4 näkyy-kuinka EmojionEngine-olio rakennetaan ja options-tiedostot ajetaan rakennusmetodiin (new EmojionEngine (options)) sisälle. Tämä on käytännössä ainut asia-mitä käyttäjän tarvitsee tehdä interaktiomoottorin käynnistämiseen, ohjelma itsessään hoitaa loput. EmojionEngine mahdollistaa

myös pakotusfunktion käytön, jos käyttäjä haluaa pakottaa kursorin muuntumisen missä tahansa tilanteessa.

Käyttäjä voi kutsua kursorin muuttavan metodin ajamalla seuraavan komennon, ”emojionEngine.setEmojionCursor(add, remove)”. Komento lisää kursorin luokkiin ”add” parametrikentässä määritetyt mutaatioluokat ja poistaa kursorista ”remove” parametrikentässä määritetyt mutaatioluokat.

Mutaatioluokilla tarkoitan sitä, että kursori muuttuu luokkien perusteella, esim. jos kursorin luokkiin lisätään (”editing”), kursori muuttuu kynää pitäväksi kädeksi ( ), tai jos luokkiin lisätään (”pinch”), kursori muuttuu puristavaksi sormiksi ( ), jota käytetään mm. liukusäätimiin. Kun kursori on epäaktiivinen, eli omaa (”inactive”) luokan, kursori näyttää osoittavalta sormelta ( ). Alla olevasta kuvasta voi nähdä esityksen noista interaktioista eri elementeissä.



Kuva 5. Interaktiivinen kursori.

EmojionEnginen toiminta perustuu sen sisäiseen logiikkaan ja valmiiksi määritettyihin luokkiin, jonka avulla se muuttaa kursorin ulkonäköä. EmojionEngine monitoroi kursorin luokkamuutoksia ja aktivoi ”changeEmojionCursor”-metodin aina, kun luokkamuutos tapahtuu. Tämän takia käyttäjän on helppo lisätä interaktiomoottori omiin sovelluksiin. Rakensin EmojionEnginen siksi, koska halusin luoda visuaalisesti miellyttävän kursorin ja sen dynaamisen käyttäytymisen eri elementeissä.

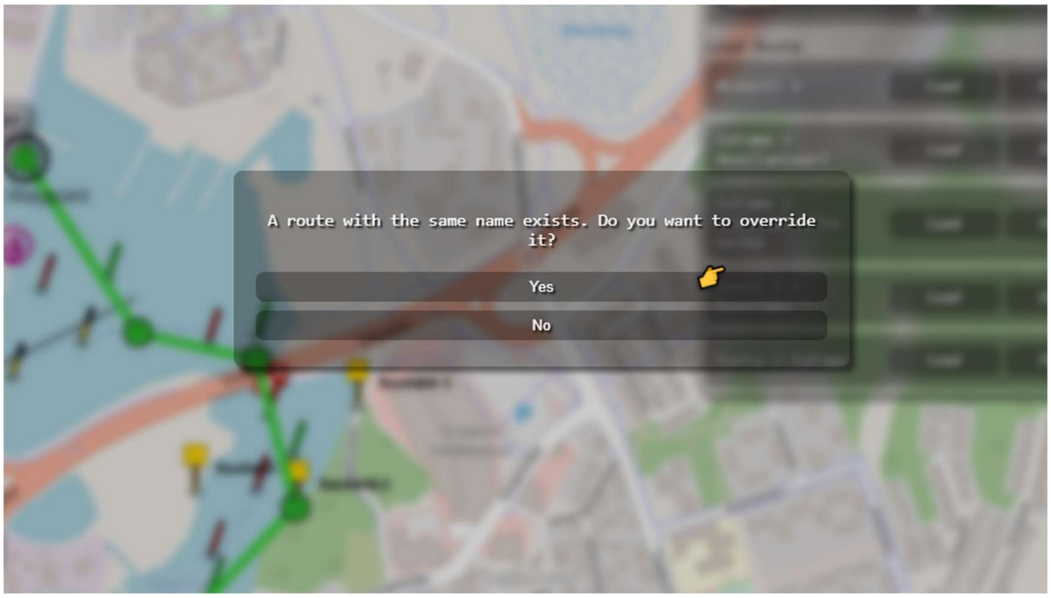
4.1.4 Reittikirjasto

AutoSailorissa on mahdollista tallentaa reitit reittikirjastoon myöhempää käyttöä varten, jotta reittejä ei tarvitse piirtää uudestaan. Jokainen reitti tallennetaan verkkosivun localstorage muistiin, jonka jälkeen reitti on helppo hakea uudestaan. Reitin voi myös poistaa muistista tarpeen vaatiessa.

Kuva 6. Reittikirjasto.

Kun reitti tallennetaan antamalle sille nimi ja painamalla Save Route -näppäintä, reitti lisätään localstorage-säiliöön ja alla oleva Route Load -säiliö populoidaan kaikilla localstoragessa olevilla reittiobjekteilla. Kun ”Load”-näppäintä painetaan, verkkosivu lähettää MQTT-viestin, joka sisältää haetun reittidatan. Kun tämä on tehty, ladattu reitti ajetaan Back-Endin muistiin ja verkkosivu vastaanottaa päivitetyn tiedon, jonka jälkeen reitti päivittyy reittipiirtosyklin mukana.

