TAMPEREEN
AMMATTIKORKEAKOULU

REELIN' DOWNRIGGER -SÄHKÖTAKILA

Moottorinohjaus ja käyttöliittymäsuunnittelu

Saku Myyryläinen

Opinnäytetyö
Toukokuu 2016
Kone- ja tuotantotekniikka
Kone- ja laiteautomaatio

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Kone- ja tuotantotekniikka
Kone- ja laiteautomaatio

MYYRYLÄINEN, SAKU:
Reelin' Downrigger -sähkötakila
Moottorinohjaus ja käyttöliittymäsuunnittelu

Opinnäytetyö 104 sivua, joista liitteitä 15 sivua
Huhtikuu 2016

Tämän opinnäytetyön tavoitteena oli vuonna 2014 kalastusharrastuksen innoittamana opiskelijaprojektina alkuun saatetun Reelin' Downrigger -sähkötakilan käyttöliittymän suunnitteleminen sekä aikaisemmin releohjauksena toteutetun moottorinohjauksen päivittäminen älykkäämmäksi. Tarkoituksena oli rakentaa valmis prototyyppi, jonka vahvuuksia ja heikkouksia analysoimalla sähkötakilaa voitiin lähteä kehittämään kaupalliseksi tuotteeksi asti. Täysin valmista myyntiin kelpaavaa tuotetta ei ollut vielä tarkoitus valmistaa.

Työn toteutus koostui myynnissä oleviin sähkötakiloihin ja työhön liittyvään mekatroniikan teoriaan perehtymisestä, eri ohjaustoimintojen ja käyttöliittymän graafisen ilmeen suunnittelusta, komponenttivalintojen tekemisestä, komponenttien testaamisesta sekä ohjauksen toteuttavan koodin ohjelmoimisesta ja testaamisesta osana kokoonpantua laitetta.

Opinnäytetyön tuloksena syntyi aluksi asetettuja vaatimuksia hyvin vastaava sähkötakilan prototyyppi, jossa saatiin toimimaan useimmat siihen suunnitellut toiminnot. Prototyypistä tuli visuaalisesti edustava niin rakenteensa kuin käyttöliittymänsäkin osalta. Käyttöliittymästä tuli myös selkeä ja helppokäyttöinen.

Valmiista prototyypistä löydettiin lukuisia kehityskohteita ja vikoja, jotka tulee ottaa tarkempaan tarkasteluun takilan kaupallista versiota suunniteltaessa. Prototyypin valmistuessa tuotteeseen ideoitiin myös täysin uusia ominaisuuksia, joita ei toteutettu vielä tämän opinnäytetyön puitteissa, vaan ne tullaan lisäämään osaksi sähkötakilaa tulevaisuudessa kehitystyön jatkuessa.

Asiasanat: sähkötakila, moottorinohjaus, käyttöliittymä, mekatroniikka

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Mechanical and Production Engineering
Machine Automation

MYYRYLÄINEN, SAKU:
Reelin' Downrigger
Motor Control and User Interface Design

Bachelor's thesis 104 pages, appendices 15 pages
August 2015

The aim of this thesis was to design a user interface and to update a previously relay controlled motor controller of an electric downrigger to be more intelligent. The downrigger project began as a student project inspired by a fishing hobby in 2014. The purpose was to build a finished prototype which could later be developed into a commercially applicable product by analyzing the prototype's strengths and weaknesses.

The execution of the thesis consisted of becoming familiar with commercially available electric downriggers and the theory of mechatronics related to this topic, the design of different control features and the graphical image of the user interface, the testing of the components and the programming and testing of the control unit's software as part of the fully assembled device.

As a result of the thesis an electric downrigger was created that met well the requirements set for it in the beginning. Most of the planned functionalities worked as intended. Both the structure and the user interface of the prototype became visually presentable. The user interface was also clear and user-friendly.

Many flaws and features for development were found in the finished prototype that will need to be inspected more closely when designing the commercial version of the electric downrigger. While the prototype was being finalized many completely new features that were not part of the initial idea were invented, and they will be added into the downrigger later when the development of the product continues.

Key words: electric downrigger, motor controller, user interface, mechatronics

SISÄLLYS

ERITYISSANASTO

aktiivinen syvyysseuranta	sähkötakilan toimintatila, jossa takila pyrkii pitämään kuulan tietyn etäisyyden päässä järven pohjasta
normaalisti suljettu	NC (eng. Normal Closed), vaikuttamattomassa tilassa johtava kytkin
rpm	kierrosta minuutissa (eng. rounds per minute)
syklinen syvyysseuranta	aktiivista syvyysseurantaa muistuttava toiminta, jossa takila kelaa kuulaa vuorotellen kahdelle eri syvyydelle tietyn aikajakson välein
takila	uistelukalastukseen tarkoitettu vinssimäinen apuväline
väkipyörä	narun tai köyden kulkusuunnan muuttamiseen tarkoitettu pyörä

1 JOHDANTO

Tämän opinnäytetyön tavoitteena oli suunnitella ja rakentaa vuoden 2014 aikana kahden hengen opiskelijaprojektina aluille laitetun Reelin' Downrigger -sähkötakilan mekaniikan osaksi halpa moottorinohjausjärjestelmä ja selkeä käyttöliittymä, joka toteuttaisi tietyt takilan käytön kannalta oleelliset ja monipuolisetkin toiminnot. Tarkoituksena oli kone- ja laiteautomaation suuntautumisen antamien oppien todentamisen lisäksi saattaa takilan prototyyppi esittelykuntoiseksi ja ottaa sen rakentamisesta oppia kaupallisen tuotteen suunnittelua varten. Täysin valmista myyntiin kelpaavaa tuotetta ei siis vielä ollut tarkoitus rakentaa, vaan sellainen tullaan suunnittelemaan myöhemmin tämän opinnäytetyön tuloksena syntyneen prototyypin hyviä ja huonoja puolia kuvaavan analyysin perusteella.

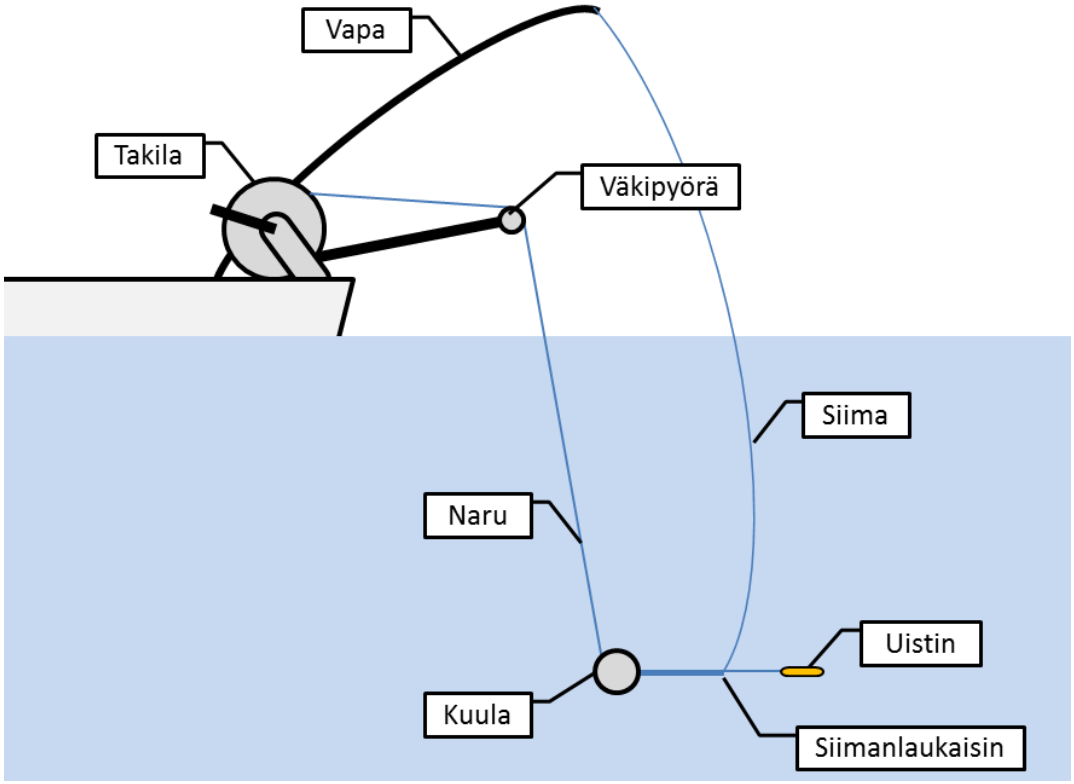
1.1 Ohjausjärjestelmälle asetetut vaatimukset

Sähkötakilan ohjausjärjestelmälle oli heti alusta alkaen asetettu joitakin keskeisiä vaatimuksia (laajempi vaatimusluettelo on liitteessä 1). Sen tuli kyetä muun muassa liikuttamaan sähkömoottoria eri nopeuksilla kumpaankin suuntaan, sekä pitämään kirjaa ja näyttämään lukuarvoa siitä, millä syvyydellä takilan kuula milloinkin roikkui. Näitä toimintoja tuli pystyä ohjaamaan jonkinlaisen helppokäyttöisen näytöllisen käyttöpaneelin välityksellä. Helppokäyttöisyyttä ajatellen käyttöliittymän näytöllä olevien elementtien tuli olla selkeitä ja helposti luettavia, ja lisäksi tekstit tuli olla vaihdettavissa usealle kielelle ja mittayksiköt sekä metreiksi että jaloiksi.

Takilaan haluttiin tavallisen manuaalikelauksen lisäksi myös älykkäitä toimintoja, kuten kuulan automaattinen pintaan tai muuhun ilmoitettuun syvyyteen kelaus, automaattinen syvyysseuranta, joka piti kuulan halutulla etäisyydellä järven pohjasta, sekä syklinen syvyysseuranta, joka liikutti kuulaa vuorotellen kahden eri syvyyden välillä tietyn aikavälin mittaisissa jaksoissa. Ohjausta haluttiin pystyä tekemään myös etänä kaukosäätimen välityksellä, sillä tätä ominaisuutta ei ole kaupallisissa sähkötakiloissa ainakaan laajalti tarjolla. Takilan kelaa käyttävän moottorin tuli olla riittävän tehokas, jotta se pystyi kelaamaan kuulan ylös nopeasti.

2 TAKILAT

Takilat ovat uistelukalastuksessa käytettyjä apuvälineitä, joilla uistin saadaan kulkemaan halutulla syvyydellä vedessä. Takila on veneen sivu- tai takalaitaan kiinnitetty vinssi, jolla lasketaan narun varassa muutaman kilon painoinen kuula veteen kuvan 1 osoittamalla tavalla. Veneessä olevasta vavasta lähtevä siima kulkee kuulaan tai naruun kiinnitetyn siimanlaukaisijan kautta alas veteen. Näin siiman perässä uiva uistin siis pystytään kuulan mukana laskemaan tarkasti mille syvyydelle tahansa, usein lähelle järven pohjaa. Kun kala nappaa kiinni uistimeen, saa tästä aiheutuva nykäisy siiman irtoamaan siimanlaukaisimesta, minkä jälkeen kala voidaan kelata siiman mukana ylös veneeseen. Yhteen takilanaruun voidaan kiinnittää useita siimanlaukaisimia, jolloin sen mukana voidaan vetää veteen useampia uistimia.



Kuva 1. Takilan käyttö uistelukalastuksessa.

Normaalisti uistin ui noin 1 metrin syvyydessä sitä veneen perässä vedettäessä, mutta koska kesäisin kalat uivat lähellä pohjan viileämpiä vesiä, pitää uistin saada kulkemaan syvemmällä lähempänä saalista. Kuulan oikean laskusyvyyden selvittämiseksi ja takilan helpomman käytön apuna käytetäänkin yleensä ääniaaltojen heijastumiseen perustuvaa kaikuluotainta, jolla nähdään helposti, millä syvyydellä pohja on, ja missä kalat uivat. (Hutri & Hutri 2016.)

Takilat voidaan jakaa kahteen ryhmään: käsi- ja sähkökäyttöisiin. Kummassakin takilatyypissä on joitakin yhteisiä rakenteellisia piirteitä. Näitä ovat ainakin

- runkorakenne, joka pitää takilan kasassa kestäen samalla rasitusta ja kosteita sääolosuhteita
- kela, jonka ympärille kiedottu naru kulkee vavan päädyssä olevan väkipyörän kautta veteen kuulaa kannatellen
- runkoon laakeroitu akseli, jonka varassa kela pääsee pyörimään
- vaihteisto, jolla käyttövoima siirretään kammelta tai moottorilta kelan akselille sopivin välityssuhtein
- tukeva takilan veneeseen kiinnittävä mekanismi.

Näiden lisäksi joihinkin takiloihin voi kuulua kiinteä vapateline, sillä takilaa käytetään aina yhdessä erillisen vavan tai vapojen kanssa. Vapateline voi kuitenkin olla kiinnitetty myös esimerkiksi veneen laitaan, eikä sellaista kaikissa kaupallisissa takiloissa aina olekaan.

Käsikäyttöisten takiloiden kuula nostetaan ylös kampea pyörittämällä. Alas laskeminen voi tapahtua joko sekin kampea pyörittämällä, tai avaamalla kammen ja sen vaihteistossa olevan lukitusmekanismin ja päästämällä näin kuulan vajoamaan omalla painollaan veteen. Käsikäyttöiset takilat ovat sähkökäyttöisiä halvempia, sillä niiden hinnat liikkuvat muutamassa sadassa eurossa. (Motonet 2016, Cannon Easi-Troll ST Takila.) Tällainen takilavaihtoehto onkin hyvä satunnaiskalastajalle, mutta pidemmän päälle sellaisen käyttäminen on kammen pyörittämisen takia puuduttavaa ja aikaa vievää.

Sähkötakilat ovat jatkuvaan käyttöön käsikäyttöisiä takiloita mukavampi vaihtoehto, sillä niissä kelan pyörittäminen tapahtuu vaivattomasti sähkömoottorin avulla. Monimutkaisemman rakenteensa takia ne ovat kuitenkin hinnaltaan paljon kalliimpia, ominaisuuksista riippuen jopa yli 1000 euroa. Käyttövoiman lisäksi ehkäpä suurin käsikäyttöisiin takiloihin verrattava ero on sähkötakiloiden moottorinohjaus. Moottoria pitää käyttää jonkinlaisen vesitiiviiseen koteloon suljetun ohjausjärjestelmän avulla, joka ottaa vastaan kalastajan syötteitä vaikkapa painonappien tai kosketusnäytön välityksellä. Tärkein ohjauksen toiminto on kuulan kelaaminen ylös ja alas eri nopeuksilla, mutta sähkötakiloihin on voitu yhdistää myös muita, älykkäitäkin, ominaisuuksia. Melko yleinen ominaisuus

on kuulan syvyyslukeman näyttäminen takilan ohjauspaneelin näytöllä. Kun kuulan syvyys on tiedossa, on mahdollista toteuttaa myös vaikkapa kuulan automaattinen pintaankelaustoiminto, joka pysäyttää kelauksen, kun kuula saavuttaa takilan väkipyörän. (Motonet 2016, Scotty High Performance Tele 5-2116 sähkötakila.)

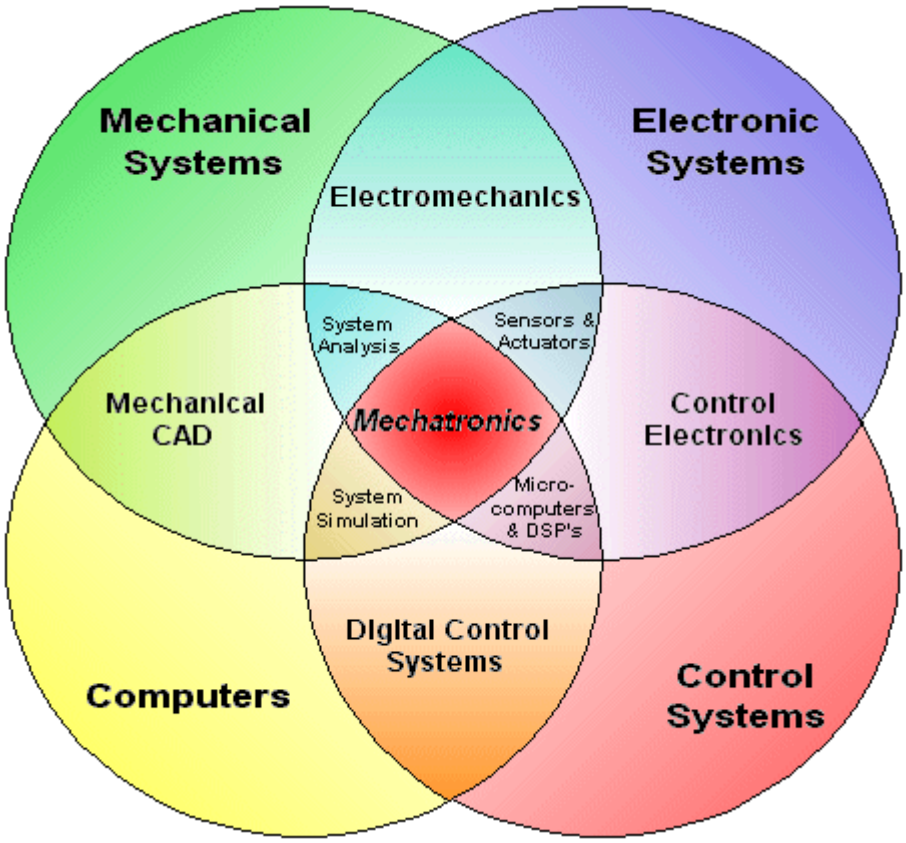
2.1 NMEA 0183 -standardi

NMEA 0183 (National Marine Electronics Association) on merenkäynnissä käytettyjen laitteiden, kuten kaikuluotainten tai GPS-paikantimien, väliseen kommunikoitiin tarkoitettu standardi, joka määrittelee sähköisen rajapinnan ja dataprotokollan näiden laitteiden välille. Tämä vapaaehtoistyöhön perustuva standardi sai alkunsa vuonna 1983, ja sitä on päivitetty useita kertoja. (K. Betke 2000.)