Jos käyttäjä yrittää lisätä samannimistä reittiä reittikirjastoon, verkkosivu avaa hyväksymisikkunan ja kysyy, haluatko varmasti yliajaa vanhan reitin uudestaan. Alla olevassa kuvassa 7 näkyy hyväksymisikkuna.



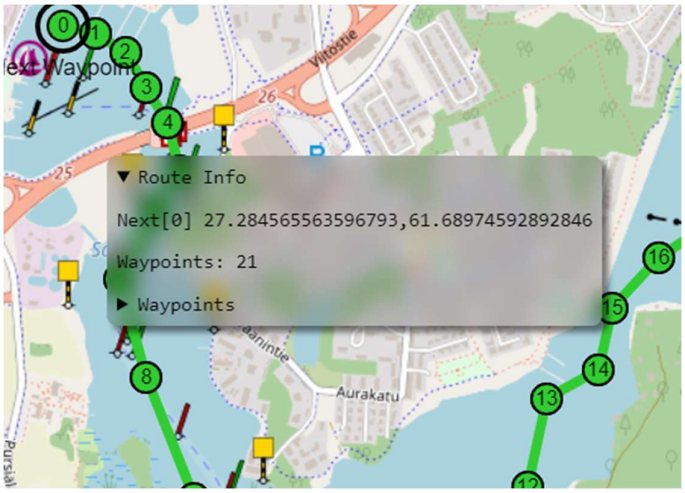
Kuva 7. Hyväksymisikkuna

Käyttäjä voi valita joko kyllä tai ei. Jos käyttäjä valitsee kyllä, uusi reitti asetetaan vanhan reitin nimen päälle ja vanha reitti poistetaan kokonaan. Mutta jos käyttäjä vastaa ei, reittikirjastoa ei muuteta ollenkaan ja verkkosivu palautuu normaaliin näkymään.

4.1.5 Reittipistenäkymä

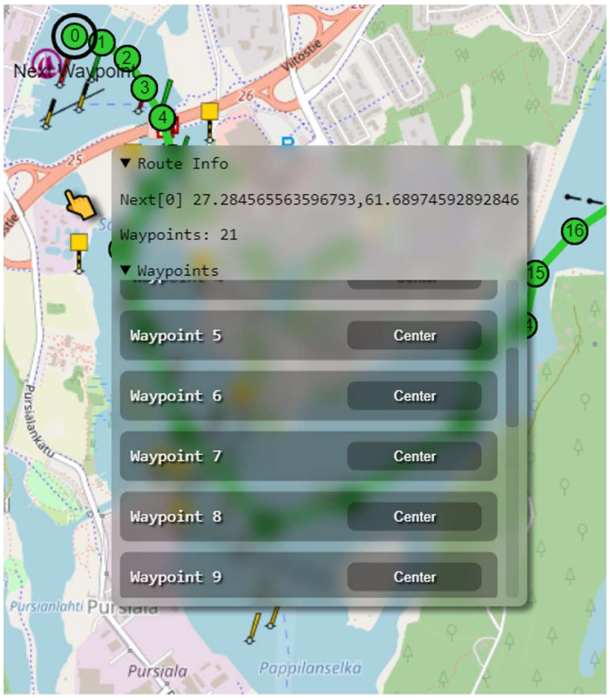
Reittipistekirjasto oli helppo toteuttaa käyttämällä hyväksi valmiina olevaa kirjastorakennetta, joka käytiin läpi edellisessä kappaleessa. Reittipistenäkymä mahdollistaa kaikkien reittipisteiden seuraamisen, ja sitä kautta on helppo hypätä mihin tahansa reittipisteeseen.

Alla olevassa kuvassa 8 näkyy reitin tiedot ja ”waypoints”-alihakemisto, joka on kiinni.



Kuva 8. Reittitieto-ikkuna

Käyttäjä voi avata reittipistehakemiston painamalla ”Waypoints”-kohtaa, jonka jälkeen käyttäjää kohtaa seuraavanlainen näkymä. Jokaisesta reittipisteestä on luotu oma elementti, joka sisältää koordinaattitiedot. (Kuva 9.)



Kuva 9. Reittitieto-ikkunan reittipiste-kirjasto avattuna

Jokainen reittipiste on kirjattu hakemistoon, ja painamalla ”Center”-painiketta ohjelma lukee elementtiin tallennetun koordinaattitiedon ja karttanäkymä kohdistuu haluttuun pisteeseen.

4.2 Kaksisuuntainen MQTT-kommunikaatio

Kommunikaation toteuttaminen verkon kautta millä tahansa laitteella on olennainen osa monia nykyaikaisia sovelluksia. AutoSailor hyödyntää MQTT-verkkokommunikaatioprotokollaa tiedonvälitykseen päätelaitteiden välillä. Tämän protokollan käyttäminen edellyttää WebSocket-yhteyden muodostamista verkkosivun ja MQTT-viestien välillä.

Raspberry Pi toimii MQTT-Brokerina, joka jakaa viestit lähiverkossa oleville laitteille. Broker-toiminto tarkoittaa käytännössä sitä, että Raspberry Pi vastaanottaa viestejä ja välittää ne eteenpäin oikeille vastaanottajille. Viesteihin pääsee käsiksi kirjautumalla oikeaan IP-osoitteeseen ja antamalla tarvittavat kirjautumistiedot, jotka on määritetty Brokerina toimivan laitteen asetuksissa.