Standardissa sitä käyttävät laitteet jaetaan joko lähettäjiin (eng. talker) tai vastaanottajiin (eng. listener). Standardi sallii kerrallaan yhden lähettäjän, mutta lähetettyä dataa voi vastaanottaa useampi vastaanottaja. Standardissa ei määritellä yhteyteen käytettävää standardiliitintä, mutta tiedonsiirtoon tarvitaan sen mukaan kaksi johdinta, jotka on nimetty kirjaimin A ja B. Kaikki NMEA:n siirtämä data muodostuu ASCII-merkistön merkeistä koostuvista lauseista (eng. sentence), joista jokainen alkaa $-merkillä ja sitä seuraavilla viestin tyypin ilmaisevilla 5 merkillä, jotka ovat tilannesidonnaisia. Lause päättyy rivinvaihtoon. Näiden välissä ovat varsinaiset datakentät pilkulla toisistaan eroteltuina, sekä mahdollinen tarkistussumma tiedonsiirrossa mahdollisesti aiheutuneiden häiriöiden havaitsemiseksi. Kulloinkin käytetyt datakentät määritellään alussa olevien 5 merkin avulla, ja eri merkkiyhdistelmien datakentät ja niiden merkitykset ovat osa NMEA 0183 -standardia. Datakenttien avulla voidaankin ilmaista esimerkiksi autopilottilaitteiden, GPS-paikantimien tai kaikuluotainten käyttämiä suureita. Näitä voivat olla vaikkapa veden lämpötila, tuulen nopeus ja suunta, erilaiset etäisyydet, sijaintitiedot tai tämän opinnäytetyön kannalta oleellinen kaikuluotaimelta saatava järven syvyyden kertova lukuarvo. (K. Betke 2000.)

3 MEKATRONIIKAN TEORIAA

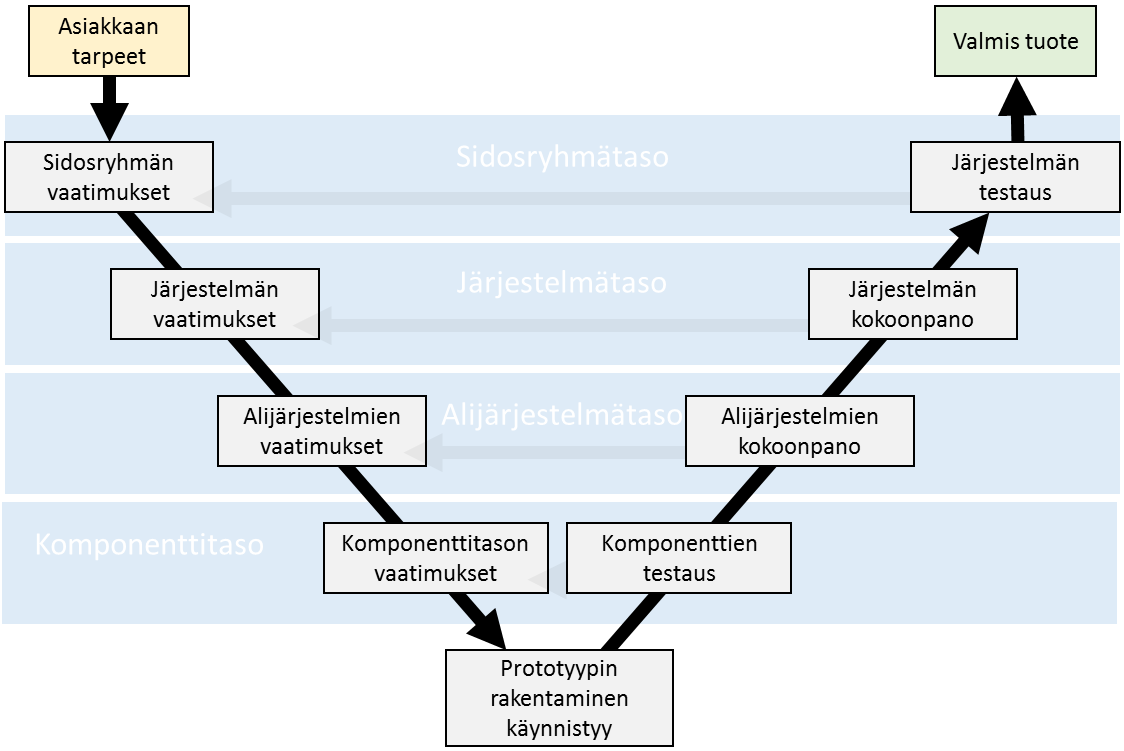
Mekatroniikka on jo 1960-luvulla nimensä saanut ja 1980-luvulla mikroprosessorien yleistymisen myötä kehittynyt tekniikan osa-alue, joka yhdistelee perinteistä konetekniikkaa elektroniikkaan, automaatioon ja tietotekniikkaan (kuva 2). Sille keskeistä on siis jonkinasteisen älyn tuominen osaksi mekaanista laitetta. Tästä hyviä äärimmilleen vietyjä esimerkkejä ovat esimerkiksi teollisuusrobotit, autot tai vaikkapa kopiokoneet, joissa kaikissa mikroprosessoripohjainen ohjainyksikkö saa laitteen tekemään selkeitä fyysisiä toimintoja erilaisia toimilaitteita, kuten sähkömoottoreita, käyttäen. (NC State University 2015.) Tässä luvussa käydään läpi tämän opinnäytetyön kannalta oleellisia mekatroniikan osa-alueita ja komponentteja.



Kuva 2. Mekatroniikka yhdistää keskenään useita tekniikan osa-alueita (NC State University 2015).

3.1 Tuotekehitys mekatroniikan näkökulmasta

Mekatronista laitetta suunniteltaessa on hyvä käyttää niin kutsuttua mekatronista tuotekehitysmallia. Tämä prosessi selviää kuvasta 3. Laitteen kehittäminen lähtee liikkeelle asiakkaan tarpeesta ja hänen tuotteelle asettamiensa vaatimusten listaamisesta. Näitä korkeimman tason vaatimuksia kutsutaan nimellä sidosryhmän vaatimukset, ja ne kuvaavat yleisellä tasolla, miten laitteen tulisi päällepäin katsoen toimia. Näihin vaatimuksiin syvennytään ja ne jaotellaan yksityiskohtaisemmiksi järjestelmävaatimuksiksi, jotka kuvaavat laitteen toimintaa jo paljon teknisemmästä lähtökohdasta. Järjestelmä jaetaan edelleen alijärjestelmiin ja alijärjestelmät yksittäisiin komponentteihin, joille kullekin asetetaan tietyt, myöskin kaikki edelliset tasot täyttävät vaatimukset. (Jouppila 2015.)



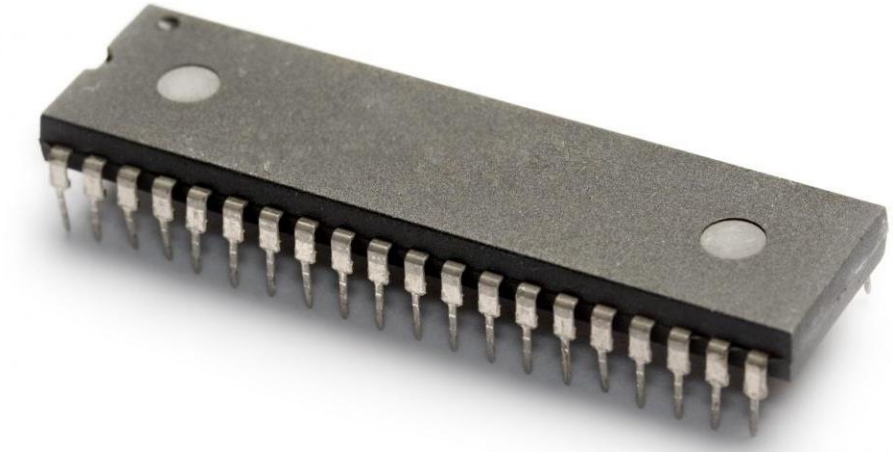
Kuva 3. Mekatroninen tuotekehitysmalli. Prosessi lähtee käyntiin ylävasemmalta asiakkaan tarpeesta ja päättyy yläoikealle valmiiseen tuotteeseen.

Kun suunnittelussa on päästy komponenttitasoisiin vaatimuksiin, voidaan tuotteesta lähteä rakentamaan prototyyppiä näitä vaatimuksia seuraten. Komponentit valitaan komponenttitason vaatimusten mukaan, ja niistä rakennetaan alijärjestelmiä ja edelleen kokonainen tuotteen muodostava järjestelmä kunkin kyseisen tason vaatimuksia noudattaen. Lopulta prototyyppi saadaan valmiiksi ja asiakkaankin vaatimuksia vastaavaksi, ja tuote

voidaan luovuttaa asiakkaalle tai saattaa tuotantoon asti. Oleellista mekatronisessa tuotekehitysprosessissa on siis jatkuva niin sanottu ”taakseen katsominen” ja eri tasoille asetettujen vaatimusten noudattaminen. Muutoin vähänkin monimutkainen järjestelmä menisi helposti todella sekaisin suunnitteluprosessin aikana. (Jouppila 2015.)

3.2 Mikrokontrollerit ja I/O

Elektronisiin laitteisiin älyä tuova komponentti eli mikrokontrolleri (tai mikro-ohjain, kuvassa 4) on pienikokoinen transistoreista ja muista elektroniikkakomponenteista valmistettu ohjelmoitava mikropiiri, siis yksinkertainen tietokone. Oleellista mikrokontrollereissa on niiden ohjelmoitavuus, eli niihin voidaan tallentaa ohjelma, joka mikrokontrollerin sisään- ja ulostulopinnien, eli I/O-pinnien, kautta aistii ympäristöään ja käyttää sitä ympäröivää muuta elektroniikkapiiriä ja edelleen erilaisia toimilaitteita, kuten servomoottoreita. Ne ovat suosittuja muun muassa pienen kokonsa ansiosta, sillä aiemmin ohjauksessa käytettyihin logiikkakomponentteihin verrattuna ne ovat todella pieniä, ja niihin on pakattu paljon toiminnallisuutta. (wiseGEEK 2016.) Ensimmäiset mikrokontrollerit tulivat markkinoille 1970-luvun alussa, ja niiden kehitys on jatkunut siitä asti aiheuttaen niiden hinnan romahtamisen. Nykyään niitä käytetäänkin lähes kaikissa elektronisissa laitteissa yksinkertaisista leluista aina autojen ohjausjärjestelmiin ja vielä monimutkaisempiin järjestelmiin saakka. (Circuits Today 2013.)



Kuva 4. Tyypillinen mikrokontrolleri on suljettu mustalla koteloinnilla, ja sen sivuilla näkyvät niiden I/O-pinnit (wiseGEEK 2016).

Mikrokontrolleri koostuu tyypillisesti seuraavista toisiinsa kytketyistä osista:

- CPU (Central Processing Unit) eli suoritin, joka suorittaa mikrokontrolleriin ohjelmoidun ohjelman kertomia konekielisiä käskyjä
- ROM-muisti (Read Only Memory) eli lukumuisti, suorittimen apunaan käyttämä pysyväismuisti, jota ei normaalisti voi muokata
- RAM-muisti (Random Access Memory) eli keskusmuisti, johon suoritin tallentaa laskentatietoa ohjelman suorittamisen ajaksi
- I/O-pinnit, joiden kautta mikrokontrolleri saa ja välittää tietoa binäärisessä muodossa
- ajastinpiiri suorittimen käyntitaajuuden määrittämiseksi.

Näiden lisäksi mikrokontrollereissa voi olla myös muita ominaisuuksia, kuten

- A/D-muuntimia (Analog to Digital), joiden välityksellä piiri voi käsitellä digitaalitiedon lisäksi myös analogista dataa
- sähköisesti uudelleenkirjoitettavaa EEPROM-muistia (Electrically Erasable Programmable ROM)
- sarjaportteja tiedonsiirtoa varten
- PWM- eli pulssinleveysmodulaatiogeneraattoreita (Pulse Width Modulation).

Nämä lisäkomponentit tuovat mikrokontrollerille paljon lisää toiminnallisuutta. Esimerkiksi A/D-muunnin yhdessä PWM-generaattorin kanssa on hyödyllinen kaksikko, kun kontrollerilla ohjataan analogista tietoa, esimerkiksi jännitteen suuruutta, portaattomasti. Pulssinleveysmodulaatiosta kerrotaan tarkemmin luvussa 3.4.1.

Mikrokontrollereiden ohjelmointi tapahtuu tietokoneen ja siihen asennetun kääntäjäohjelmiston avulla, joka kääntää usein C-ohjelmointikielellä kirjoitetun koodin mikrokontrollerin ymmärtämään konekielimuotoon. Tämä ohjelma voidaan siirtää mikrokontrollerin muistiin esimerkiksi USB-portin tai sarjaportin kautta, mikäli sellainen on mikrokontrolleriin yhdistetty. Kun siirto on tehty, mikrokontrolleri alkaa noudattaa tätä uutta ohjelmakoodia ja sen määrittelemiä toimintoja. (H. Choudhary 2012.)

3.2.1 Arduino-kehitysalusta

Arduino (kuvassa 5) on vuonna 2005 alkunsa saanut avoimeen lähdekoodiin perustuva mikrokontrolleripohjainen prototyyppien kehitysalusta, joka monien muiden mikrokontrollereiden tapaan pystyy vastaanottamaan tietoa digitaalisena ja analogisena, sekä lähettämään ulostulosignaaleja joko digitaalisena tai PWM-metodia käyttäen. Arduinoa ohjelmoidaan sen omalla C-pohjaisella ohjelmointikielellä käyttäen Arduino IDE -koodausympäristöä. Tämä kehitysympäristö on helppokäyttöinen, ja sen ja Arduinon nettisivuilla olevien ohjeiden kanssa kokemattomankin käyttäjän on melko helppo lähteä kirjoittamaan omaa koodiaan ja toteuttamaan projektiaan. Tästä huolimatta Arduino sopii myös vaativampaan ammattilaiskäyttöön, sillä sen avulla on mahdollista rakentaa todella monimutkaisiakin järjestelmiä. Arduino onkin saavuttanut laajan suosion helppokäyttöisyytensä, halpuutensa ja joustavuutensa ansiosta, mahdollistaen esimerkiksi monet opiskelijaprojektit, joiden toteuttamiseen opiskelijoilla ei muutoin olisi ollut varaa. Arduino on joustava myös ohjelmoitavuutensa puolesta, sillä sitä voidaan ohjelmoida sekä Windows-, Macintosh OSX- että Linux-käyttöjärjestelmillä, toisin kuin useimpia muita mikrokontrollereita, jotka ovat rajoittuneet ainoastaan Windowsiin. (Arduino 2016, What is Arduino?.)

Kuva 5. Arduino UNO -piirilevy (Arduino 2016, Getting Started with Arduino on Windows).

Arduino-piirilevyjä on useita eri malleja, ja niiden ominaisuudet vaihtelevat niille tarkoitetun käyttötarpeen mukaan. Tärkeimmät erot lienevät niiden koossa, I/O-pinnien lukumäärässä ja eri muistien (keskusmuisti, EEPROM) suuruksissa. Seuraavaan taulukkoon on listattu joitakin Arduino-malleja tärkeimpine ominaisuuksineen.

Taulukko 1. Arduino piirilevyjen ominaisuuksia.

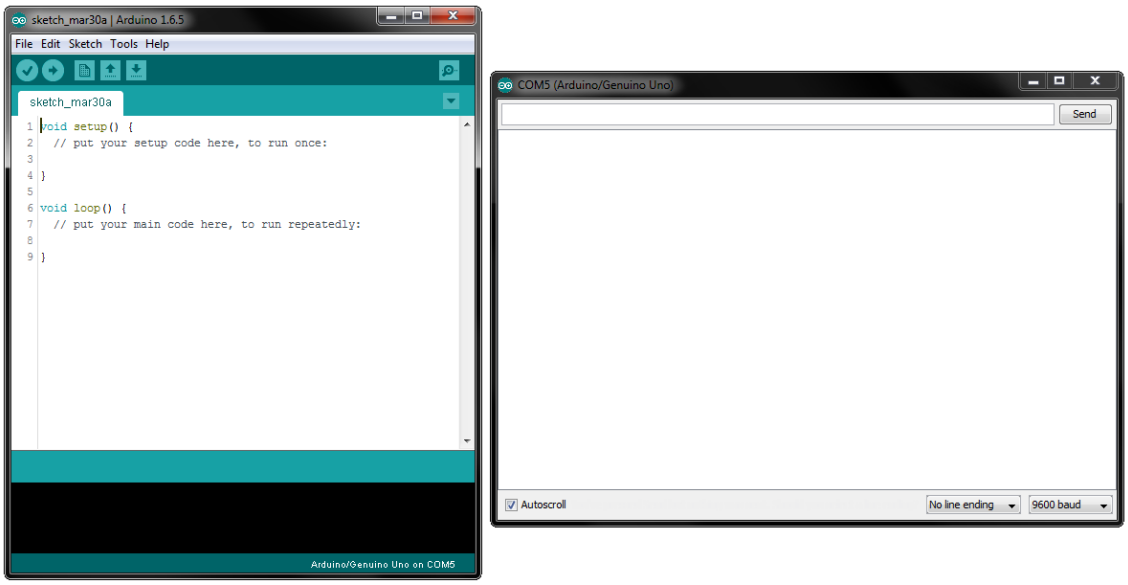
Malli	Flash-muistia / kb	EEPROM-muistia / kb	kello-taajuus / MHz	Digitaali-pinnejä	…joista PWM	Analogisia pinnejä	mitat / mm
UNO	32	1	16	14	6	6	68,6 x 53,4
ZERO	256	0	48	20	18	6	68,6 x 53,4
Nano	16 / 32	0,5 / 1	16	14	6	8	45 x 18
MEGA 2560	256	4	16	54	15	16	101,5 x 53,3
Due	512	0	84	54	12	12	101,5 x 53,3

Arduino-piirilevyt eivät yksinään ole kovinkaan toiminnallisia, vaan niihin tulee kytkeä muita komponentteja, kuten ledejä tai sähkömoottoreita, joita ne ohjaavat, sekä sisääntulotietoa antavia antureita, painonappeja ja vastaavia. Aloittelijapakkauksissa mukana tulevien kytkentäalustojen avulla aloittelijatkin voivat helposti rakentaa omia virtapiirejään Arduinon ohjattavaksi, ja testata niitä. Arduinoihin on myös saatavilla laaja määrä joko yrityksen itsensä valmistamia tai muiden valmistamia shield-piirilevyjä, jotka ovat suoraan Arduino-piirilevyn päälle liitettäviä erillisiä, jonkin tietyn lisätoiminnallisuuden tuottavia, piirilevyjä (kuten kuvassa 6). Tällainen voi olla esimerkiksi moottorinohjaukseen tarkoitettu shield, jossa on kiinteänä kaikki sähkömoottorin ohjaamiseen tarvittavat elektroniikkakomponentit ja valmiit liittimet moottorin johdoille. Lisäksi shieldin mukana tulee monesti sen ohjelmoimiseen tarkoitettu koodikirjasto. Arduino siis ohjelmoitaisiin käyttämään shieldiä, joka tässä esimerkkitapauksessa edelleen käyttäisi itse sähkömoottoria. Shieldit ovat siis ikään kuin rajapintoja Arduinon ja toimilaitteiden välillä. (Arduino 2016, Shields.)

Kuva 6. Arduino-piirilevy, jonka päällä on muistikortinlukijan sisältävä erillinen shield-piirilevy. Shieldin välityksellä Arduino saadaan lukemaan tietoa muistikortilta. (forum.arduino.cc 2011.)