Viestit lähetetään Brokerille ”Topic”-lokeroihin, jotka ovat viestien aihealueita. Kun viesti halutaan lähettää, täytyy sille määrittää aihe sekä aiheen viesti. Viesti voi sisältää mitä tahansa tietoa, ja sen koko voi vaihdella kahdesta tavusta aina 256 megatavuun tarpeen mukaan. MQTT Client voi tilata viestilokeroita, jonka jälkeen se voi vastaanottaa viestejä näistä tilatuista lokeroista.

Kuvassa 3 näkyy Python-ohjelmointikielessä vakioiksi määritetyt aihelokerot, joita voimme käyttää myöhemmin ohjelmassa. Kuvassa näkyy myös, kuinka MQTT:n viestilokerot on järjestetty aihehierarkisesti. MQTT-viestien ”lokerot” ovat vakioita, eivätkä ne siis koskaan muutu, ainoastaan niiden sisältämä tieto muuttuu.

```
TOPIC_NAVIGATION = "AUSA/BACK/navigation"
TOPIC_STATES     = "AUSA/BACK/states"
```

Kuva 10. MQTT Topic -vakiot

AutoSailorin lähettämät viestit ovat kooltaan kahdestakymmenestä tavusta, yhteen kilotavuun. Käytännössä se vastaa pienen tekstitiedoston kokoa. Kommunikaatiossa käytetyt viestit ovat kooltaan siis hyvin pieniä, sillä ne sisältävät vain navigaatiodataa.

```
{
  "positions": {
    "start": {
      "lon": 27.2670582,
      "lat": 61.6915256
    },
    "current": {
      "lon": 27.2670766,
      "lat": 61.691516
    },
    "next": {
      "lon": 27.265878193902132,
      "lat": 61.692290506474734
    }
  },
  "heading": 327.08,
  "relative_angle": -24.206202615521057,
  "distance_to_next": 0.107,
  "waypoint_eta": 2.3750332948592736,
  "speed": 0.045052,
  "waypoints": [
    [
      27.265878193902132,
      61.692290506474734
    ],
    [
      27.26114640862606,
      61.695373199280226
    ],
    [
      27.277975105825305,
      61.699177363948905
    ],
    [
      27.262298495475886,
      61.703195613075906
    ]
  ],
  "current_waypoint_index": 0,
  "final_waypoint_index": 3
}
```


Kuva 11. Esimerkki projektin navigointidatan JSON-datapaketista

Kuvassa 11 on esitetty prosessoitu navigointidata, jota kuvassa 3 esitetty lokero ”AUSA/BACK/navigation” pitää sisällään. Back-End lähettää lokeroon tämän datapaketin reaaliaikaisesti, noin 500 millisekunnin välein.

Seuraavaksi käydään läpi, mitä datapaketin kentät kuvaavat ja mihin niitä käytetään.

"positions" sisältää sijaintitiedot kolmessa eri muodossa:

- "start" kuvaa lähtösijaintia koordinaatteina (longitude, latitude).
- "current" kuvaa veneen nykyistä sijaintia koordinaatteina.

- "next" kuvaa seuraavan reittipisteen sijaintia.

"virtual" sisältää virtuaalisen sijainnin, jota käytetään tuulen suiston kompensointia varten, kordinaatti lasketaan interpoloimalla ”start”- ja ”next”-koordinaatin välillä.

"heading" kuvaa veneen suuntaa asteina.

"relative_angle" kuvaa kulmaa veneen ja seuraavan reittipisteen välillä, joka avulla venettä käännetään kohti seuraavaa pistettä.

"distance_to_next" kuvaa etäisyyttä kohteen nykyisestä sijainnista seuraavaan reittipisteeseen.

"waypoint_eta" kuvaa ennustettua aikaa, joka tarvitaan seuraavan reittipisteen saavuttamiseen.

"speed" sisältää veneen nopeuden.

"waypoints" sisältää taulukon koordinaatteja kaikista reittipisteistä.

"current_waypoint_index" kuvaa nykyisen reittipisteen indeksiä "waypoints"-taulukossa.

"final_waypoint_index" kuvaa viimeisen reittipisteen indeksiä "waypoints"-taulukossa.

Kommunikaatio on kaksisuuntaista, mikä tarkoittaa sitä, että kaikki käyttäjän tekemät toiminnot heijastuvat kaikkiin laitteisiin. Tämä johtuu siitä, että jokainen käyttäjän toiminto lähettää signaalin takapäähän (Back-Endiin), mikä tarkoittaa, että muutokset näkyvät kaikilla laitteilla samanaikaisesti. Tämä tarkoittaa myös sitä, että mitään tärkeitä tiloja ei tallenneta selaimen muistiin.

Käyttäjän ja laitteen välillä tapahtuva kommunikaatio mahdollistaa sen, että kaikki käyttäjän tekemät muutokset välittyvät nopeasti kaikille osapuolille. Tämä lisää tehokkuutta ja vähentää mahdollisia virheitä, kun kaikki näkevät samat tiedot ja statukset reaaliajassa.

Kaksisuuntaisen kommunikaation lisäksi olisi myös tärkeää, että ohjelmassa käytettävät tiedot pysyvät turvassa ja salassa. Koska AutoSailor ei pidä sisällään arkaluontoista tietoa, en projektissa ottanut tietoturvaa huomioon ensinnäkään siksi, että järjestelmä toimi opinnäytetyön ajankohtana ainoastaan lähiverkossa, ja myös siksi, koska en halunnut kuluttaa aikaa tietoturvaratkaisujen kehitykseen. Tosin huomioitavaa on se, että jos järjestelmä olisi kaupallinen, tämä tietoturvaongelma tulisi ratkaista.

4.3 Python-moduulit

AutoSailor Back-End -ohjelmisto on rakennettu Python-moduuleista, jotka toimivat yhtenä kokonaisuutena. Syynä tähän on se, että halusin tehdä ohjelmistosta laajennettavan ja sen ohjelmointirajapinnasta helppokäyttöisen sekä yksinkertaisen.

Tässä on listaus, minkälaisia moduuleja AutoSailor pitää sisällään, mitkä yhdistetään ”main.py”-ohjelmassa.

- **wayfinder.py**

  Moduuli auttaa navigoinnissa laskemalla reittipisteiden välisen matematiikan, kuten matkan ja kulman, sekä interpoloi virtuaalisen navigointipisteen, jota vene yrittää tavoittaa.

- **ausanavi.py**

  Hoitaa GPS- sekä kompassitiedon hakemisen sekä tallentaa ne järjestelmään myöhempää käyttöä varten.

- **navimath.py**

  Kaikki navigaatiossa käytetyt matemaattiset funktiot ovat rakennettuna täällä, näin jokainen moduuli voi hyödyntää niitä, eikä niitä tarvitse rakentaa monta kertaa uudestaan.

- **ausamqtt.py**

  Tämä moduuli sisältää topic-vakiot ja mahdollistaa MQTT-yhteyden, jonka avulla moduulit voivat lähettää ja vastaanottaa viestejä MQTT brokerilta.

- **ausadb.py**

  Lyhenne sanoista AutoSailor Database, tämä moduuli lukee ”settings.ini”-asetustiedostot ja tallentaa ne järjestelmään. Moduulin arvoja käytetään joka puolella ohjelmistoa.

Jokainen moduuli on rakennettu niin että, jos moduuli ei jostain syystä toimi tai ohjelmaan tulee jokin virhe, se ei keskeytä koko pääohjelmaa vaan ilmoittaa konsoliin virheestä ja jatkaa normaalia prosessia. Tämä mahdollistaa sen, että pääohjelma ei kaadu helposti vaan voi toimia ilman virhettä aiheuttavaa moduulia.

4.4 Sijainti- ja suuntatieto

Seuraava ongelma oli tärkein ja suuri askel projektin etenemisessä. Minun tuli ratkaista, kuinka saisin reaaliaikaisen sijainti- ja suuntatiedon, jota voisin hyödyntää navigoinnissa.

Aloin tutkia erilaisia vaihtoehtoja ja ratkaisuja, jotka voisin ottaa käyttöön projektissani. Tutkin erilaisia sensoreita, antureita ja laitteita, joita voisin käyttää sijainnin ja suunnan määrittämiseen.

Nopeasti projektin alussa oli selvää, että projekti vaatisi ainakin kaksi ulkoista laitetta, jotta saisin navigaatioon tarvittavan datan. Tarvitsisin laitteen, jolla voisin hyödyntää GPS-satelliitteja, sekä laitteen, jolla pystyisin tietämään veneen suunnan.

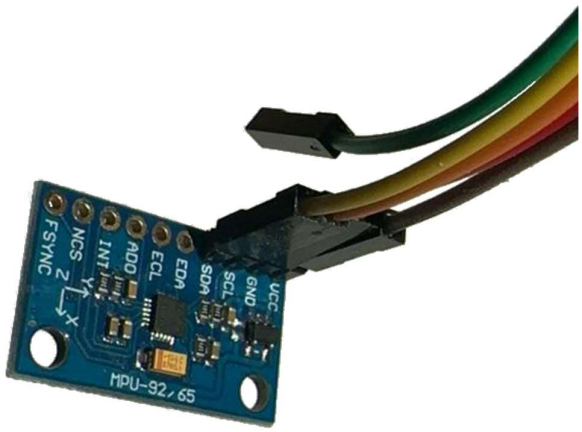
4.4.1 Sijainti

Sijaintiongelman ratkaisu lähti käyntiin ajatuksesta, että matkapuhelimen GPS-dataa voisi käyttää hyväksi. Päivän tutkimusten jälkeen selvisi, että tämä tapa ei ollut toteutuskelpoinen minun taidoillani. Puhelimesta voisi kyllä saada ulos GPS- ja suuntadataa, mutta puhelimen tulisi olla paikoillaan navigoinnin aikana. Päädyinkin lopulta käyttämään itsenäistä GPS-vastaanotinta, joka on kytkettävissä suoraan laitteeseen.

GPS-vastaanotin oli lopulta ratkaisu sijaintiongelmaani. Se oli helppo asentaa, data oli helppo saada ulos ja sitä oli helppo käyttää, ja sen avulla pystyin tarkasti määrittämään laitteen sijainnin. Vastaanotin kerää jatkuvasti tietoa laitteen sijainnista ja välittää sen eteenpäin sovellukselle, jota pystyn käyttämään myöhemmin navigaatiossa. Veneen sijaintitiedon hankkiminen oli suhteellisen helppo toteuttaa, kun oikeat laitteet ja toteuttamismenetelmät olivat tiedossa.

4.4.2 Suunta

Ensimmäinen askel suuntaongelman ratkaisua alkoi, ja aloitin sen kokeilemalla suoraan GPIO-pinneihin kiinnittyvää pientä MPU-92/65-sensoria (kuva 12).