3.2.2 Arduinon ohjelmointi

Kuten aiemmin todettiin, on Arduino IDE -kehitysympäristö (kuvassa 7) saatavilla Windowsille, Macintosh OSX:lle ja Linuxille. Sen avulla käyttäjä voi kirjoittaa C-kieltä läheisesti muistuttavaa Arduinon koodia, kääntää sen mikrokontrollerin ymmärtämään konekielimuotoon ja ladata ohjelman sen muistiin USB-portin kautta. Kehitysympäristö on myös yhteydessä Arduinoon niin kutsutun sarjamonitorin (serial monitor) välityksellä, joka näyttää tietokoneen ruudulla mikrokontrollerin tietokoneelle lähettämää dataa. Tätä dataa voidaan käyttää hyödyksi monissa sovelluksissa Arduinon ja muiden laitteiden väliseen kommunikointiin, sekä apuna koodia kirjoittaessa ohjelmointivirheiden löytämiseksi. (Arduino 2016, Arduino Software (IDE).)



Kuva 7. Arduino IDE -kehitysympäristö.

Tyypillinen Arduino-koodi sisältää ainakin kaksi osaa: setup()- ja loop()-funktiot. Setup() sisältää koodinpätkän, joka suoritetaan aina Arduinon käynnistyessä, ja loop()-funktion sisällä oleva koodi suoritetaan kerran jokaisen ohjelmakierron aikana yhä uudelleen ja uudelleen niin kauan, kuin laite on käynnissä. Näiden, kuten kaikkien muidenkin, funktioiden sisältämä koodi merkitään funktion nimen perässä olevien kaarisulkeiden perään aaltosulkeiden sisään suljettuna. Setup()-funktiossa usein määritellään koodissa käytettyjä muuttujia, määritellään piirilevyn I/O-pinnit joko sisään- tai ulostuloiksi ja käynnistetään erilaisia ulkoisten komponenttien ohjaamiseen liittyviä koodikirjastoja. Loop()-funktioon kirjoitettavalla koodilla sen sijaan saavutetaan laitteen varsinainen toiminnallisuus ja saadaan se reagoimaan asioihin. Näiden lisäksi koodia voidaan jäsennellä jakamalla se muihinkin, samaa syntaksia noudattaviin, funktioihin, joita kutsutaan suoritettaviksi muista funktioista käsin. Tyypillisesti yksittäinen funktio toteuttaa jonkin tietyn toiminnon, jolloin funktiota voidaan kutsua aina kun kyseinen toiminto halutaan suorittaa. Näin samaa koodia ei tarvitse kirjoittaa moneen kertaan kaikkialle, missä sitä tarvitaan. Funktiolle voidaan sitä kutsuttaessa syöttää sen toimintaa muuttavaa tietoa, ja se voi toimintansa pääteeksi myös palauttaa tietoa. (Arduino 2016, Language Reference.)

Tärkeä osa ohjelmointia ovat erilaiset muuttujat ja niiden käyttö koodin osana. Muuttujiin on tallennettuna tietoa jossakin muodossa, ja tämän muodon määrittelee muuttujan tyyppi. Esimerkiksi boolean-tyyppinen muuttuja sisältää totuusarvon, tosi tai epätosi, tai int (lyhenne sanasta integer, kokonaisluku) 16-bittisen kokonaisluvun. Char-tyyppinen

muuttuja sisältää yksittäisen kirjainsymbolin, ja string kokonaisen kirjainjonon. Muuttujien arvoja muuttamalla ohjelma saadaan tekemään asioita ja mikrokontrolleri vaikuttamaan ympäristöönsä, ohjaamalla esimerkiksi digitaalista ulostuloa päälle tai pois päältä boolean-tyyppisen muuttujan välityksellä. (Arduino 2016, Language Reference.)

Koodin etenemisjärjestykseen voidaan vaikuttaa erilaisia ohjausrakenteita käyttäen. Näitä ovat esimerkiksi if-, for- ja while-rakenteet. If-rakenteessa koodin kulku hajautetaan useaan eri reittiin; jos ehto 1 pitää paikkansa, toteutetaan toiminnot 1, jos ehto 2 pitää paikkansa, toteutetaankin toiminnot 2 ja niin edelleen. For- ja while-silmukkarakenteita käytetään toistamaan jotakin koodinpätkää tietyn lukumäärän verran. Yksinkertainen esimerkki tästä voisi olla ohjelma, joka nappia painettaessa vilkuttaa lamppua 5 kertaa. Koodiin ohjelmoitaisiin yksi lampun vilkutus, ja sitä toistettaisiin silmukkarakenteella 5 kertaa, kun napin painallus on havaittu. Näiden rakenteiden lisäksi ohjelmointikieleen kuuluu joitakin erikoissanoja, kuten break tai continue, joiden avulla ohjelman etenemistä voidaan muokata monipuolisemmin. (Arduino 2016, Language Reference.)

Arduino-koodin toiminnallisuutta voidaan kasvattaa helposti asentamalla ja lisäämällä sen osaksi toisten käyttäjien kirjoittamia koodikirjastoja. Nämä kirjastot sisältävät valmiiksi ohjelmoituja toimintoja, jotka usein liittyvät jonkin ulkoisen komponentin ohjaamiseen. Kirjastossa olevia funktioita voidaan kutsua koodista, ja funktio suorittaa jonkin asian, minkä koodaaminen olisi muuten ollut todella työlästä. Tästä esimerkkinä on vaikkapa erillisen LCD-nestekidenäytön käyttäminen ja kirjainten kirjoittaminen sen näytölle komponentin mukana tulevan koodikirjaston juuri tätä tarkoitusta varten kirjoitettuja funktioita käyttäen. Näin Arduinon toiminnallisuutta voidaan lisätä helposti myös koodin puolella ilman, että erillisen komponentin ohjelmointiin tarvitsee tutustua kohtuuttoman syvällisesti. (Arduino 2016, Libraries.)

3.3 Anturointi

Anturit ovat laitteita, joita käytetään elektronisissa järjestelmissä ympäristön havainnoimiseen. Anturi mittaa jotakin fyysistä suuretta, kuten kiihtyvyyttä, lämpötilaa tai pH-arvoa, ja muuttaa sen signaaliksi, jota mittalaite voi lukea ja käsitellä. Signaalin aikaansaaminen perustuu sähkönjohtavuuden muuttumiseen anturin niin sanotussa tuntoelimessä mitattavan fyysisen suureen vaikutuksesta. (S. Mäkelä 2014.)

Eri anturityyppejä on valtava määrä, ja niiden toiminta perustuu mitä erilaisimpiin mekanismeihin ja fysikaalisiin ominaisuuksiin. Eri tyyppejä voidaan lajitella esimerkiksi mitattavan suureen, anturin kytkentätavan tai tunnistusetäisyyden perusteella. Anturin antama signaali voi olla joko analoginen, jolloin sen arvo voi vaihdella liukuvasti mini- ja maksimiarvon välillä (esimerkiksi 4 – 20 mA), tai digitaalinen, jolloin sen arvo on joko 0 tai 1 (epätosi tai tosi, vaikkapa 0 tai 5 volttia). Standardoidut signaalit ovat joko jännite- tai virtasignaaleja jollakin vaihteluvälillä. Tyypillisiä automaatiossa ja elektroniikassa käytettyjä vaihteluvälejä ovat ainakin 4 - 20 mA, 0 - 24 V ja 0 - 5 V. (S. Mäkelä 2014.)

Yksi yksinkertaisimmista ja halvimmista anturityypeistä on niin kutsuttu mikrokytkin, joka on tyypillisesti pieni muoviin koteloitu kytkin. Kotelosta työntyy esiin nappi, vipu tai muu vastaava fyysinen rakenne, jota painamalla mikrokytkin kytkeytyy päälle ja johtaa virtaa lävitseen. Kotelointi saattaa olla vesitiivis, mikä lisää sen käyttökohteiden määrää, mutta samalla myös hintaa. Mikrokytkin, kuten muutkin digitaaliset anturityypit, voi olla joko avautuva tai sulkeutuva, eli se normaalitilassa (vaikuttamattomana) sen läpi joko kulkee tai ei kulje virtaa. Taulukossa 2 on eritelty myös muita yleisiä anturityyppejä niiden toimintaperiaatteen perusteella.

Antureiden ominaisuuksien kuvaamiseen käytetään monia termejä. Yksi näistä on lineaarisuus, joka kuvaa sitä, kuinka suoraviivaisesti anturin antama signaali seuraa mitattavaa suuretta. Anturin epätarkkuus kertoo, kuinka paljon sen antama arvo eroaa todellisesta arvosta. Resoluutio kuvaa anturin kykyä havaita mitta-arvon pieniä muutoksia, sillä hyväresoluutioinen anturi havaitsee mitattavassa suureessa koko mittausalueen suhteen pienetkin muutokset. Mittausalue kertoo, minkä minimi- ja maksimiarvon väliltä anturi kykenee mitattavaa suuretta havaitsemaan. On esimerkiksi suuri merkitys sillä, toimiiko lämpötila-anturi välillä -100 - 500 C° vai 35 - 42 °C. Anturin mittausarvon toistuvuus on sitä parempi, mitä tarkemmin se antaa saman mittausarvon mittauksia toistettaessa, kun mitattava suure ja mittausolosuhteet pysyvät muuttumattomina. (S. Mäkelä 2014.)

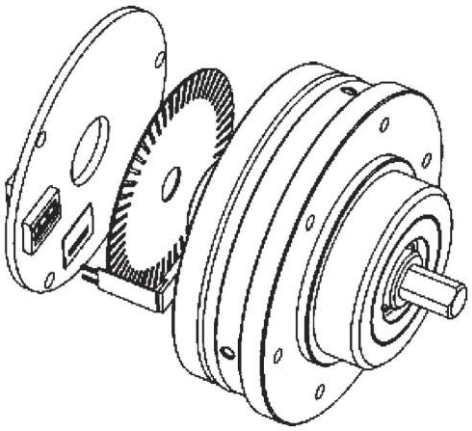
Taulukko 2. Eri anturityyppejä ja niiden ominaisuuksia.

Tyyppi	**Toimintaperiaate**	**Signaali**	**Edut**
Rajakytkin / mikrokytkin	Ulkoinen mekaaninen voima, kosketus	Digitaalinen	Halpa, yksinkertainen
Induktiivinen	Tunnistettava metallinen esine häiritsee anturin luomaa magneettikenttää	Digitaalinen	Kosketukseton tunnistus, ei liikkuvia osia
Kapasitiivinen	Kondensaattorin kapasitanssin muutos tunnistettavan aineen vaikutuksesta	Digitaalinen	Tunnistaa laajasti erilaisia materiaaleja
Magneettinen	Magneetti aiheuttaa fyysisen liikkeen anturissa	Digitaalinen	Sietää hyvin likaa ja kosteutta
Hall	Hall-ilmiö, magneettivuossa tapahtuvan muutoksen havaitseminen	Digitaalinen	Nopea toiminta
Optinen	Valon kirkkauden muutoksen havaitseminen	Digitaalinen	Erittäin pitkä tunnistusetäisyys
Ultraääni	Mittaa ultraäänen kulkuajan kohteeseen	Analoginen / Digitaalinen	Riippumaton tunnistettavan aineen materiaalista ja väristä, sekä valaistuksesta
Laser	Lasersäteen kulkuajan mittaaminen	Analoginen / Digitaalinen	Tarkka etäisyyden tunnistus

3.3.1 Pulssianturi

Tämän opinnäytetyön kannalta oleellisimmaksi anturityypiksi osoittautui inkrementtianturi, eli pulssianturi. Tätä anturityyppiä käytetään mittaamaan koneen akselin kääntymiskulmaa (tai pyörimisnopeutta, eli kääntymiskulmaa ajan suhteen). Rakenteeltaan (kuvassa 8) tyypillinen pulssianturi on yksinkertainen, sillä se koostuu vain valoa säteilevästä ledistä, tämän valon tunnistavista fotodiodeista ja valon kulkua estävästä anturin akselille kiinnitetystä pulssikiekosta. Pulssikiekossa on tasavälein kapeita uria, joiden läpi valo pääsee lediltä fotodiodeille. Kun anturin akseli pyörii, pyörii pulssikiekko sen mukana

päästäen uriensa läpi lyhyitä valopulsseja, jotka fotodiodien ja muun elektroniikan välityksellä muutetaan anturin antamiksi sähköisiksi pulsseiksi. (OEM Automatic 2016.) Vaikka tämä optinen toteutustapa on yleisin, voidaan pulssianturin antamat pulssit muodostaa myös muilla tavoilla (J. Savolainen 2011).



Kuva 8. Pulssianturin rakenne (OEM Automatic 2016).

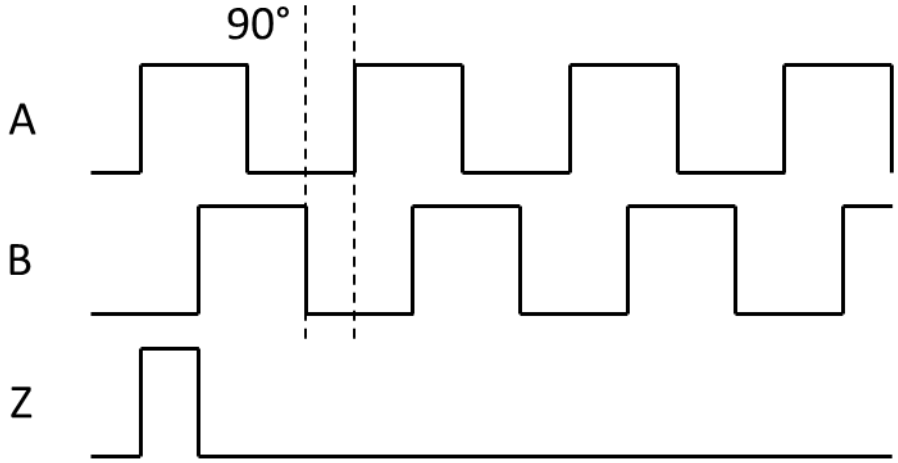
Anturoitavan akselin kiertymän selvittäminen tapahtuu pulssianturin pulsseja laskemalla. Yksi anturin antama pulssi vastaa aina jotakin vakiosuuruista akselin kiertymää, ja tämän kiertymän suuruus riippuu anturin resoluutiosta eli tarkkuudesta. Resoluutio taas määräytyy anturin pulssikiekossa olevien urien lukumäärän perusteella. Laskentakaava pulssianturin resoluutiolle ja yhtä pulssia vastaavalle kulman kiertymälle on

$$R = \frac{360°}{n} \tag{1}$$

missä R on resoluutio asteissa ja n pulssikiekolla olevien urien lukumäärä. Pulssianturin antamien pulssien lukumäärän laskemiseen tarvitaan erillinen laskuripiiri, joka pitää kirjaa pulssien lukumäärästä ja sitä kautta akselin kiertymästä. (J. Savolainen 2011.)

Pulssiantureita voi olla kahta tyyppiä: yksikanavaisia ja kaksikanavaisia. Yksikanavainen anturi antaa pulsseja vain yhdestä ulostulosta, ja tästä syystä sillä ei pystytä havaitsemaan, kumpaan suuntaan akseli pyörii. Tämä tieto ei kaikissa sovelluksissa ole tarpeellinen, mutta kun myös pyörimissuunta halutaan tietää, tulee anturissa olla kaksi pulssikanavaa, jotka antavat pulsseja 90° vaihesiirrossa toisiinsa nähden. Pyörimissuunnan havaitsemista on havainnollistettu kuvassa 9. Pulssianturi antaa kanttimaisia pulsseja kahdella kanavalla

(A ja B). Koska kanavien pulssit menevät ajallisesti lomittain edellä mainitun 90° vaihesiirron takia, voidaan pulssien laskentapiiriin rakentaa yksinkertainen looginen vertailu, joka tunnistaa, kumman kanavan signaali on ollut päällä ensin. Tästä tiedosta saadaan selville, kumpaan suuntaan akseli pyörii anturin antaessa pulssinsa, ja näin laskuri pystyy pitämään kirjaa akselin todellisesta kiertymästä. A ja B -kanavien lisäksi pulssianturissa voi olla vielä niin sanottu Z-kanava, joka antaa yhden pulssin täyttä akselin kierrosta kohti. (OEM Automatic 2016.) Tällaisesta referenssipisteestä voi olla hyötyä esimerkiksi akselin kiertymiskulman nollakohtaa etsittäessä.

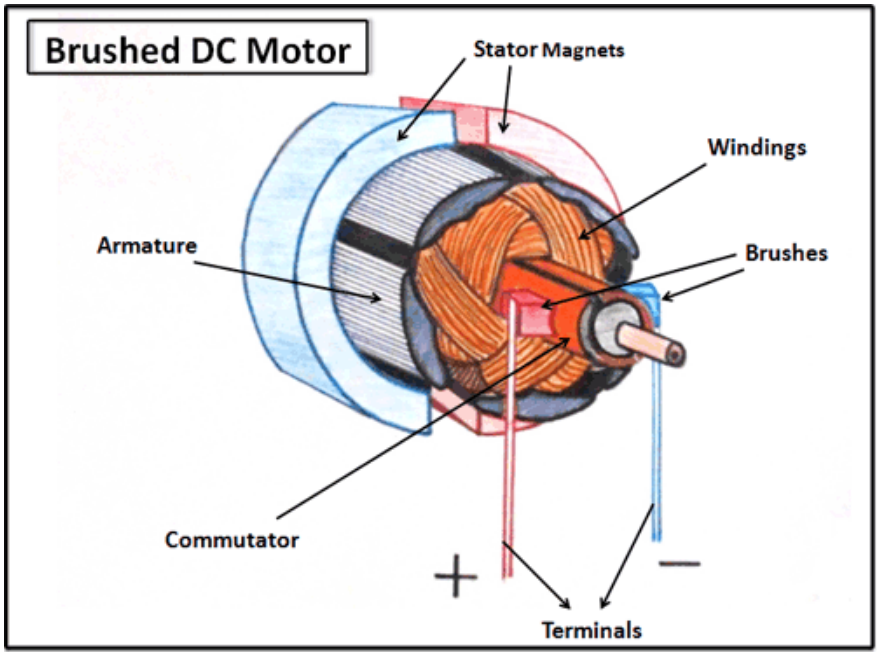


Kuva 9. Pyörimissuunnan selvittäminen kaksikanavaisen pulssianturin pulsseista. Kunkin kanttiaallon alareuna vastaa anturin kanavan digitaaliarvoa 0, ja yläreuna arvoa 1. Aika kulkee kuvassa vasemmalta oikealle.