Kuva 12. MPU-92/65-sensori johtoineen

Ongelmakohtia tuli sensorin kanssa: raakadata, jota sain ulos sensorista, oli vaikealukoista ja sen kanssa työskenteleminen oli hankalaa. En vielä tuolloin tiennyt, että sensori tulee kalibroida.

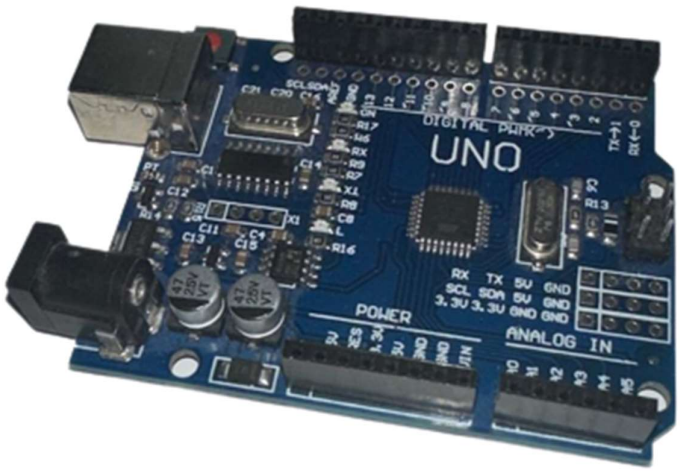
Päivän tutkimuksen jälkeen päädyin siihen, että en rupeaisi tekemään omaa ajurikirjastoa tuolle sensorille. Ajurikirjastot, jota löysin netistä, eivät omalla laitteellani toimineet, vaan ne vaativat ulkoisia kirjastoja, jotka asensin, mutta silti sensori ei toiminut luotettavasti.

Päädyinkin myöhemmin käyttämään Raspberry Pi:lle tehtyä Sense HAT -moduulia, jonka avulla suuntatiedon saaminen kävi kädenkäänteessä. Sense HAT:ille oli valmiit kirjastot, jotka olivat hyvin helppokäyttöisiä.

4.5 Arduino-ajurimoduuli

Seuraava ja viimeinen kysymys projektissa oli, kuinka prosessoitua navigointidataa voitaisiin hyödyntää ohjainlaitteen ohjauksessa. Tämä onnistuisi jonkinlaisella mikrokontrollerilla, tosin on mahdollista, että venettä voisi ohjata myös suoraan AutoSailor-tietokoneella GPIO-pinnien kautta, mutta tätä ohjausmenetelmää en ole toteuttanut.

Venekohtainen ratkaisu oli sarjaporttiyhteydellä toimiva Arduino-mikrokontrolleri, jossa itsessään on sisäinen ohjauslogiikka (kuva 13).

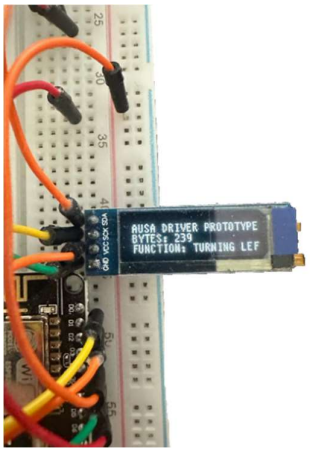


Kuva 13. Projektissa käytetty Arduino-Uno

AutoSailor-tietokoneen ja veneen välikappaleena toimii Arduino, jonka toiminnan käyttäjä voi itse määrittää venekohtaisesti. Tämän projektin tapauksessa Arduino ohjaa veneen peräsimen kulmaa näin ollen ohjaten pelkästään veneen kurssia. AutoSailor lähettää ohjaustietoa prosessoidun navigointidatan perusteella Arduinolle.

Veneen ohjausyksikköön kiinnittyvä ”ajuri” vaatii pientä nikkarointia eri moottoriohjaimien välillä. Olen tehnyt järjestelmästä joustavan, joten parametritiedot ovat täysin muokattavissa.

Käyttäjä pystyy määrittämään veneen ohjaukselle tyyppitiedot ”settings.ini”-tiedostossa. Tiedosto luetaan, ja ohjelma rakentaa ohjainlaitteelle ohjauslogiikan käyttäen tiedostossa määritettyjen parametritietojen mukaisesti. Tämä mahdollistaa sen, että AutoSailor pystyy kommunikoimaan eri ohjauslaitteille, kun parametrit on asetettu jokaisen ohjainlaitteen mukaisesti. Se varmistaa myös sen, että ohjelma lähettää oikeanlaista dataa sarjaporttiyhteyden välityksellä ohjainlaitteelle, tässä tapauksessa ohjaimena toimivaan Arduinoon.



Kuva 14. Prototyyppiajuri

Kuvassa 14 on NodeMCU-ESP8266:lla toteutettu prototyyppiajuri, jonka ohjelmointi ja käyttöympäristö on hyvin samanlainen kuin Arduinolla. Ajuri näyttää sarjaporttiyhteydellä välittyvän datan. Kuvassa näyttää 239 tavua, mikä tarkoittaa sitä, että venettä ohjataan melkein täydellä teholla kääntymään vasemmalle. Tässä tapauksessa moottoriohjaimen liukuma on tavuvälillä 1–249.0 ta-

vua tarkoittaa sitä, että moottoria ei ohjata ollenkaan, eli toisin sanoen autopilotti on pois päältä, ja 1 tavu tarkoittaa, että ohjain ohjaa täysin oikealle ja 249 täysi vasemmalle. 125 tarkoittaa sitä, että ajuri ohjaa kohtisuoraan.

5 NAVIGOINTI

Navigoinnissa tarvitaan ensisijaisesti tieto siitä, missä ollaan ja minne ollaan menossa. Raspberry Pi -tietokoneessa ei ole valmiina komponentteja tätä varten, eikä se pysty suorittamaan navigointia itsenäisesti. Tämä voidaan ratkaista käyttämällä ulkoisia komponentteja, kuten GPS-vastaanotinta ja kompassimoduulia.

GPS-vastaanotin on USB-liitännällä toimiva komponentti, joka vastaanottaa GPS-satelliiteilta lähetettyä tietoa sijainnista, nopeudesta ja ajasta. Tämä data lähetetään Raspberry Pi:lle sarjaportin kautta.

Kompassimoduulina käytän Raspberry Pi Sense HAT -moduuliin integroitua magnetometriä. Sense HAT on Raspberry Pi:n päälle asetettava laajennuskortti, jossa on useita eri sensoreita. GPS-vastaanottimen ja kompassimoduulin yhdistäminen Raspberry Pi -tietokoneeseen mahdollistaa paikannuksen ja suunnan määrittämisen, jolloin navigointi voidaan toteuttaa.

Näiden komponenttien lisäksi tarvitaan ohjelmisto, joka osaa käsitellä GPS-tietoa ja laskea esimerkiksi reitin ja matkan tiedot. Tällaisia ohjelmistoja on saatavilla eri ohjelmointikielillä ja käyttöjärjestelmillä, mutta päätin rakentaa järjestelmän kokonaan itse. Kaiken kaikkiaan navigointijärjestelmän toteuttaminen Raspberry Pi -tietokoneella vaati ulkoisten komponenttien lisäksi ohjelmistokehitystä ja käyttöliittymän suunnittelua.

5.1 GPS-moduuli

Käytin projektissa Beitian GPS -vastaanotinta (kuva 15). Se on USB-liitännällä kiinnittyvä vastaanotin, joka on nopeasti asennettava ja helppokäyttöinen.

Kuva 15. Projektissa käytetty Beitian-GPS-vastaanotin

Raspberry Pi lukee GPS-moduulista satelliittisignaalit ja purkaa niistä sijaintitietoja. Satelliittitietojen purkamisessa käytän Pythonin "gpsd"-kirjastoa, joka helpottaa GPS-datamassojen lukemista ja käsittelyä. Kirjaston avulla Raspberry Pi voi kommunikoida GPS-moduulin kanssa ja lukea siitä GPS-dataa käyttäen "gpsd" daemonia (taustaprosessia).

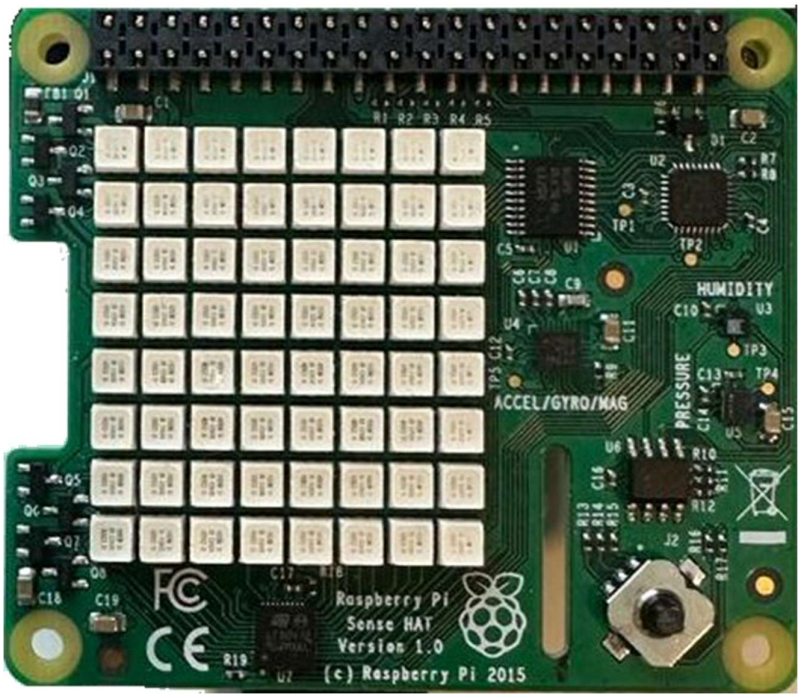
Kirjasto tarjoaa Python-rajapinnan, joka antaa mahdollisuuden lukea sijaintidataa GPS-moduulista. Tämän lisäksi "gpsd" tukee useita erilaisia GPS-moduuleita ja antureita, kuten USB- ja Bluetooth-yhteyksiä, joten sen avulla voi käyttää laajaa valikoimaa GPS-laitteita.

Kirjasto sisältää myös useita erilaisia toimintoja GPS-datamassojen käsittelemiseksi, kuten sijainnin tarkkuuden arviointi ja GPS-signaalin tilan tarkistaminen. Näiden toimintojen avulla Raspberry Pi voi suodattaa ja analysoida GPS-datamassoja, jotta saadaan tarkka sijaintitieto, jota käytän navigoinnissa.

5.2 Sense HAT-Moduuli

Raspberry Pi Sense HAT on pieni laajennusmoduuli Raspberry Pi -tietokoneelle, joka sisältää monia erilaisia sensoreita, kuten lämpötila-, ilmanpaine- ja kosteusanturit. Sense HAT sisältää myös 9-akselisen liiketunnistimen (gyroskooppi, kiihtyvyysanturi ja magnetometri), joka mahdollistaa kompassin käytön.