Ongelmana inkrementtiantureissa on se, että niiden kertoma paikkatieto häviää sähkökatkoksen aikana. Kun anturi kytketään takaisin päälle, ei sillä ole mitään tapaa tietää, mikä akselin senhetkinen kiertymä todellisuudessa on (poikkeuksena Z-signaalin käyttö sijainnin selvittämiseen joissakin tapauksissa). Mikäli tarkka sijainti kuitenkin halutaan tietää, voidaan inkrementtianturin sijaan käyttää hieman monimutkaisempaa absoluuttianturia, joka antaa ulostulonaan tiettyä anturin kiertymää vastaavan n-bittisen binääriluvun. Absoluuttianturin kohdalla kuitenkin törmätään samaan ongelmaan, mikäli akseli pyörii useita kierroksia esimerkiksi liukuhihnaa liikuttaen, sillä kokonaisten kierrosten lukumäärää se ei ilmoita, vaan ne pitää laskea samoin kuin inkrementtianturin yksittäiset pulssit. Todellisen sijainnin selvittämiseksi tulee turvautua muihin keinoihin.

3.4 Harjallisen tasavirtamoottorin ohjaus

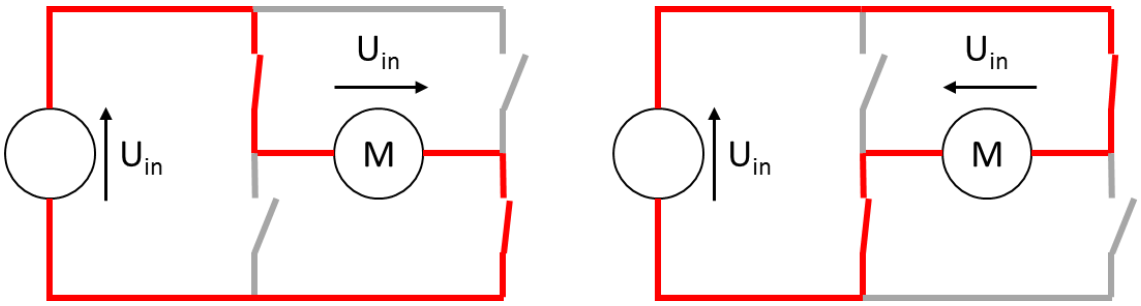
Harjalliset tasavirtamoottorit ovat yksi sähkömoottoreiden alalaji, ja nimensä mukaisesti ne tarvitsevat toimiakseen tasavirtaa. Niiden toiminta perustuu sähkö- tai kestomagneettien luoman magneettikentän sisässä pyörivän sähköjohtokäämin napaisuuden kääntämiseen pyörimisliikkeen eri vaiheissa. Magneetit sijaitsevat moottorin ulkokehällä muodostaen niin kutsutun staattorin (eng. static, paikoillaan pysyvä), ja niiden keskellä on roottorin (eng. rotate, pyöriä) muodostava johtokela (tätä on havainnollistettu kuvassa 10). Kun johtokelan läpi syötetään sähkövirtaa, indusoituu kelan ympärille magneettikenttä, joka pyrkii kääntymään ympärillä olevien magneettien luoman magneettikentän kanssa samansuuntaiseksi. Tästä aiheutuu vääntömomentti akselille, johon kela on kiinnitetty. Kun kela on kääntynyt ulkoisen magneettikentän suuntaiseksi, täytyy sen läpi kulkevan sähkövirran kulkusuunta vaihtaa päinvastaiseksi, jotta kelan indusoima magneettikenttä vaihtaisi suuntansa ulkoiseen kenttään nähden päinvastaiseksi ja jotta kela jatkaisi edelleen pyörimistä. Tämä virran suunnan muuttaminen eli kommutointi tehdään harjallisissa sähkömoottoreissa kommutaattorin hiili- tai metalliharjojen avulla. Nämä harjat hankaavat kelan napoina toimivia kontaktipintoja vasten, osuen vuorotellen niistä kumpaankin. Näin harjoilta tuleva virta kulkee 180 asteen välein kontaktipintojen kautta kelalle vuorotellen eri suuntiin. (M. Brain 2006.)



Kuva 10. Harjallisen tasavirtamoottorin rakenne (CVEL 2016).

Harjallisen tasavirtamoottorin ohjaaminen on melko suoraviivaista. Ne vaativat toimiakseen esimerkiksi 12 tai 24 voltin jännitteen, joka kytketään moottorin kahden navan välille. Jos jännitteen napaisuuden kääntää, kääntyy myös akselin pyörimissuunta. Jos jännitettä pienentää, pienenee myös pyörimisnopeus. (B. Earl 2015.) Tämä nopeuden aleneminen johtuu siitä, että Ohmin lain mukaan jännitteen alentuessa pienenee myös moottorin kelan läpi kulkevan virran suuruus, sillä käämin resistanssi pysyy vakiona. Kun kelan läpi kulkevan virran suuruus pienenee, pienenee myös siihen indusoituvan magneettikentän voimakkuus ja näin myös kentän aiheuttavan vääntömomentin suuruus, aiheuttaen akselin hitaamman pyörimisnopeuden. (Mäkelä, Soininen, Tuomola & Öistämö 2012, 120, 125.)

Käytännössä moottorin jännitteen napaisuuden muuttaminen toteutetaan esimerkiksi kuvan 11 tapaan H-siltakytkennällä. Tässä kytkennässä käytetään 4 kytkintä, esimerkiksi transistoria tai relettä, ohjaamaan virta moottorille jompaankumpaan suuntaan. Kytkennässä vastakkaisissa kulmissa olevat kytkimet toimivat parina. Kytkinparit ovat aina toisiinsa nähden eri asennoissa; kun toisen parin kytkimet ovat kiinni, toisen ovat auki. Kytkinten ohjaus tehdään ulkoisen ohjauspiirin, kuten mikrokontrollerin, avulla. (J. Brown 1998.)



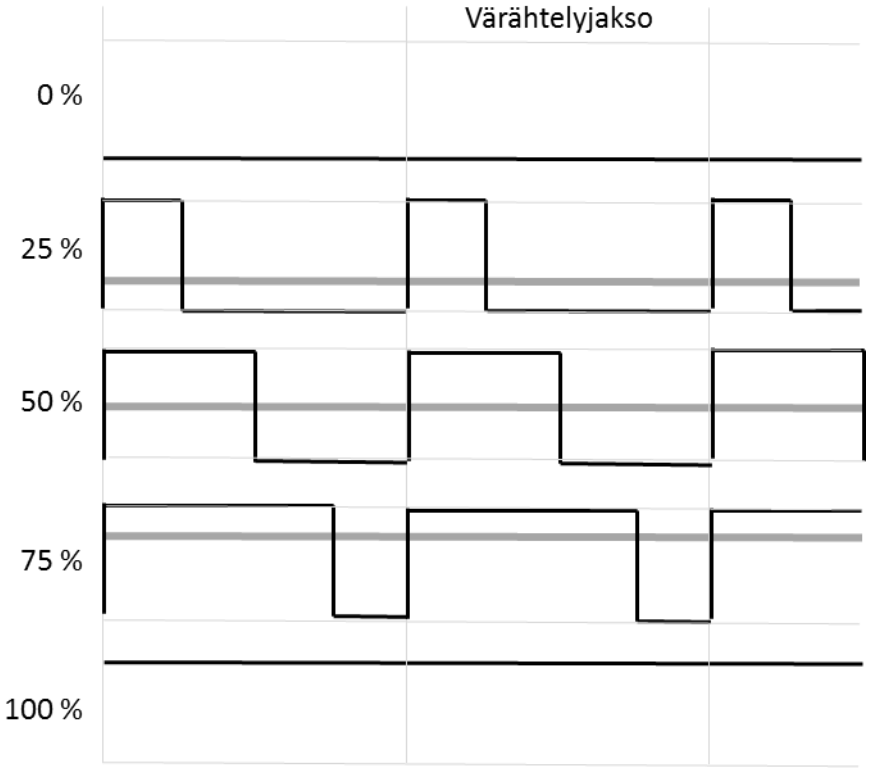
Kuva 11. H-siltakytkentä. Neljää kytkintä oikein käyttämällä saadaan virta kulkemaan moottorissa kumpaan suuntaan tahansa.

3.4.1 Pulssinleveysmodulaatio

Kuten edellä todettiin, riippuu harjattoman tasavirtamoottorin pyörimisnopeus sen napojen välisen jännitteen suuruudesta. Käytännössä jännitteen suuruutta voidaan muuttaa esimerkiksi potentiometrin avulla, mutta kun moottoria halutaan ohjata mikrokontrollerilla,

on turvauduttava digitaalisiin signaaleihin analogiasignaalien sijaan. Tällöin voidaan nopeuden muuttamiseen käyttää niin kutsuttua pulssinleveysmodulaatiota.

Pulssinleveysmodulaatiossa on kyse digitaalisen ulostulon värähtelystä päälle ja pois päältä tarpeeksi suureella taajuudella kuvan 12 osoittamalla tavalla. Yhden värähtelyjakson aikana ulostulon signaali on päällä tietyn osuuden värähtelyjakson kokonaisajasta. Tätä aikaa kuvataan pulssisuhteeksi nimetyllä prosenttiluvulla väliltä 0 - 100 % siten, että 0 % tarkoittaa signaalin olevan kaiken aikaa pois päältä, 100 % arvolla signaali on koko ajan päällä, ja 50 % kohdalla signaali on päällä tasan puolet värähtelyjakson ajasta. Vaikka ulostulo värähtelee päälle ja pois päältä, ei ihminen tätä vilkkumista pysty huomaamaan, kunhan värähtelytaajuus on riittävän suuri. Tarpeeksi suurella taajuudella ulostuleva jännite onkin ulostulon maksimiarvo kerrottuna pulssisuhteella. Signaalin ollessa päällä 100 % ajasta on ulostulolta tuleva jännite esimerkiksi täydet 5 V, 0 %:lla 0 V ja 50 % pulssisuhteella keskimääräinen ulostuleva jännite on 50 % maksimiarvosta, eli 2,5 V. (T. Hirzel 2016.)



Kuva 12. Periaatekuva pulssinleveysmodulaation pulssisuhteen ja ulostulevan jännitteen muodostumisesta.

3.5 Käyttöliittymät ja käyttäjäystävällisyys

Käyttöliittymä on rajapinta, jonka avulla käyttäjä hallitsee laitetta. Käytännössä tämä voi tarkoittaa esimerkiksi koneen käyttöpaneeliin kiinnitettyjä painonappeja, kytkimiä ja numeronäyttöjä, tai nykyaikaisempia kosketusnäyttöjä ja niille ohjelmoituja painikkeita, tekstejä ja muita grafiikoita. Oleellista on, että käyttäjä pystyy sen kautta syöttämään tietoa laitteen ohjaukselle, ja tarvittaessa saa sen toiminnasta tietoa takaisin.

Hyvää käyttöliittymää suunniteltaessa on tärkeää ennakoida, mitä käyttäjä haluaa kulloinkin tehdä. Käyttöliittymän tulee vastata näihin tarpeisiin nopeasti helposti ymmärrettävällä ja intuitiivisella tavalla. Tärkeää onkin käyttää käyttöliittymässä jo muualla yleisesti käytettyjä, ihmisten hyvin ennalta tuntemia vuorovaikutuskeinoja, kuten painikkeita, liukuja, tekstinsyöttökenttiä, ikoneita ja muita vastaavia. Yleensä jokin laitteen toiminto voidaan saavuttaa monin eri keinoin eri vuorovaikutuskeinoja käyttäen. Tällöin kannattaa harkita kunkin vaihtoehdon hyviä ja huonoja puolia, etenkin käyttäjän näkökulmasta. Esimerkiksi paljon pieniä painikkeita täyteen ahdettu kosketusnäytön ruutu voi toteuttaa monia toimintoja käytettyyn tilaan nähden, mutta käyttäjän näkökulmasta tällainen ratkaisu on sekava ja sen käyttäminen vaatii paljon sisäistämistä. (Usability.gov 2016.)

Käyttäjäystävällisen käyttöliittymän suunnitteluun on olemassa joitakin hyväksi todettuja ohjeita. Suunnittelijan tulee

- pitää käyttöliittymä yksinkertaisena ja tehdä siitä mahdollisimman näkymätön käyttäjälle
- tehdä käyttöliittymästä johdonmukainen siten, että eri elementtien toiminnat ja sivujen asettelu ovat mahdollisimman samanlaisia kaikkialla
- olla huolellinen eri elementtien asettelussa, sillä elementtien koko ja sijainti kuvastaa niiden tärkeyttä laitteen toiminnan kannalta
- käyttää värejä, kontrastia, tekstuureita ja fontteja hyödykseen käyttöliittymän selkeyden ja luettavuuden parantamiseksi
- varmistaa, että käyttöliittymä antaa käyttäjälle palautetta järjestelmän toiminnasta
- nopeuttaa käyttöliittymän käyttöä valitsemalla sopivia oletusarvoja käyttäjien toimintaa ennakoiden.

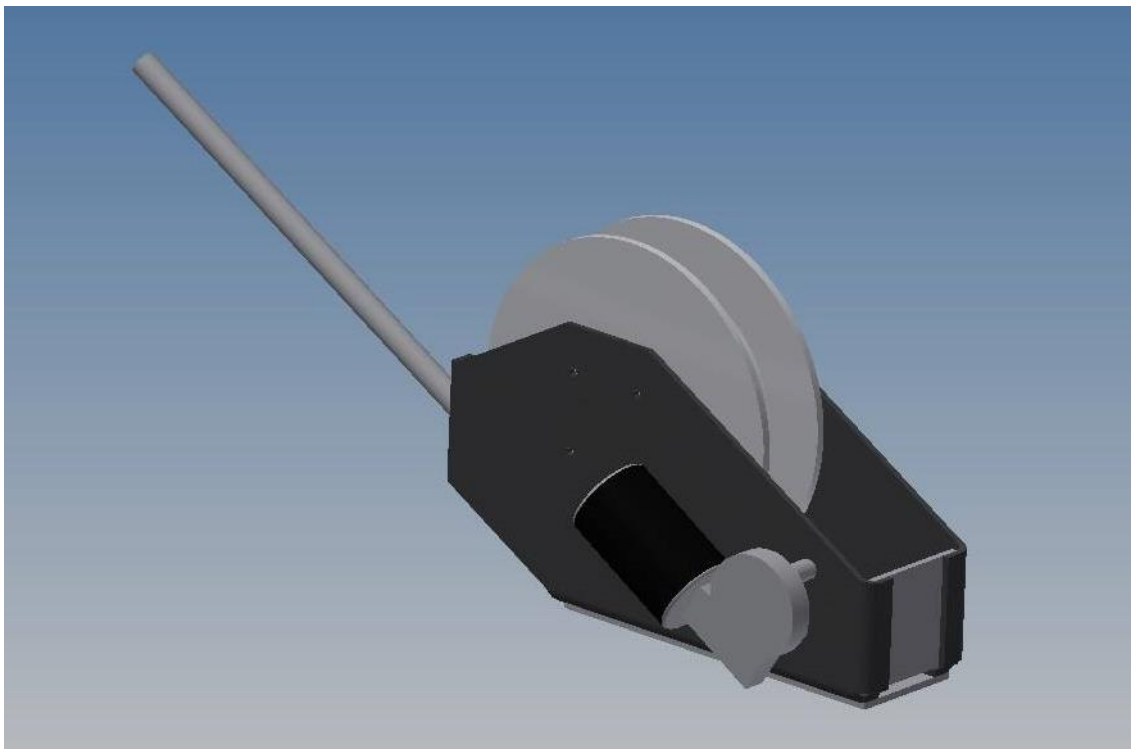
Näitä keinoja noudattaen on käyttöliittymän suunnittelijan mahdollista tehdä liittymästä sellainen, että se ei saa käyttäjäänsä turhautumaan, vaan tekee laitteen käyttämisestä joustavaa ja nopeaa. (Usability.gov 2016.)

4 TAKILAPROJEKTIN SYNTY JA HISTORIA

Tähän opinnäytetyöhönkin johtanut sähkötakilaprojekti sai alkunsa vuoden 2014 alussa tarvelähtöisenä opiskelijaprojektina. Tavoitteena alusta lähtien oli suunnitella ja rakentaa yksityiskäyttöön tarkoitettu opiskelijabudjetille sopiva yksinkertainen sähkötakila. Suunnitelmat kuitenkin paisuivat projektin edetessä jatkuvasti, ja jossakin vaiheessa johtavaksi ajatukseksi tuli prototyypin saattaminen myyntiin kelpaavaksi tuotteeksi asti. Tämä ajatus onkin pidetty mielessä tässä opinnäytetyössä kuvattua pitkälle vietyä prototyyppiä ja sen moottorinohjausta suunniteltaessa ja rakentaessa.

4.1 Alkuluonnokset ja projektivaiheen prototyyppi

Aivan projektin alkuvaiheessa ajatustyö keskittyi lähinnä erilaisten mekaanisten ominaisuuksien, eli rungon rakenteen ja voimansiirron, punnitsemiseen. Takilasta tehtiin erilaisia alustavia 3D-malleja, kuten kuvassa 13 näkyvä rungon muodon ja hihnavälityksen suunnitteluun tarkoitettu malli. Kelan pyörittäminen päätettiin lopulta kustannussyistä tehdä suoralla välityksellä siten, että kelan akseli kiinnitettiin suoraan sähkömoottorin akselille ja ainoaksi vaihteistoksi jäi moottoreissa usein valmiina oleva matovaihde. Matovaihdetta käyttämällä saatiin ohjausta yksinkertaistettua, sillä vaihde jaksoi kannatella takilan kannattelemaa alle 10 kg massaa helposti lepokitkansa varassa, eikä erillistä jarrua tarvittu ollenkaan. Jo tässä vaiheessa alettiin tekemään taustatyötä ohjauksen suunnittelua varten tutkimalla esimerkiksi erilaisia relekytkentöjä.

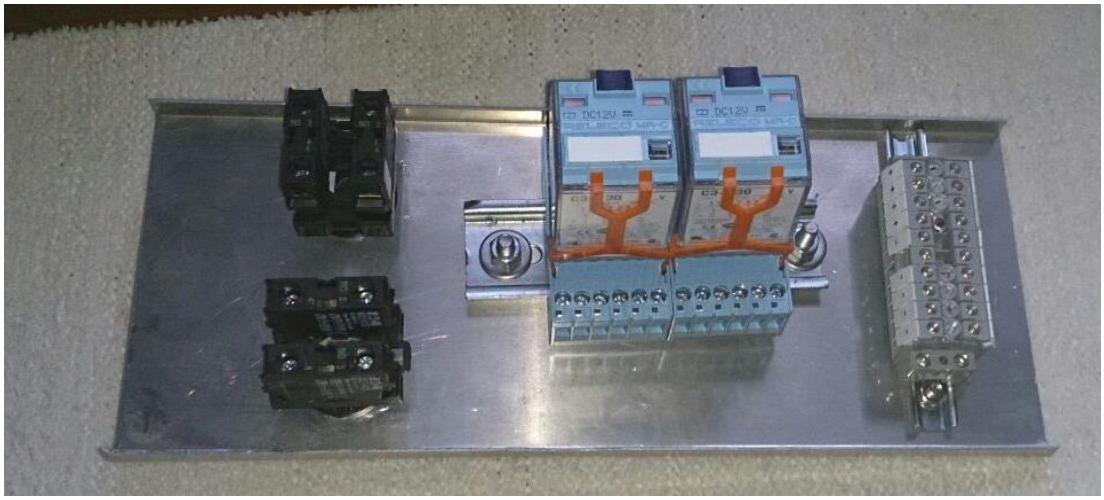


Kuva 13. Takilan mekaniikan hahmottelua varten tehty ensimmäinen 3D-malli.