Kuvassa 16 on esitetty projektissa käytetty Sense HAT-moduuli.



Kuva 16. Sense HAT-moduuli

Opinnäytetyössä käytän pelkästään Sense HAT:in magnetometriä. Valitsin Sense HAT:in kompassiksi, sillä se on yksinkertainen, helppokäyttöinen ja Raspberry Pi:llä on sille valmiiksi asennettu ”sense”-Python-kirjasto.

Sense HAT:in magnetometri antaa meille tietoa siitä, mihin suuntaan vene on kääntynyt ja minne päin se osoittaa. Tämä tieto tallennetaan järjestelmään, ja sitä käytetään navigoinnissa. Sense HAT:in antureiden avulla voisi seurata myös muita tärkeitä tietoja, kuten lämpötilaa, ilmanpainetta ja kosteutta. Näiden tietojen avulla voisimme saada kokonaisvaltaisen kuvan ympäristöstä, jossa vene liikkuu.

5.2.1 Kalibrointiongelma

Koska sähkölaitteet ja magneettikentät ei tunnetusti sovi hyvin yhteen ainakaan tarkkuusmielessä, projektissa ilmeni kalibrointiongelmia vähän väliä. Suurin kalibrointiongelma tuli vastaan, kun yritin käyttää MPU-92/65-sensoria kompassitiedon hakemiseen. Elektroniset magnetometrikompassit on väistämättä kalibroitava, mikäli tarkkuus heittää.

Käytännössä turhauduin projektissa siihen, että kompassi ei näyttänyt toimivan kunnolla. Mutta useiden ongelmanratkaisuyritysten ja tutkimisen jälkeen huomasin, että Raspberry-Pi:ssä kiinnioleva HDMI-johto aiheuttaa häiriöitä anturiin, sillä Sense HAT:in magnetometri on suoraan sen yläpuolella ja aiheuttaa jonkinlaista sähköistä häiriötä.

Näin myöhemmin ajateltuna tämä ongelma ilmeni vasta Multiplexing-laajennuskortin asetuksen jälkeen. Ongelma syntyi siitä, että laajennuskortti siirsi Sense HAT:in suoraan HDMI-johdon päälle, eli jos ongelma halutaan ratkaista ja saada käyttökelpoista kompassitietoa, on ainakin kaksi vaihtoehtoa: joko siirtää Sense HAT kauemmas laitteesta tai irrottaa HDMI-johto, josta jälkimmäinen on kaikista helpoin ja toteuttamiskelpoisin ratkaisu.

5.3 Kokonaisuus

Raspberry Pi 4 (kuva 17) vaati lisälaajennuskortin, jotta Sense HAT mahtuisi sen päälle hyvin ja tukevasti. Ilman lisälaajennuskorttia pinnien kontakti oli heikko ja aiheutti lukuisia ongelmia, kuten tietokoneen sammumisen.

Kuva 17. Raspberry Pi-4 moduuleineen

Lisälaajennuskortin avulla Sense HAT on kiinnitetty tukevasti Raspberry Pi 4:ään, jolloin sen pinnien kontakti on vahva ja luotettava.

AutoSailor-tietokone on noin kämmeneen kokoinen kompakti paketti, jonka avulla on mahdollista ohjata venettä autonomisesti. Järjestelmä on erityisen hyödyllinen pitkillä matkoilla sekä esim. vetouistelussa.

Autonomisen veneen ohjauksen kehittäminen on teknisesti haastavaa ja projektia tehdessä on ymmärrettävä monia eri osa-alueita, kuten ohjelmointia, antureita, mekaniikkaa ja sähkötekniikkaa. AutoSailor-projekti on yksi esimerkki siitä, kuinka näitä osa-alueita voidaan yhdistää tehokkaaksi järjestelmäksi, joka tarjoaa halvan itsetehdyn ratkaisun liikkua järvissä tai jopa merillä.

5.4 Navigointi-”algoritmi”

Nyt kun meillä on tieto siitä, missä olemme ja mihin suuntaamme, voidaan siirtyä navigaation seuraavaan vaiheeseen. Käyttäjä on tässä vaiheessa määrittänyt navigaatioreitin, ja tämä reitti on tallennettu järjestelmään.

Kun autopilotti on kytketty päälle, tietokone pyrkii aina ohjaamaan venettä kohti seuraavaa navigaatiopistettä. Tietokone laskee virtuaalisen navigointipisteen tarkkuuden mukaan noin 10 metrin päähän veneestä interpoloimalla aloituspisteen ja seuraavan pisteen välillä välttääkseen tuulen aiheuttaman suistumisen. Tämän jälkeen tietokone lähettää sarjaporttiviestin "ajurille", joka yrittää kääntää venettä kohti virtuaalista navigointipistettä.

Kun navigointi aloitetaan ja reitti on piirretty, ohjelma rakentaa reitistä ”waypoints”-taulukon, joka koostuu koordinaateista. Taulukon ensimmäinen indeksi 0 on reittipisteen alkupää, jota kohti vene aloittaa suuntaamisen. Kun vene on tarpeeksi lähellä seuraavaa reittipistettä, ohjelma pyytää ”waypoints”-taulukosta vastaavan reittipisteen indeksillä 1. Tämä prosessi jatkuu jokaisen reittipisteen kohdalla, eli aina kun reittipistettä ollaan tarpeeksi lähellä nykyiseen

reitti indeksiin, lisätään +1. Kun reitin viimeiseen pisteeseen päästään, autopilotti sammuu. GPS-data ja suuntatieto päivitetään reaaliaikaisesti, jotta navigaatio on mahdollisimman tarkkaa ja luotettavaa.

6 JOHTOPÄÄTÖKSET JA KEHITYSIDEAT

Projekti opetti todella paljon eri teknologioista ja niiden käytännöllisistä sovelluksista. Opinnäytetyöksi projekti on ollut laaja ja töitä on ollut paljon. AutoSailor-projekti on laajennettavissa oleva ohjelmisto. Olen tehnyt ohjelmistosta joustavan sekä sen ohjelmointirajapinnasta helppokäyttöisen ja laajennettavan. Projektin seuraava vaihe olisi kirjoittaa Back-End puhtaaksi C++ ohjelmointikielellä, mikä lisäisi ohjelmiston tehokkuutta ja laajennettavuutta. Projektissa käytettävistä moduuleista voisi rakentaa yhtenäisen paketin, joka on suoraan Raspberry Pi:n GPIO pinneihin kiinnitettävä moduulipaketti.

AutoSailorin toimintoja ja ominaisuuksia voidaan laajentaa, jotta se pystyy käsittelemään enemmän tietoja ja antamaan tarkempia ja monipuolisempia navigointivaihtoehtoja. Esimerkiksi reitin suunnittelua voidaan parantaa ottamalla huomioon sääolosuhteet, matalikot, virtaukset ja muut tekijät, jotka voivat vaikuttaa veneen liikkeeseen.

Järjestelmän turvallisuutta voidaan parantaa lisäämällä hälytys- ja valvontajärjestelmiä. Esimerkiksi kun vene poikkeaa suunnitellusta reitistä, järjestelmä voi lähettää hälytyksen veneen omistajalle ja lopettaa ohjauksen. Koska sovellus on yhteydessä verkkoon, se mahdollistaa monenlaisten ideoiden toteutuksen.

Jos AutoSailor-autopilottijärjestelmiä olisi useampi, voisi jokainen lähettää yhteiselle palvelimelle, sijainti ja suuntatiedot. Ominaisuus otettaisiin käyttöön ainoastaan luvanvaraisesti, ja käyttäjä voisi itse päättää, onko tämä ominaisuus päällä vai ei. Näin veneilijät voisivat nähdä, missä muut autopilotit kulkevat ja minne ne ovat menossa. Esimerkiksi jos reitit menisivät päällekkäin, autopilotti tekisi reittiin tarvittavat muutokset.

7 LOPUKSI

AutoSailor-projekti on ollut lievästi sanottuna mielenkiintoinen, palkitseva sekä haastava kokemus, joka on antanut minulle paljon lisää tietoa ja taitoja erityisesti automaatioteknologioista. Projektin toteuttaminen on opettanut myös erilaisista verkkopohjaisista teknologioista, kuten JavaScript-ohjelmoinnista, jota olin tuskin koskaan ennen harrastanut.

Opinnäytetyön päättymisen jälkeen aion julkaista rakentamani JavaScript-rakenteen ja EmojionEngine-interaktiomoottorikirjaston niiden siistimisen ja viimeistelyn jälkeen, eli käytännössä tämä mahdollistaa sen, että muutkin voivat rakentaa vastaavanlaisia verkkosivuja käyttämällä tätä modulaarista mallia. Toivonkin, että näistä olisi muille hyötyä ja joku rakentaisi niiden avulla jotain hienoa.

Opinnäytetyö on avannut silmiä sille, mikä todella on mahdollista ja toteuttamiskelpoista sekä miten eri teknologioita voidaan yhdistää yhdeksi toimivaksi kokonaisuudeksi. Aionkin kehittää omaa osaamistani jatkuvasti tällä tekniikan alalla, koska minusta se on loputtoman mielenkiintoista. Opinnäytetyön ohella eri teknologioiden opiskelu on avannut ovia, joihin en ennen opinnäytetyötä omistanut avaimia. Luulenkin, että tulevaisuuteni ja uravalintani nojautuvat enemmän kohti vastaavanlaisia teknologioita.

LÄHTEET

Kaspersky. 2023, SSL-varmenne – määritelmä ja selitys. WWW-dokumentti. Saatavissa: https://www.kaspersky.fi/resource-center/definitions/what-is-a-ssl-certificate [viitattu 24.02.2023].

OpenLayers. 2023, JavaScript-kirjasto. Saatavissa: https://openlayers.org/ [viitattu 18.02.2023]

OpenSeaMap. 2023, Merikartta. Karttapalvelu. Saatavissa: https://map.openseamap.org/ [viitattu 23.03.2023].

OpenStreetMap. 2023, Maailmankartta. Karttapalvelu. Saatavissa: https://www.openstreetmap.org/ [viitattu 23.03.2023].

Raspberry Pi Foundation, Raspberry Pi-4, PNG-kuva. Saatavissa: https://assets.raspberrypi.com/static/raspberry-pi-4-labelled-f5e5dcdf6a34223235f83261fa42d1e8.png [viitattu 14.02.2023].

Scott Campbell, 2016, Basics of the I2C Communication Protocol. WWW-dokumentti. Saatavissa: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/ [viitattu 17.02.2023].

KUVALUETTELO