Ensimmäinen toimiva versio takilasta (kuvassa 14) syntyi nopeasti kevään 2014 aikana. Sitä varten suunniteltiin uusi parempi runko, joka valmistettiin vesileikkaamalla ja kanttaamalla Muototerä Oy -nimisessä yrityksessä. Samalla hankittiin vavan virkaa toteuttava muovinen teleskooppiharjanvarsi, putkikiinnikkeet vavan kiinnittämiseksi runkoon, Scottyn valmistama väkipyörä sekä erinäisiä muita kiinnitystarvikkeita. Kela valmistettiin kahdesta ruuvein toisiinsa kiinnitetystä muovikiekosta. Voimansiirtoa varten hankittiin akseli, laakeripesä sekä auton pyyhkijänmoottoriksi tarkoitettu harjallinen tasavirtamoottori, joka toimi akusta saadulla 12 voltin jännitteellä. Harjattomaan moottoriin verrattain lyhemmästä kestoiästään huolimatta tämä moottorityyppi valittiin siksi, että sen ohjaaminen on yksinkertaista; pyörimissuuntaa pystyy vaihtamaan helposti virran suuntaa vaihtamalla, ja pyörimisnopeuden muuttaminen tapahtuu jännitteen suuruutta muuttamalla, kuten luvussa 3.4 kerrottiin. Moottorin pyörimissuunnan muuttamisen kanssa kuitenkin ilmeni ongelmia, eikä suuntaa saatu vaihdettua ennen kuin moottorissa ollut runkomaadoitus onnistuttiin poistamaan.

Kuva 14. Takilan ensimmäinen prototyyppi testauksessa kesällä 2014.

Tämän version ohjaus toteutettiin vielä melko yksinkertaisella tavalla käyttäen kahta moottorille kulkeneen muutaman ampeerin virtaa kestävää relettä, kahta painonappia, kolmiasentoista kytkintä sekä mikrokytkintä. Painonapeilla ja kolmiasentoisella kytkimellä haluttiin pystyä kelaamaan takilan kuulaa sekä ylös että alas. Kytkimen pystyi jättämään päälle jatkuvan kelauksen mahdollistamiseksi, ja napit oli tarkoitettu lähinnä syvyyden hienosäätöön. Napit ja kytkin kiinnitettiin teräslevystä kantattuun alustavaan kuvassa 15 näkyvään ”etäohjainpaneeliin”. Jotta kuulaa ei pystyisi kelaamaan liian ylös rikkomaan väkipyörää ja takilan muuta rakennetta, päätettiin väkipyörään kiinnittää mikrokytkin, joka havaitsee tällaisen takilaa vahingoittavan liikkeen ja pysäyttää moottorin pyörimisen ajoissa.



Kuva 15. Ensimmäisen prototyypin etäohjauspaneelin tausta komponentteineen.

Ohjausvirtapiirin pohjana käytettiin aiemmin kuvattua kuvan 11 kaltaista h-siltakytkentää, mutta usein käytettyjen transistoreiden sijaan kytkiminä käytettiin kahta relettä, joissa oli kummassakin kolme sekä sulkeutuvaa että avautuvaa kärkeä. Näin suunniteltiin liitteessä 2 nähtävä kytkentä, jossa painonapit ja kolmiasentoinen kytkin käyttävät releitä, jotka edelleen päästävät virtaa moottorille jompaankumpaan suuntaan. väkipyörässä oleva mikrokytkin katkaisi virran kelattaessa moottoria ylös. Virta ei kulkenut moottorille myöskään silloin, jos kumpaakin nappia painoi yhtä aikaa.

Kaikki kytkennät tehtiin riittävän paksujen sähköjohtojen ja riviliittimien avulla. Valmista kytkentää jouduttiin releiden 24 voltin jännitevaatimuksen takia testaamaan kahdella sarjaan kytketyllä 12 voltin akulla, ja kytkentä toimi juuri suunnitellulla tavalla. Jo tämän ensimmäisen prototyypin aikana kuitenkin todettiin tällaisen ohjauksen olevan ominaisuuksiltaan liian yksinkertainen, ja tilalle haluttiin jotakin monimutkaisempaa. Hyväksi vaihtoehdoksi nousi luvussa 3.2.1 esitellyn Arduino-kehitysalustan käyttäminen ohjaukseen älyä tuovana komponenttina. Kehitysalustan edullisuus ja hyvä muunneltavuus sopivat hyvin projektin tavoitteisiin ja antoi samalla projektin tekijöille hyvän syyn tutustua mikrokontrollereiden ohjelmointiin.

Tässä luvussa kuvattuun vaiheeseen rakennettu sähkötakila toteutettiin 5 opintopisteen suuruisena opiskelijaprojektina TAMKin ja lehtori Petri Pohjolan opastuksella. Kuten johdannossa jo todettiin, monipuolisemman ohjauksen ja käyttöliittymän kehittäminen päätettiin takilan projektivaiheen jälkeen toteuttaa tämän opinnäytetyön muodossa. Sen kehittäminen lähti käyntiin siitä, mihin takila tässä luvussa kuvatun työpanoksen jäljiltä oli jäänyt.

5 SUUNNITTELU JA KOMPONENTTIVALINNAT

Tämän opinnäytetyön tuloksena syntyneen sähkötakilan ohjauksen suunnittelu ja valmistaminen toteutettiin luvussa 3.1 kuvattua mekatronista tuotekehitysmallia seuraten. Koska takilaprojekti lähti käyntiin jo noin kaksi vuotta sitten, ennen neljäntenä lukuvuotena suoritettua mekatroniikan kurssia, eivät takilan tekijät aluksi tietoisesti noudattaneet tätä mallia. Suunnittelu mukaili kuitenkin luonnostaan luvussa 3.1 kuvailtuja vaiheita, sillä ohjauksen suunnittelu ja rakenne oli järkevintä jakaa osakokoonpanoihin, joihin tässä luvussa yksitellen paneudutaan. Kaikista työvaiheista ei tehty yksityiskohtaisia kirjallisia suunnitelmia, kuten suunnitteluprosessin mukaan olisi ehkä ollut hyödyllistä tehdä, vaan monia asioita suunniteltiin ja vaihtoehtoja vertailtiin ainoastaan sanallisesti takilan tekijöiden ja myös joidenkin ulkopuolisten henkilöiden kesken. Kirjallisiakin suunnitelmia kuitenkin myös tehtiin, ja moni niistä on sisällytetty tähän opinnäytetyöhön. Joitakin osakokonaisuuksia myös rakennettiin jo paljon ennen kuin kaikkia kokonaisuuksia oli edes suunniteltu.

Sähkötakilan ohjausjärjestelmän muodostavat komponentit voitiin jakaa ohjattavaan moottoriin ja mekaniikkaan, anturointiin sekä ohjainkoteloon sisältöineen. Ohjainkotelo jaettiin edelleen pienemmiksi kokonaisuuksiksi, eli mikrokontrolleriin, käyttöliittymään, komponenttien välisiin kytkentöihin sekä itse kotelorakenteeseen. Seuraavaksi paneudutaan näihin alijärjestelmiin, niihin liittyviin vaatimuksiin, suunnitteluun ja komponenttivalintoihin.

5.1 Moottori ja moottorinohjaus

Ehkäpä oleellisin komponentti takilan toiminnassa oli sen sähkömoottori, ja siksi sille asetettiinkin melko tarkat vaatimukset. Niistä johtuen sopivan moottorin löytäminen oli myös yksi työn hankalimpia vaiheita. Ohjauksen kannalta oleellisimmat moottorille asetetut vaatimukset olivat, että sitä tuli pystyä ohjaamaan 12 voltin jännitteellä kumpaankin suuntaan eri pyörimisnopeuksilla, ja siinä tuli olla matovaihde, jonka lepokitkan varassa kuula roikkui ilman, että erilliselle takilaa monimutkaistavalle jarrumekanismille olisi ollut tarvetta. Toisaalta mekaniikan kannalta tärkeää oli, että moottorin tuli olla mahdollisimman kevyt ja pienikokoinen, sekä jaksaa nostaa 10 kg massaa, eli vääntömomentin tuli olla vähintään

$$M = Fr = mgr \tag{2}$$
$$M = 10\ kg * 9{,}81\ \frac{m}{s^2} * 0{,}045\ m \approx 4{,}5\ Nm$$

Kaavassa (2) F on kuulan naruun, eli kelan ulkokehälle, kohdistama voima, joka lasketaan kuulan massan m ja putoamiskiihtyvyyden g tulona. Kun tätä voimaa kerrotaan kelan säteellä r, saadaan moottorin akselille kohdistuva enimmäismomentti M.

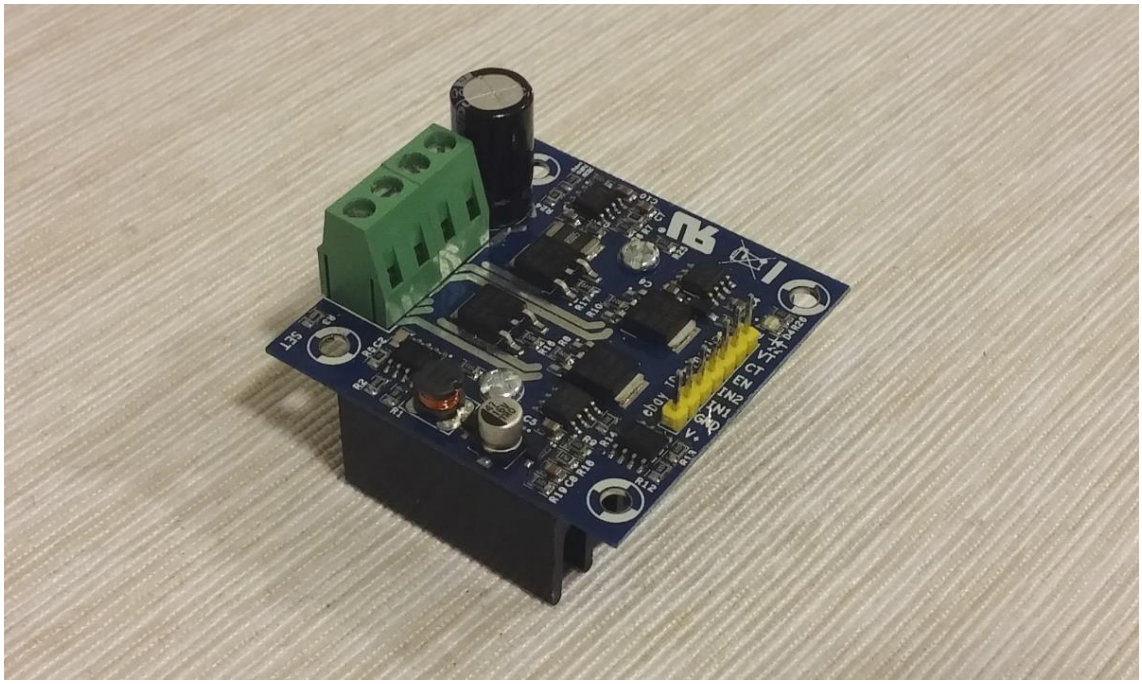
Näiden vaatimusten puitteissa päädyttiin hankkimaan kuvassa 16 näkyvä auton tuulilasinpyyhkijän moottoriksi tarkoitettu 12 voltilla toimiva matovaihteellinen harjallinen tasavirtamoottori. Moottori etsittiin ja tilattiin huuto.net sivuston kautta vain noin 30 € hintaan. Vaikka harjaton moottori olisi ollut huoltovapaampi ja pitkäikäisempi kuluvien kommutointiharjojen puuttumisen vuoksi, ja siksi parempi vaihtoehto, on sen ohjaaminen vaikeampaa. Harjattomissa moottoreissa kommutointi tapahtuu elektronisesti ohjainpiirin avulla, mistä johtuen sen pyörimissuunnan ja -nopeuden muuttaminen on hankalampaa harjallisiin moottoreihin nähden. Harjallisten moottoreiden ohjaaminen tapahtuu sen sijaan helposti pelkästään jännitteen suuntaa ja suuruutta muuttamalla.

Kuva 16. Takilaan valittu tuulilasinpyyhkijän moottori.

Moottorin ohjaamista varten tarvittiin erillinen, luvussa 4.1 kuvattua projektivaiheen ohjausta varten itse rakennettua kytkentää hienostuneempi, H-siltakomponentti, sillä valmiita komponentteja oli paljon tarjolla eikä sellaista haluttu lähteä itse rakentamaan. Tieto

H-siltakytkennästä hyvänä moottorin ohjaamiseen soveltuvana kytkentänä tuli jo takilaprojektin alkuvaiheessa erään tämän työn tekijöiden tunteman kontaktin kautta, eikä muita kytkentätapoja ollut tarvetta harkita enää tässä vaiheessa tämän kytkennän yksinkertaisuuden, toimivuuden ja yleisyyden vuoksi. Näiden komponenttien tiedettiin myös olevan hinnaltaan täysin kohtuullisia työn budjettiin nähden.

H-siltakomponentille oli tiedossa selkeitä vaatimuksia. Sen tuli kestää moottorin ottamaa usean ampeerin suuruista virtaa 12 voltin jännitteellä hajoamatta, ja sitä piti pystyä ohjaamaan mikrokontrollerin I/O-porttien käyttämällä 5 voltin ohjausvirralla. Lisäksi sen tuli toimia PWM-pulssituksen kanssa, jotta myös moottorin nopeutta pystyttiin ohjaamaan. Sopiva, usean kymmenen ampeerin virtaa kestävä ja PWM-pulssituksen kanssa toimiva, komponentti löytyi helposti ebay.com -nettikaupasta vaihtoehtoja selailemalla noin 10 € hintaan (kuvassa 17). Komponentteja tilattiin 2 kappaletta, sillä uuden tilaamiseen ei haluttu käyttää aikaa, mikäli komponentti hajoaisi sitä testattaessa tai olisi jo saapuessaan valmiiksi hajalla.



Kuva 17. Moottorin ohjaamiseen käytetty H-siltakomponentti. Alapuolella oleva jäähdytyssiili piti huolen siitä, ettei komponentti ylikuumentunut suurillakaan virroilla.

5.2 Anturointi

Kuten liitteen 1 vaatimusluettelossa on kuvattu, haluttiin takilan pystyvän muun muassa

- näyttämään näytöllä, mille syvyydelle kuula on laskettu
- kelaamaan kuula mille tahansa halutulle syvyydelle, automaattinen pintaan kelaus mukaan lukien
- pysäyttämään ylöskelauksen, mikäli kuula kelattaisiin kiinni väkipyörään
- pitämään kuula tietyn etäisyyden päässä järven pohjasta (aktiivinen ja syklinen syvyysseuranta).

Jotta nämä ominaisuudet pystyttiin toteuttamaan, tuli takilan ohjaimelle syöttää erinäistä anturidataa. Todettiin, että kaikki yllä kuvatut toiminnot pystyttiin saavuttamaan selvittämällä vain kolme suuretta, jotka olivat

- kelan pyörähdyskulma
- kuulan nouseminen liian lähelle väkipyörää
- järven syvyys.

Kelan pyörähdyskulman avulla pystyttiin pitämään kirjaa kuulan sijainnista, kun kelan ulkohalkaisija oli tiedossa ja syvyyslaskuri oli nollattu kuulan roikkuessa veden pinnalla. Kuulan nousemisesta lähelle väkipyörää riitti yksinkertainen on/off-tieto. Aktiivinen ja syklinen syvyysseuranta pystyttiin toteuttamaan yhdessä kelan pyörähdyskulman ja järven syvyyden kertovan tiedon avulla. Seuraavaksi punnitaan eri anturointivaihtoehtoja ja lopullisia valintoja jokaisen suureen kohdalla.

5.2.1 Kelan pyörähdyskulman anturointi

Kelan pyörimisen anturointiin löydettiin useita eri ratkaisuvaihtoehtoja, joissa kaikissa esiintyi pulssianturin piirteitä ainakin jossakin muodossa. Absoluuttianturit jätettiin kokonaan huomiotta, sillä takilan toiminnan kannalta ei ollut oleellista tietää tarkkaa pyörähdyskulmaa sähkökatkon jäljiltä, koska kela pyörisi ympäri useita kierroksia ja siksi kuulan syvyystieto jouduttaisiin joka tapauksessa nollaamaan vedenpinnan tasolle mahdollisesti jokaisen käyttökerran aluksi.

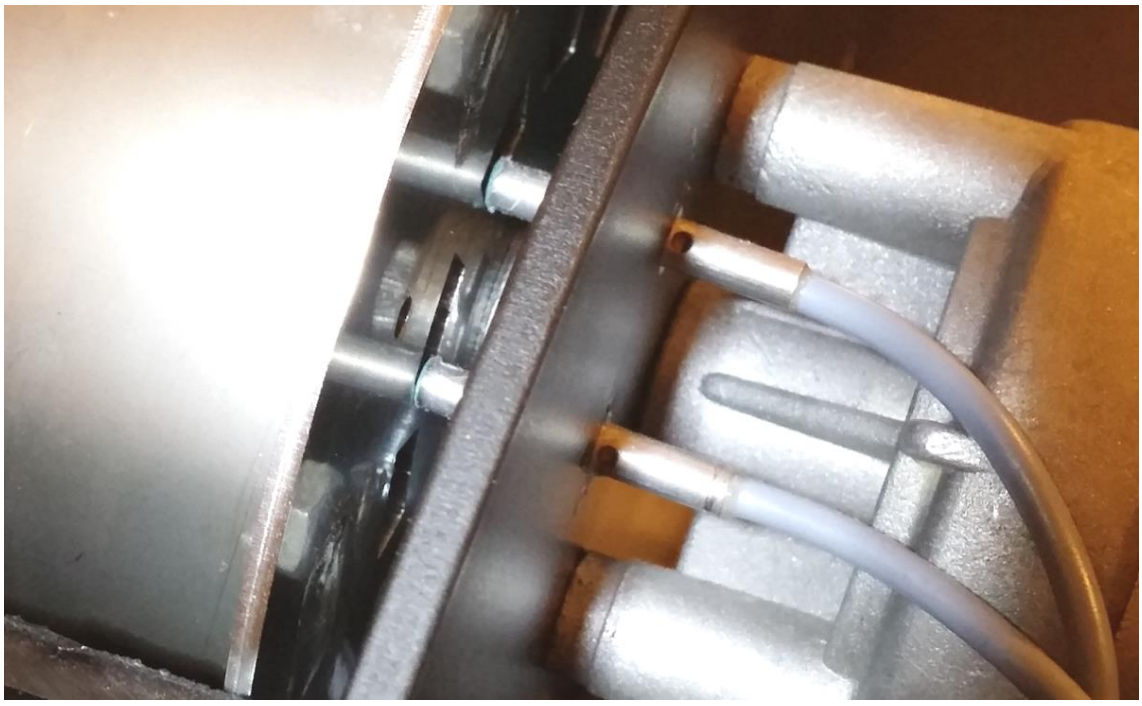
Eräs vaihtoehto tämän suureen anturointiin oli sähkömoottoriin valmiiksi integroitu pulssianturi tai Hall-anturi, jolta olisi saatu suoraa kaksikanavainen pulssitieto mikrokontrollerille. Tällaista, myös kaikki muut vaatimukset täyttävää, moottoria ei kuitenkaan onnistuttu löytämään, tai löydetyt osoittautuivat liian kalliiksi. Takilan rakenteen yksinkertaistamisen kannalta tämä vaihtoehto olisi kuitenkin ollut loistava.

Koska moottoriin integroitua pulssianturia ei onnistuttu löytämään, alettiin yhtenä vaihtoehtona miettiä jollakin tavalla kelan akselille kiinnitettyä erillistä pulssianturia. Ongelmaksi tässä kuitenkin muodostui takilan rakenteen ylenpalttinen monimutkaistuminen. Erillistä pulssianturia ei olisi mitenkään pystytty kiinnittämään akselin kumpaankaan päätyyn, sillä toinen pää oli kiinni moottorissa ja toinen oli laakeroitu laakeripesän sisään takilan rungon sisäpuolta vasten. Runkoon ei myöskään haluttu tehdä reikää anturia varten, sillä anturi olisi pitänyt sijoittaa takilan rungon ulkopuolelle, jolloin tuotteen ulkomitat olisivat suurentuneet kohtuuttoman paljon. Erillisen pulssianturin kohdalla ainoaksi vaihtoehdoksi jäi anturin akselin yhdistäminen kelan akselille jotakin välitystä, kuten hammaspyörä- tai hihnavälitystä, käyttäen. Tämä tapa olisi ollut tilan käytön kannalta hankala, ja se olisi vaatinut monia uusia (kuluvia) osia takilan rakennetta monimutkaistamaan.

Koska erillisen pulssianturin käyttäminen ei tullut kyseeseen, alettiin kelan pyörimisen anturoimiseksi suunnitella itse rakennettavaa pulssianturia, joka voitiin rakentaa vapaasti omilla mitoilla osaksi takilan, ja etenkin kelan, rakennetta. Aluksi ideana oli käyttää magnetismiin perustuvia Hall-antureita yhdessä kelan kehälle kiinnitettyjen magneettien kanssa. Kelan pyöriessä magneetit olisivat kulkeneet Hall-antureiden ohitse antaen näin pulsseja mikrokontrollerille. Magneettien määrää lisäämällä olisi anturoinnin resoluutiota saatu kasvatettua. Tätä varten hankittiin muutama pienikokoinen Hall-anturikomponentti, joita testattiin yhdessä pienen, mutta voimakkaan, kestomagneetin kanssa. Osoittautui kuitenkin, että tästä vaihtoehdosta olisi tullut todella hankala toteuttaa, sillä toimiakseen Hall-anturit vaativat toisen magneetin taakseen, joka ilmeisesti vetäisi sisällä olevan kytkimen takaisin lähtöasentoon sen jälkeen, kun kelan kehällä ollut voimakkaampi magneetti oli vetänyt kytkimen toiseen ääriasentoon. Tällainen magneettien tarkka asettelu olisi ollut todella hankala saada toimivaksi.

Hall-antureiden hylkäämisen jälkeen lopulliseksi ratkaisuksi päätyi TAMKin opettaja Seppo Mäkelältä saatujen induktiivisten antureiden käyttäminen Hall-antureiden tilalla.

Magneetit korvattiin kelan osaksi integroidulla teräksisellä pulssikiekolla, jossa oli koloja, joiden kohdalta induktiiviset anturit eivät tunnistaneet metallia. Tämän idean yhteydessä koko kela suunniteltiin uudestaan, ja se päädyttiin tekemään kahdesta keskenään identtisestä 2 mm teräslevystä laserleikatusta kiekosta, joiden väliin taivutettiin ohuemmasta 0,5 mm teräslevystä leikattu keskiö. Levyjen valmistuskuvat ovat liitteessä 3. 4 mm halkaisijaltaan olleet anturit päätettiin liimata rungon seinämän läpi porattuihin reikiin sopivalle etäisyydelle kelan moottorinpuoleisesta kiekosta. Näin ne antaisivat pulsseja mikrokontrollerille kelan pyöriessä ja sen kyljessä olevien urien kulkiessa antureiden ohi, kuten kuvasta 18 voi nähdä.



Kuva 18. Kelan sivuun integroitu pulssikiekko ja sitä lukevat induktiiviset anturit.

Tämän itse rakennetun pulssianturin kumpikin induktiivinen anturi antoi mikrokontrollerille 12 pulssia yhtä kelan kierrosta kohti, mikä oli takilan toiminnan kannalta riittävä ja kyseisillä antureilla, kelan muodolla ja takilan muulla tähän liittyvällä rakenteella saavutettava paras resoluutio. Kiekon uritusta ei voitu tehdä kapeammalla välillä, sillä muuten 4 mm halkaisijaltaan olleet anturit eivät olisi tunnistaneet uria kunnolla, mutta onneksi tälle ei ollut tarvetta, vaan 12 uran jaolla päästiin kuulan sijaintitiedon suhteen helposti haluttuun noin 10 cm tarkkuuteen. Koska kelan halkaisija oli 90 mm, vastasi yksi pulssi tällöin kuulan syvyyden muuttumista noin

$$\frac{\pi \cdot 90\ mm}{12} \approx 24\ mm \quad (3)$$

verran jompaankumpaan suuntaan. Kun nollataso on ensin kalibroitu, voi mikrokontrolleri pitää kirjaa pulssien määrästä ja muuntaa sen yllä lasketun lukeman avulla tiedoksi syvyydestä, jolla kuula roikkuu. Nollatason kalibrointi suunniteltiin tehtäväksi käyttöliittymän asetusvalikosta kelaamalla kuula veden pinnalle ja painamalla nollausnäppäintä.

Kuten luvun 3.3.1 teoriassa kerrottiin, tulee pulssianturissa olla kaksi kanavaa, jos siltä halutaan saada tieto myös pyörimissuunnasta. Takilan toiminnan kannalta pyörimissuunnan selvittäminen oli oleellinen asia, sillä kuulaa voitiin liikutella kumpaan suuntaan tahansa. Tämä tieto saatiin induktiivisilta antureilta poraamalla niille takilan runkoon reiät siten, että anturit olivat pulssikiekon suhteen 90° vaihesiirrossa toisiinsa nähden. Tämän vaihesiirron toteuttavat mitat selvitettiin liitteen 3 sivulla 2 olevan kelan kiekon valmistuskuvan mittoja hyödyntämällä.

Myöhempää sähkökytkentöjen tekemistä varten tuli induktiivisten antureiden oikea kytkentätapa selvittää. Tämä tehtiin TAMKin automaatiolaboratoriossa virtalähdettä, johtimia ja yleismittaria käyttäen. Todettiin, että antureiden kolmesta johtimesta kaksi olivat 12 ja 0 voltin virtajohtimet, ja kolmas oli signaalijohdin. Anturit tarvitsivat siis toimiakseen 12 voltin jännitteen, eikä niille riittänyt mikrokontrollereiden käyttämä 5 voltin ohjausjännite. Anturilta tuleva signaali tulikin jotenkin pudottaa 12 voltista 5 voltin tasolle. Tähän päätettiin käyttää tyypillisiä 7805-regulaattorikomponentteja, jotka muuttivat antureilta tulleet 12 voltin ohjaussignaalit mikrokontrollerille sopiviksi 5 voltin signaaleiksi. Kaikkien kytkentöjen suunnittelu ja toteutus esitetään tarkemmin luvussa 5.5.

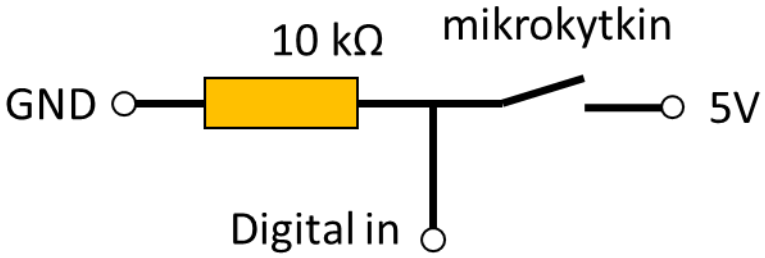
5.2.2 Automaattinen ylöskelauksen pysäytys

Jotta takilasta saatiin käytettävyydeltään turvallisempi ja varmempi, piti siihen suunnitella toiminto, joka esti kuulan kelauksen liian ylös kiinni väkipyörään, jolloin naru pingottui kuulan ja kelan väliin ja saattoi mahdollisesti aiheuttaa vahinkoa takilan rakenteille. Vaikka kuulan syvyystieto oli kelan anturoinnin perusteella selvillä, ei tätä tietoa voitu

aina käyttää moottorin pysäyttämiseen kuulaa liian ylös kelattaessa, sillä nollatasoa kalibroitaessa tuli kuulaa pystyä kelaamaan vapaasti kumpaankin suuntaan. Tällöin kuulan siis pystyi epähuomiossa kelaamaan väkipyörään kiinni.

Jo takilan suunnittelun alkuvaiheista saakka ajatuksena oli kiinnittää väkipyörään mikrokytkin, joka mekaanisesti havaitsisi kuulan nousevan liian korkealle. Vaikka tähän ratkaisuun päädyttiinkin, harkittiin myös muita vaihtoehtoja. Eräs varteenotettavimmista oli magneettikytkimen kiinnittäminen väkipyörän sisään ja magneetin kiinnittäminen takilan narun sisään. Mikrokytkimen kuitenkin ajateltiin olevan yksinkertaisempi vaihtoehto, sillä magneetin kiinnittäminen narun olisi ollut hankalaa, eikä käyttäjä olisi sen olemassaoloa välttämättä tiedostanut, mikäli tämä halusi vaihtaa kelaan oman narunsa, jolloin magneetti olisi hävinnyt vanhan narun mukana. Mikrokytkimen tarvitsema suurempi naruun kiinnitettävä osa, joka sen vartta liikuttaisi, olisi sen sijaan ollut näkyvämpi vaihtoehto.

Mikrokytkin suunniteltiinkin osaksi kaikkia sähkökytkentöjä heti alusta lähtien, kuten liitteestä 2 voi havaita. Koska kyseessä on sulkeutuva kosketin, Arduinon kanssa mikrokytkimen todettiin käyttäytyvän tavallisen painonapin tapaan, eli se tarvitsi virtapiiriinsä yhden 10 kΩ alasvetovastuksen, kuten alla olevasta kuvasta näkee. Sisääntuloportti siis pidettiin sen avulla normaalisti nollapotentiaalissa, mutta mikrokytkimen kontaktori päästi portille 5 voltin jännitesignaalin sen kytkeytyessä päälle. Tämä kytkentä liitettiin osaksi myöhemmin luvuissa 5.5 ja 5.5.1 kuvattuja itse rakennettuja piirilevyjä.



Kuva 19. Mikrokytkimen ja muiden painonapin tavoin toimivien komponenttien kytkentätapa.

Vaikka mikrokytkin saatiin virtapiirin ja myös ohjelman osalta toimimaan halutulla tavalla, tuotti sen kiinnittäminen väkipyörään hankaluuksia. Mikrokytkimen vartta oli yl-

lättävän vaikea saada liikkumaan hallitusti millään tavalla, sillä vaikka naruun olisi kuulan yläpuolelle kiinnittänyt jonkinlaisen osan, joka olisi mikrokytkimen vartta liikuttanut, olisi se pakostikin liikkunut kytkimen varren ohi mahdollisesti jopa vääntäen varren hajalle, sillä moottori ei olisi pysähtynyt riittävän nopeasti tarpeeksi lyhyellä matkalla. Ylipäätään sopivan kiinnityskohdan löytäminen mikrokytkimelle osoittautui hankalaksi, sillä väkipyörän sisässä ei ollut juurikaan ylimääräistä tilaa, ja sen ulkopuolelta kytkimen varsi ei ulottunut kunnolla narulle saakka. Näistä syistä mikrokytkimen onnistunut kiinnittäminen jäi tekemättä, ja alettiin harkita sen jättämistä kokonaan pois, jolloin kuulan kelaaminen liian ylös ja siitä aiheutuva takilan vahingoittaminen jäisi käyttäjän vastuulle.

5.2.3 Aktiivinen syvyysseuranta

Aktiivisessa syvyysseurannassa sähkötakila pitää kuulan ennalta määrätyllä etäisyydellä järven pohjasta, vaikka järven syvyys muuttuisikin veneellä eteenpäin ajettaessa. Syklisessä syvyysseurannassa määritellään kaksi tällaista etäisyyttä, joiden välillä takila liikuttaa kuulaa tietyn aikavälin välein. Takilalle annettava syvyyden tavoitearvo saadaan siis yksinkertaisesti kaavalla

$$D = D_L - d \tag{4}$$

missä D on takilan tavoittelema kuulan syvyys, D_L on järven syvyys sekä d kuulan etäisyys pohjasta. Koska etäisyys pohjasta määritellään käyttöliittymän kautta käyttäjän toimesta, jää tuntemattomaksi muuttujaksi järven syvyys, jonka arvo tulee selvittää jonkinlaisella anturoinnilla.

Syvyysseurantaan tarvittava tieto järven syvyydestä suunniteltiin saatavan luvussa 2.1 kuvattua NMEA 0183 -standardia tukevan kaikuluotaimen NMEA-liitännän kautta. Käytännössä tämä tarkoitti kahden johtimen kytkemistä jonkinlaiseen virtajohtimen yhteyteen kiinnitettävään liittimeen, jolta syvyystiedon vastaanottamista varten tarvittavat gnd- ja rx-piuhat kuljetettaisiin ohjainkotelon rungon seinämässä olleen vesitiiviin läpivientikumin kautta mikro-ohjaimen I/O-pinneille. Suunnittelussa tuotti kuitenkin ongelmia oikeanlaisen liitintyypin valitseminen, sillä NMEA 0183 -standardilla ei ole standardoitua liitintä, vaan eri takiloissa on käytössä erilaisia liittimiä. Myöskään NMEA 0183 -stan-

dardia tukevaa kaikuluotainta ei löydetty testattavaksi kuin lyhyeksi aikaa, mikä entisestään vaikeutti järven syvyystiedon anturoinnin suunnittelua. Sopivan liitäntätavan valitseminen jäikin vielä tämän opinnäytetyön osalta tekemättä, mutta sitä tullaan vielä pohtimaan tarkemmin, kun takilasta aletaan suunnitteleman myyntiin kelpaavaa versiota tässä raportissa kuvatuista testeistä saatujen tulosten pohjalta.

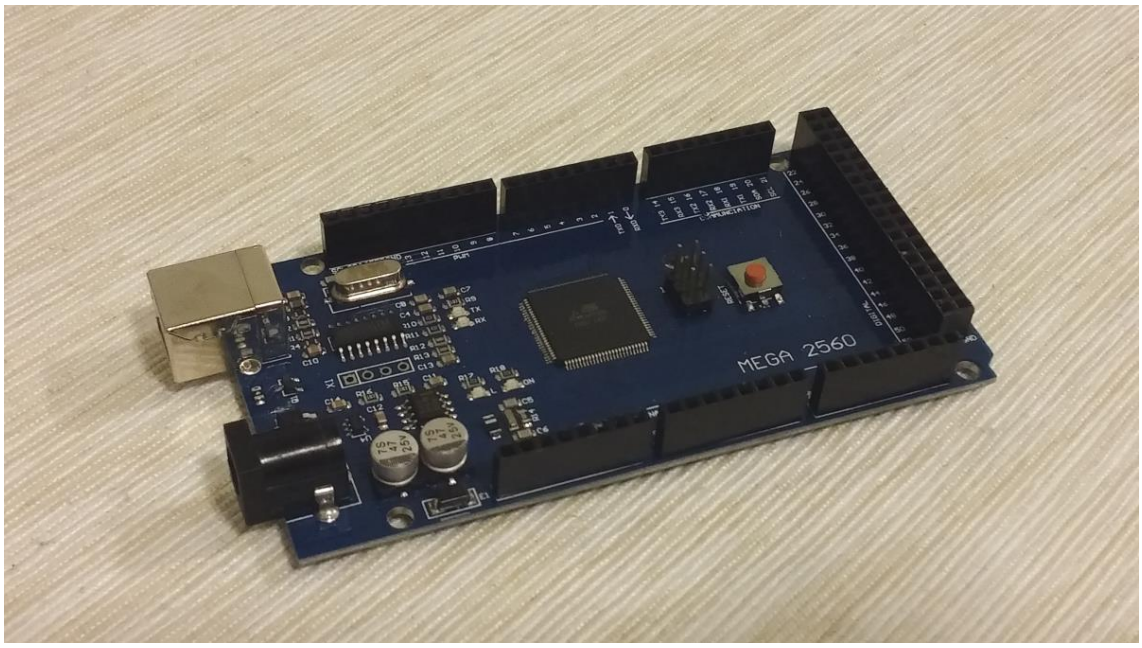
5.3 Mikro-ohjain

Kuten luvussa 4.1 kerrottiin, haluttiin sähkötakilaan aluksi rakennetun yksinkertaisen releohjauksen tilalle kehittyneempi mikrokontrolleripohjainen ohjain, jolla pystyttiin tuomaan takilaan monipuolisempaa toiminnallisuutta ja älyä. Tunnettavuutensa vuoksi kahdeksi hyväksi vaihtoehdoksi nousivat heti Arduino ja Raspberry Pi. Näistä vaihtoehdoista päädyttiin Arduinoon, sillä pieneksi tietokoneeksi luonnehdittuun Raspberryyn verrattuna se täytti paremmin takilan ohjaukselle asetetut vaatimukset, mutta ei ylittänyt niitä liikaa. Mikrokontrollereina Arduinot on tarkoitettu nimenomaan erilaisten laitteiden ohjaukseen ja osaksi pieniä elektroniikkaprojekteja, kun taas Raspberry tietokoneena vaatii kokonaisen käyttöjärjestelmän toimiakseen (J. Bruce 2013). Vaikka Raspberryn avulla olisi varmasti pystynyt toteuttamaan samat toiminnot kuin Arduinolla, olisi sen käyttäminen ollut turhan monimutkaista helppokäyttöiseksi ja helposti opittavaksi suunniteltuun Arduinoon verrattuna.

Aluksi mikro-ohjaimena käytettiin Verkkokauppa.com -sivuston kautta tilatun Arduino-aloituspakkauksen mukana tullutta Arduino Unoa. Pakkauksessa tuli myös mukana monia muita elektroniikkakomponentteja, johtimia sekä ohjekirja, joiden avulla mikrokontrollerin toimintaan oli helppo tutustua. Unon avulla tehdyt testit antoivat hyvän käsityksen siitä, mitä kyseisellä mikrokontrollerilla pystyy ja ei pysty tekemään. Testejä tehdessä ja ohjauksen toimintaan liittyvien ideoiden paisuessa kävi ilmi, että Uno ei kuitenkaan ollut riittävä ohjain sähkötakilaa varten. Tähän oli kaksi selkeää syytä; Unon 32 kilobitin muisti ei ollut riittävän suuri koko käyttöliittymän ohjelmointia varten, eivätkä sen I/O-pinnit riittäneet kaikkien tarvittavien komponenttien, kuten H-sillan, radiovastaanottimen ja kosketusnäytön, kytkemistä varten. Testausvaiheeseen se kuitenkin oli hyvä valinta.

Koko moottorinohjaimen ja käyttöliittymän rakentamista varten päädyttiin käyttämään suurempaa kuvassa 20 näkyvää Arduino MEGA -mikrokontrolleria, jonka 256 kilobitin

muisti riitti paremmin käyttöliittymän ohjelmoimiseen, ja jossa oli riittävästi I/O-pinnejä muiden komponenttien ohjaamiseksi. Uno olisi ollut MEGAa paljon pienikokoisempi, mutta tämä ei ollut suuri ongelma, sillä kummankin komponentin uskottiin mahtuvan takilan ohjainkotelolle suunniteltuun tilaan helposti. Käytössä olleen muistin määrää lukuun ottamatta kummankin mikrokontrollerin ohjelmoiminen oli täysin samanlaista, ja ainoa ohjelmointia tehdessä huomioitava ero oli Arduino IDE -kehitysympäristön asetusten muuttaminen kulloinkin kyseiselle Arduinolle sopivaksi. Työn aikana Arduinoja jouduttiin hankkimaan yhteensä kolme kappaletta rikkoutumisten takia. Näistä kaksi tilattiin kiinasta halvalla, ja niistä toinen oli ilmeisesti jo saapuessaan viallinen aiheuttaen ongelmia sitä ohjelmoitaessa. Toinen vaurioitui myöhemmin tehdyn kytkentävirheen takia. Kolmas MEGA saatiin ilmaiseksi opettaja Seppo Mäkelältä TAMKin automaatiolaboratoriosta työn suorittamista varten.



Kuva 20. Sähkötakilan mikrokontrollerina käytetty Arduino MEGA.

5.4 Käyttöliittymä

Liitteen 1 vaatimusluettelossa määriteltiin, että käyttäjän haluttiin pystyvän käyttämään sähkötakilaa sekä takilan käyttöpaneelissa olevia painikkeita painamalla ja näyttöä seuraamalla että etänä kaukosäätimen avulla. Kauko-ohjaimesta muodostuikin heti tärkeä osa takilan käyttöliittymää ohjainkotelossa olevien komponenttien lisäksi. Syvyyslukeman näyttämistä ja muita toimintoja varten tuli kotelon kyljessä olla jonkinlainen näyttö,

sekä painikkeita näiden toimintojen, kuten vaikkapa ylös- ja alaskelauksen, toteuttamista varten. Näytölle ohjelmoidun valikkorakenteen tuli noudattaa luvussa 3.5 kuvattuja hyvälle käyttöliittymälle asetettuja vaatimuksia. Erillistä virtakatkaisijaa ei käyttöliittymän oheen tarvittu, vaan koska Arduinon pystyi sammuttamaan ja käynnistämään yksinkertaisesti virran kytkemällä riitti, että ohjain otti virtansa helposti irrotettavan liittimen kautta ja sammui, kun tämän liittimen kytki irti veneen akusta.

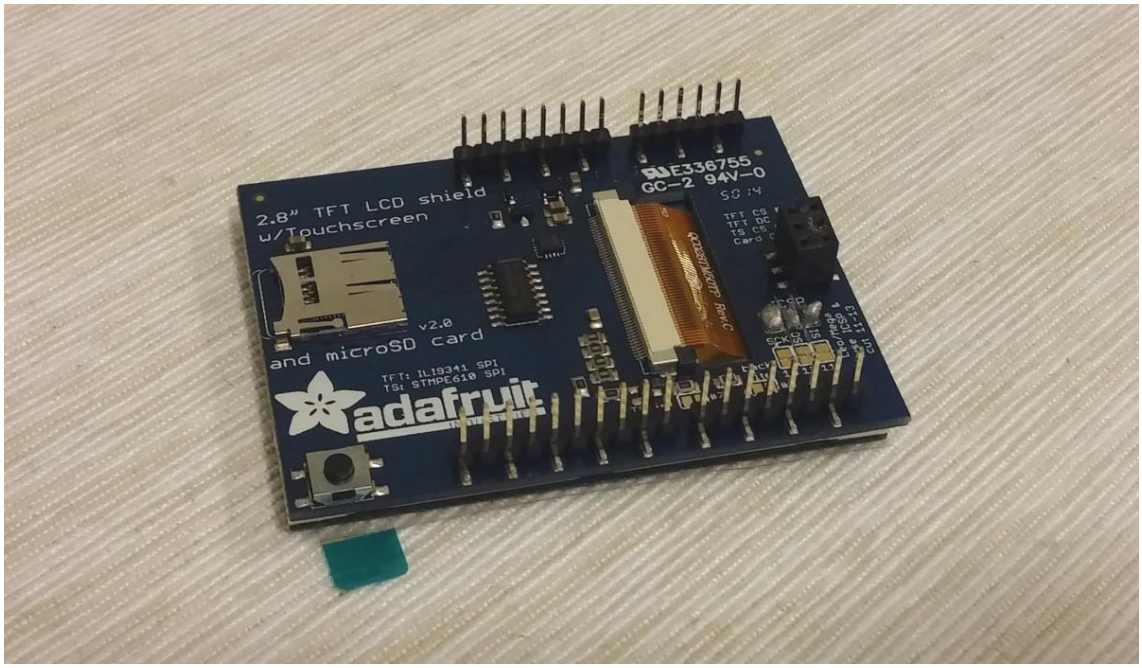
5.4.1 Näyttö ja painikkeet

Aluksi käyttöliittymän näyttönä suunniteltiin käytettävän luvussa 5.3 mainitun Arduino-aloituspakkauksen mukana tullutta kaksirivistä 2 x 16 merkin kokoista LCD-näyttöä, jolle pystyi helposti tulostamaan erilaisia symboleja Arduinon omaa koodikirjastoa käyttäen. Näytössä oli yhteensä 16 I/O-pinniä, joista 12 piti ottaa käyttöön. Osa pinneistä kytkettiin suoraa Arduinon pinneihin, mutta osa 5 voltin jännitteeseen tai maahan. Näyttö tarvitsi toimiakseen myös 220 ohmin vastuksen yhden pinnin ja 5 voltin jännitteen välille kytkettynä.

Näytön lisäksi ohjainkotelon pintaan päätettiin myös kiinnittää jonkinlainen näppäinpaneeli, jonka avulla takilan valikossa navigoitaisiin. Sopiva ohut nelipainikkeinen näppäinpaneeli löydettiin ebay.com -verkkosivustolta muutaman euron hintaan, ja niitä tilattiin samalla kertaa useampi kappale. Näppäinpaneeli vaati toimiakseen 5 I/O-pinniä; 4 signaalipinniä ja 5 voltin jännitepinnin. Väkipyörän mikrokytkimen tapaan myös nämä painonapit tarvitsivat kytkentäänsä 10 kilo-ohmin vastukset, yhden kutakin painiketta kohti. LCD-näytön ja näppäinpaneelin pinnien suuren määrän takia mikrokontrolleriksi jouduttiin valitsemaan Arduino MEGA Unon sijaan, sillä Unon 14 digitaaliporttia meinasivat loppua jo näiden kahden komponentin kohdalla. Molemmat komponentit näkyvät kuvassa 21.

Kuva 21. Kaksirivinen LCD-näyttö ja näppäinpaneeli ensimmäiseen ohjainkotelotulosteeseen kiinnitettynä.

Kuten kuvasta 21 näkee, ei tämä käyttöliittymäratkaisu ollut järin imarteleva takilan ulkonäön kannalta. Näppäinpaneelin kiinnittäminen osoittautui hankalaksi, ja LCD-näyttö oli ahdas ja sille oli vaikea saada mahtumaan kaikkea oleellista tietoa. Lisäksi tälle näytölle ohjelmoitu käyttöliittymä oli todella vaatimattoman näköinen, sillä näytölle ei saanut piirrettyä eri värejä lainkaan. Toisinaan käyttäjän oli myös hankalaa tietää intuitiivisesti, mitä toimintoja eri painikkeet toteuttivat. Näistä syistä takilassa päädyttiinkin lopulta käyttämään adafruit.com -sivustolta löydettyä Arduinolle kehitettyä 2,8 tuuman 240 x 320 pikselin kokoista resistiivistä kosketusnäyttöä (kuvassa 22), joka tilattiin sivustolta toimitus- ja tullikuluineen noin 50 € hintaan. Kosketusnäyttö oli selvästi monipuolisempi ja monimutkaisempi kuin aikaisempi LCD-näyttö, ja tästä syystä se myös vaati enemmän I/O-pinnejä toimiakseen. Arduino MEGA:ssa pinnit kuitenkin onneksi riittivät myös tämän näytön käyttämistä varten, sillä näyttö oli suunniteltu shield-tyyliseksi ja kiinnitettäväksi suoraa joko Unon tai MEGA:n päälle, joten ongelmaa ei pinnien loppumisen suhteen syntynyt.



Kuva 22. Adafruitin valmistaman resistiivisen TFT-kosketusnäytön tausta.

Kosketusnäyttöä käyttämällä saavutettiin monia selkeitä etuja. Näyttävyydeltään tämä komponentti oli paljon edustavampi edelliseen LCD-näyttöön verrattuna, sillä se oli kooltaan suurempia ja se pystyi piirtämään värejä (yhteensä 262 000 eri värisävyä), mikä mahdollisti selkeämmän valikkorakenteen suunnittelun. Koska kyseessä oli kosketusnäyttö, ei erilliselle näppäinpaneelille enää ollut tarvetta, mikä teki käyttöpaneelista yksinkertaisemman. Myös tämä vaikutti osaltaan valikkorakenteen selkeyden paranemiseen, sillä nyt näytölle voitiin ohjelmoida painikkeita paljon vapaammin ja selkeämmin aseteltuna. Käyttöliittymästä saatiin siis paljon helppokäyttöisempi. Myös takilan kaupallista arvoa saatiin oletettavasti kasvatettua, sillä kalastuspiireissä tuotteen ulkonäkö on usein tärkeä asia, ja kosketusnäyttö on edustavamman ulkonäkönsä lisäksi myös nykyaikaisempi ratkaisu jo vanhanaikaisiksi jääviin LCD-näyttöihin verrattuna. Yhtenä valintatekijänä voidaan sanoa olleen myös työn tekijöiden halu oppia uutta tähän tekniikkaan liittyen.

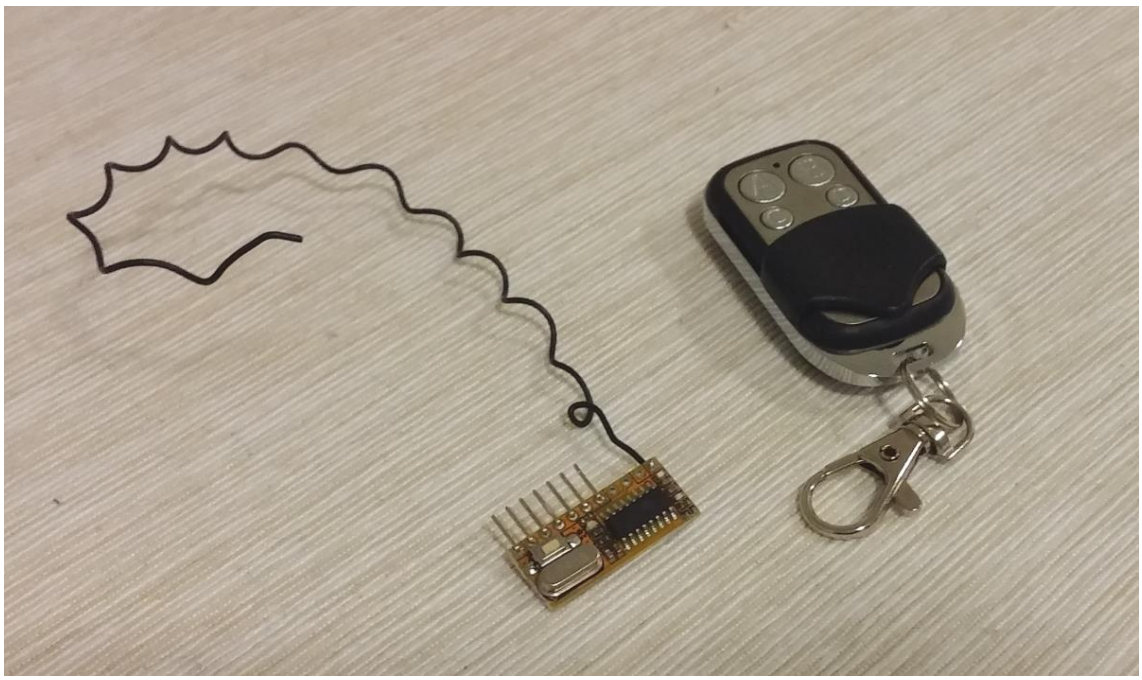
Näyttöä olisi ollut saatavilla myös kapasitiivisella kosketuksen tunnistuksella. Resistiivisen kuitenkin todettiin olevan takilaa varten sopivampi valinta, sillä resistiivinen kosketusnäyttö on halvempi, kestää paremmin pölyä ja vettä sekä havaitsee kosketuksen myös hansikkaita tai kosketuskynää käytettäessä, mitkä ovat tärkeitä huomioitavia asioita kalastusolosuhteita ajatellen. Kosketusherkkyydeltään se ei kuitenkaan ole kovin hyvä, vaan näyttöä tulee painaa melko voimakkaasti, ja sen kontrasti on myös huonohko. Se ei myös-

kään tue useampaa yhtäaikaista painallusta, mikä toisaalta ei ollut ongelma käyttöliittymän suunnittelun kannalta. Kapasitiivisten näyttöjen heikkoudet ja vahvuudet ovat kutakuinkin päinvastaiset resistiivisiin näyttöihin verrattuna. (Techexplainer 2012.)

5.4.2 Kauko-ohjaus

Jo sähkötakilaprojektin alkuvaiheissa syntynyttä ideaa takilan käyttämisestä kauko-ohjaimella pidettiin yhtenä tuotteen tärkeänä myyntivalttina. Työn tekijöiden käsityksen mukaan markkinoilla ei vielä ole sähkötakilaa, jossa olisi erillinen kauko-ohjain, ja tämän ajateltiin olevan hyvä tilaisuus tuotteen kaupallisen arvon lisäämiseksi. Päätettiinkin, että tärkeimmät takilan toiminnot, eli ylös-, alas- ja pintaankelaus sekä hätäpysäytys, tuli pystyä toteuttamaan kaukosäädintä käyttäen. Näin takilasta saatiin helppokäyttöisempi, sillä ahtaassa veneessä kalastajan ei välttämättä tarvitse liikkua takilan ääreen, vaan hän voi käyttää sitä myös muualta veneestä tärkeimpien toimintojen ollessa aina hänen kätensä ulottuvilla.

Arduino osoittautui suosionsa ja laajennettavuutensa ansiosta jälleen hyväksi mikrokontrollerivalinnaksi, sillä käytännössä kauko-ohjauksen toteuttaminen ainakin prototyyppivaiheen vaatimalla toimintavarmuudella oli mahdollista ebay.com -sivustolta löydettyä Arduinolle tarkoitettua nelipainikkeista radiotaajuuksilla toimivaa kauko-ohjainta ja sen vastaanotinpiirilevyä käyttäen (kuvassa 23). Myös näitä tilattiin 2 kappaletta mahdollisia rikkoutumisia ajatellen. Hintaa yksittäisellä ohjain-vastaanotinparilla oli noin 10 €. I/O-pinnejä vastaanotin tarvitsi yhteensä 6 kappaletta; 5 voltin jännitepinnin, maapinnin sekä 4 signaalipinniä kaukosäätimen painikkeiden painallusten tunnistamiseen.



Kuva 23. Takilaa varten tilattu radiovastaanotin ja sen kaukosäädin.

Myös muita kauko-ohjaimia harkittiin, mutta suurimmat erot vaihtoehtojen välillä vaikuttivat olevan painikkeiden lukumäärässä. Koska käyttöliittymä haluttiin tältäkin osin pitää yksinkertaisena, todettiin 4 painikkeen olevan tarpeeksi haluttujen toimintojen toteuttamiseksi. Kauko-ohjaimien toimintaetäisyyksiin ei juuri kiinnitetty huomiota, sillä kaikkien oletettiin toimivan hyvin muutaman metrin säteellä.

5.4.3 Käyttöliittymän visuaalinen ilme

Käyttöliittymää varten suunniteltiin Excel-taulukkolaskentaohjelmistoa apuna käyttäen kaksi eri valikkorakennetta: toinen aluksi käytetylle LCD-näytölle ja painikkeille sekä toinen kosketusnäytölle. Kumpaakin suunniteltaessa pidettiin mielessä valikon yksinkertaisuus ja intuitiivinen käyttökokemus, sekä muut luvussa 3.5 kuvatut asiat. Asiaan haluttiin panostaa, jotta tuotteesta saatiin kaupallisesti mahdollisimman houkutteleva. Näyttöjen erilaisuudesta johtuen myös valikoista tuli toisiinsa nähden hyvin erilaiset, vaikka ne toteuttivatkin osittain samat toiminnot. Ainoa toiminto, mitä ei vielä LCD-näytön valikkoa ohjelmoitaessa ollut suunniteltu, oli aktiivinen syvyysseuranta kaikuluotainyhteyttä hyödyntäen, ja tämä ominaisuus puuttuukin ensimmäisestä valikkorakennesuunnitelmasta kokonaan.

Ensimmäinen, LCD-näytölle suunniteltu, valikkorakenne näkyy liitteessä 4. Koska näyttö oli vain 16 merkkiä leveä ja kaksirivinen, tuli tilankäytön suhteen olla tarkkana valikkoa suunniteltaessa. Valikossa päädyttiin hieman polkupyörän matkamittareiden valikoita muistuttavaan lähestymistapaan, eli kahta painiketta painamalla pystyttiin siirtymään valikosta toiseen, ja niistä jokainen oli tarkoitettu yhden tietyn toiminnon toteuttamiseen tai alivalikkoon siirtymiseen. Valintojen kuittaaminen, ylös ja alas kelaaminen sekä muut toiminnot tehtiin kahdella muulla painikkeella. Näytön ylärivi varattiin lähes kokonaan kunkin valikon otsikkoa varten, jotta valikon nimen näyttämiseksi jäi tarpeeksi tilaa. Alariville tulostettiin navigointia helpottavat nuolisymbolit, syvyyslukema sekä kullekin valikolle ominaisia muita tietoja. Toisin sanottuna valikon eri elementit pyrittiin pitämään aina niille tarkoitetuilla paikoilla, jolloin käyttökokemuksesta tuli intuitiivisempi. Valikko yritettiin pitää mahdollisimman yksinkertaisena ja tärkeimmät toiminnot, eli manuaalikelaus ja haluttuun syvyyteen kelaus, sijoitettiin heti ylimmän tason valikkoihin, kun taas asetukset oli jäsennelty omaan asetukset-ylävalikon kautta avautuvaan alivalikkoonsa. Tätä valikkorakennetta suunniteltaessa myös päätettiin kauko-ohjaimen painikkeiden toteuttamat toiminnot, mitkä nekin käyvät ilmi liitteen suunnitelmasta.

Suurin ongelma tässä valikkorakenteessa oli sen ahtaus ja informaation välittämisen vaikeus. Valikoiden toteuttamia toimintoja oli monesti hankala kuvata täysin ymmärrettävästi, ja esimerkiksi automaattinen pintaankelaus ei ilmennyt siitä lainkaan, vaikka sellainen kyllä sen osaksi ohjelmoitiin (manuaalikelausvalikossa sekä ylös että alas-painikkeita yhtäaikaisesti painamalla). Myös nollataso-niminen asetusvalikko oli toiminnaltaan epäselvän oloinen, ja sen toiminnan ymmärtäminen olisi mahdollisesti vaatinut käyttäjältä erillisen käyttöohjeen lukemista. Koska kyseessä oli yksinkertainen LCD-näyttö, ei sille myöskään pystynyt tulostamaan eri värejä, joten valikko oli todella ”tasapaksun” näköinen eikä tärkeimpien toimintojen erottuvuutta voinut juuri mitenkään vahvistaa. Osittain näistä syistä LCD-näyttö ja painikkeet päätettiinkin korvata kosketusnäytöllä, kuten luvussa 5.4.1 kerrottiin.

Kosketusnäytölle tehty valikkorakennesuunnitelma on liitteessä 5. Tällä kertaa valikoita varten oli paljon enemmän tilaa käytettävissä, ja yksittäiseen valikkoon saatiin mahdutettua enemmän kuhunkin toimintoon liittyvää informaatiota. Esimerkiksi manuaalikelausvalikkoon saatiin mahtumaan sekä syvyyslukema, painikkeet ylös- ja alaskelausta varten, hätäpysäytyspainike sekä painikkeet pintaankelausta ja haluttuun syvyyteen kelausta varten. Tärkeimmät painikkeet pidettiin suurina, jotta niihin oli mahdollisimman helppo

osua. Ainoastaan tekstien mahduttaminen joiden painikkeiden päälle aiheutti tilankäytön kannalta ongelmia. Valikkorakenteesta tehtiin mahdollisimman yksinkertainen käyttämällä päävalikkoa, josta pääsi siirtymään johonkin neljästä alivalikosta. Sama numeronäppäimistövalikko aukesi kaikista painikkeista, jotka oli tarkoitettu numeroarvon syöttämiseen.

Myös tämä uusi valikkorakenne pyrittiin pitämään käytettävyydeltään mahdollisimman intuitiivisena asettelemalla eri elementit loogisesti aina samankaltaisesti eri valikkoihin. Kelaukseen liittyvät nuolipainikkeet pidettiin aina näytön oikeassa reunassa, ja muut toimintopainikkeet vasemmassa reunassa. Takaisinpaluupainike oli aina vasemmassa alareunassa. Ainoa elementti, jonka asettelussa jouduttiin hieman joustamaan, oli syvyyslukema, joka valikosta riippuen asetettiin joko vasempaan yläkulmaan tai oikeaan alakulmaan. Tällä kertaa valikoiden selkeyttä pystyttiin parantamaan myös värejä käyttäen, ja nuolinäppäimissä päätettiinkin käyttää hyvin erottuvaa keltaista väriä. Vihreällä ja punaisella värillä ilmaistiin hyväksyviä ja kieltäviä toimintoja, kuten käyttöliittymissä usein on tapana. Painikkeiden erottuvuutta parannettiin käyttämällä mustaa taustaväriä, jolloin kontrasti painikkeiden ja taustan välillä oli mahdollisimman suuri.

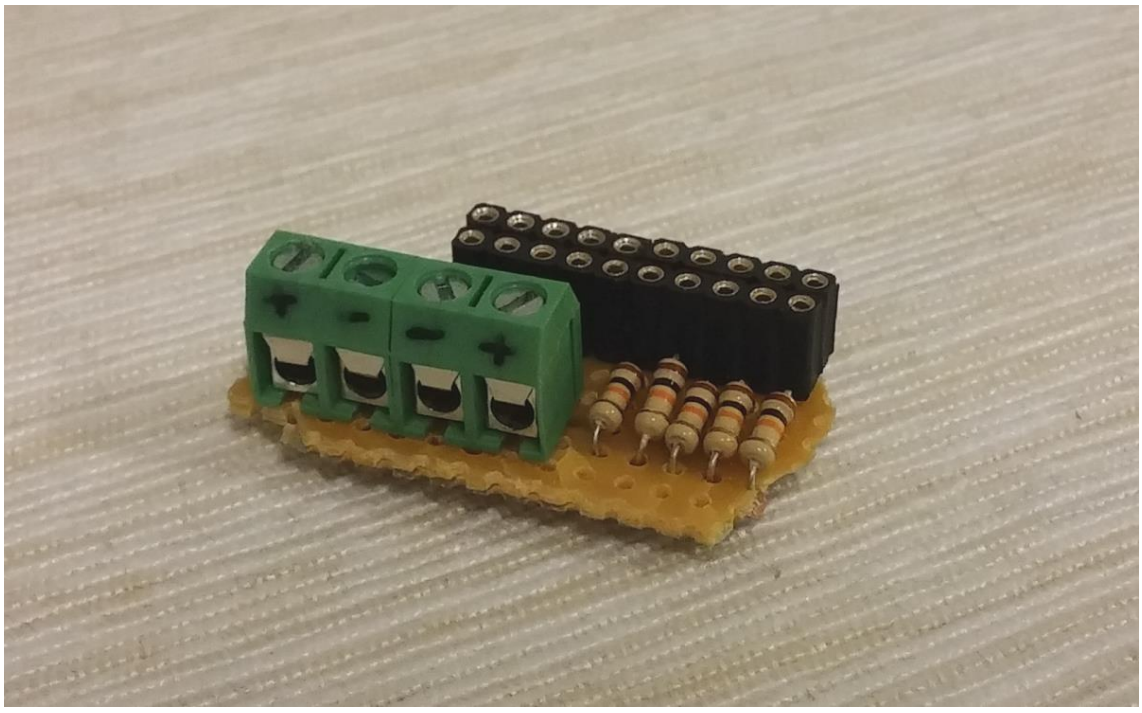
Näiden kummankin valikkorakennesuunnitelman tekeminen Excelillä oli huomattava apu valikoita ohjelmoitaessa. Kuvista pystyi suoraa näkemään, mitä elementtejä tai symboleita mihinkin kohtaan näyttöä tuli asettaa. Etenkin lopullista kosketusnäytön valikkorakennetta ohjelmoitaessa suuri apu oli Excelin ruudukkojaottelusta, jonka avulla pystyi selvittämään eri elementtien tarkat koordinaatit ja mitat todella vaivattomasti.

5.5 Kytkennät

Jo luvussa 4.1 kuvattua releohjausta rakentaessa huomattiin, että johdotusten tekeminen pelkkiä sähköjohtoja käyttäen tuli olemaan todella sotkuista ja testejä tehdessä paljon aikaa vievää. Varsinkin Arduinoon siirryttyä ongelma paisui suureksi, sillä aina eri komponentteja ja etenkin koko kokonaisuutta testattaessa kytkennät piti tehdä uudestaan, ja koska komponenttien määrä oli suurehko, oli mahdollisuus kytkentävirheiden tekemiseen suuri. Ajan säästämiseksi komponenttien välisiä kytkentöjä varten suunniteltiinkin LCD-näyttöä ja painikkeita käytettäessä erillinen apupiirilevy, ja lopulta kosketusnäyttöä käytettäessä kunnollinen Arduino-shield, joiden avulla johtojen määrää saatiin vähennettyä

ja johdotukset tulivat selkeämmiksi. Samalla näihin piirilevyihin saatiin yhdistettyä joitakin pienempiä komponentteja, kuten vastuksia.

Aluksi LCD-näyttöä ja näppäinpaneelia käytettäessä kytkentöjä varten suunniteltiin kuvassa 24 näkyvä apupiirilevy, jonka kautta eri komponentit kiinnitettiin toisiinsa. Piirilevy rakennettiin itse juottamalla komponentit Bebek -elektroniikkaliikkeestä ostetulle ja sopivan kokoiseksi leikatulle reikäpiirilevylle. Levylle juotettiin tarvittavat 20 kappaletta kytkentäsiltoja johdinten kytkemistä varten, 10 kilo-ohmin vastukset näppäinpaneelin painikkeita ja mikrokytkintä varten, LCD-näytön tarvitsema 220 ohmin vastus sekä yhteensä neljä porttia sisältävät riviliittimet virransyöttöä varten. Vastukset tulivat Arduino-aloituspakkauksen mukana, ja muut komponentit ostettiin Bebekistä muutamalla eurolla. Kytkennät suunniteltiin paremman ohjelmiston puutteessa Exceliä käyttäen (liitteenä 6), ja silloitukset eri komponenttien välille tehtiin juottamalla vierekkäisiä reikäpiirilevyn reikiä yhteen tarpeeksi tinaa käyttäen.



Kuva 24. Apupiirilevy, jolla pyrittiin helpottamaan eri komponenttien kytkemistä toisiinsa ja vähentämään johdotusten määrää.

Vaikka apupiirilevylle saatiinkin juotettua vastukset eikä niitä siten tarvinnut juottaa kiinni epäsiististi itse johtimiin, ei piirilevy yksinkertaistanut kytkentöjen tekemistä tarpeeksi, etenkään LCD-näytöstä kosketusnäyttöön siirtymisen jälkeen. Lisäksi apupiirilevyssä käytetyt liittimet olivat huonoja, sillä johtimet ainoastaan työnnettiin niihin kiinni

eikä niiden kiinni pysymistä ollut varmistettu mitenkään. Piirilevy oli myös todella pieni ja epäkäytännöllisen kokoinen kytkentöjen tekemiseen. Kosketusnäytön kytkentöjä varten suunniteltiinkin kokonaan uusi ja parempi Arduino-shield, jolla apupiirilevy korvattiin kokonaisuudessaan.

5.5.1 Shield-piirilevy

Idea oman erillisen shield-piirilevyn valmistamisesta syntyi lähes heti kosketusnäytön hankkimisen jälkeen, sillä näyttö tarvitsi yli 20 johdinta kytkentöihinsä, edellä kuvattu apupiirilevy oli riittämätön ja samoihin aikoihin päädyttiin pulssianturi toteuttamaan induktiivisia antureita käyttäen, joita varten piti kaksi 7805-regulaattoria juottaa osaksi kytkentää. Myös 12 voltin jännitteen jakelu päätettiin yhdistää osaksi tätä uutta piirilevyä.

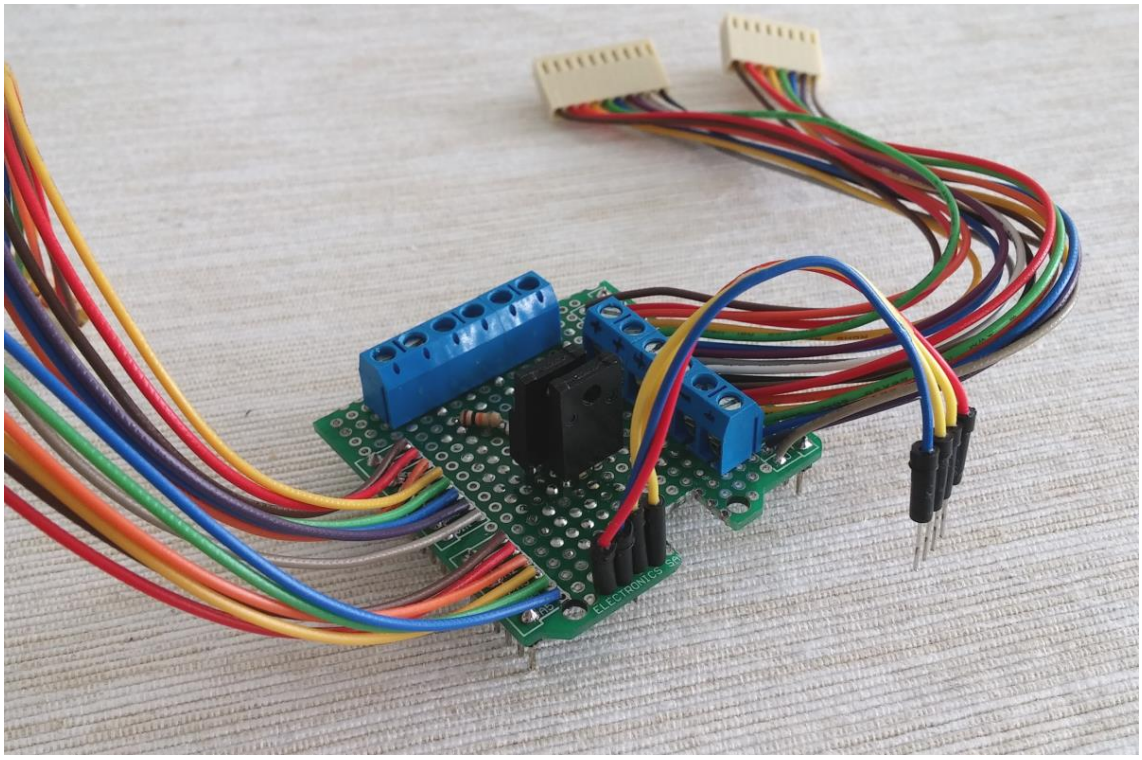
Shield suunniteltiin jälleen Exceliä käyttäen (liitteenä 7). Seppo Mäkelän kautta saatiin tieto, että on olemassa valmiita Arduino Unolle (sekä myös MEGA:lle) tarkoitettuja reikäpiirilevyjä, joille kuka tahansa voi helposti rakentaa oman shieldinsä. Tällainen Unolle tarkoitettu piirilevy saatiinkin TAMKin automaatiolaboratoriosta ilmaiseksi, ja liitteen 7 suunnitelma tehtiin sen pohjalta. Vaikka takilan mikro-ohjaimena olikin Arduino MEGA, sopi tämä Unolle tarkoitettu piirilevy myös MEGA:n päälle. MEGA:lle tarkoitetut reikäpiirilevyt olisivat olleet huonompi valinta, sillä toisin kuin ne, Unon reikäpiirilevy jätti osan MEGA:n I/O-porteista paljaaksi, jolloin shieldiin juotetut johtimet pystyttiin kiinnittämään helposti näihin portteihin ilman, että shield olisi ollut niiden tiellä. Unon reikäpiirilevystä tuli kuitenkin sahaamalla poistaa MEGA:n ICSP-pinnien ja USB-portin kohdalla olleet alueet.

Shieldiin sisällytettiin seuraavat komponentit:

- 2 kappaletta 7805-regulaattoria, jotka laskivat induktiivisilta antureilta tulleet 12 voltin jännitesignaalit 5 voltin tasolle
- 10 kilo-ohmin vastus mikrokytkimen signaalia varten
- 6 kaksiporttista riviliitintä mikrokytkinten, antureiden, akun, H-sillan ja Arduinon kytkemistä varten
- 4 Arduinoon kytkettyä johdinta antureiden signaaleja ja mikrokytkimen 5 voltin jännitettä varten

- pinnit, jotka kiinnittivät shieldin Arduinon I/O-portteihin
- johtimet kosketusnäytön kytkemistä varten.

Kaikki edellä mainitut komponentit saatiin nekin joko Arduino-aloituspakkauksesta tai ilmaiseksi TAMKin automaatiolaboratoriosta. Komponentit pyrittiin piirilevyä suunniteltaessa asettamaan levylle siten, että silloituksista tuli mahdollisimman suorat ja niin, ettei hyppylankoja jouduttu käyttämään lainkaan. Antureiden johtimet asetettiin levyn reunalle, mistä ne oli helppo kytkeä Arduinoon, ja riviliittimet juotettiin kahdelle muulle reunalle myöskin kytkemisen helppous mielessä pitäen. Komponenttien väliset silloitukset juotettiin piirilevyn vastakkaiselle puolelle kuparilankaa apuna käyttäen. Juotosten valmistuttua shieldin toiminta testattiin perinpohjaisesti tutkimalla porttien välinen johtavuus ja oikeat jännitetasot yleismittaria käyttäen. Käyttöönoton jälkeen tästä komponentista ei aiheutunutkaan minkäänlaisia ongelmia, vaan se toimi suunnitellusti ja teki komponenttien toisiinsa kytkemisestä huomattavasti siistimpää ja nopeampaa. Valmis shield näkyy alla olevassa kuvassa.



Kuva 25. Kytkentöjä varten valmistettu shield-piirilevy. Sivuilta lähtevät johtoniput ovat kosketusnäytön kytkemistä varten.

Kosketusnäytön tarvitsemat ICSP-pinnit kytkettiin Arduinon ICSP-pinneihin johdinnipulla, jota ei ollut tarvetta yhdistää osaksi shieldiä. Myös Arduinon ja kauko-ohjaimen

vastaanottimen sekä Arduinon ja H-sillan väliset liitännät jätettiin shieldin ulkopuolelle, sillä koko vastaanotin kytkettiin suoraa Arduinon I/O-pinneihin ja H-sillan tarvitsemat 3 johdinta (2 ohjaussignaalia ja maa) kytkettiin ICSP-johdinten tapaan Arduinon ja H-sillan välille ilman tarvetta käyttää shieldiä niiden välissä.

5.5.2 Virransyöttöpiiri ja liitännät

Koska sähkötakila on osa moottoriveneen laitteistoa, oli alun alkaen selvää, että sen tuli saada sähkönsä muutaman metrin mittaisen virtajohtimen välityksellä veneen akulta, joka tyypillisesti antaa 12 voltin jännitettä (tässä myös syy sille, miksi takilan moottorin tuli toimia 12 voltilla, eikä esimerkiksi 24 voltilla). Takilan virtaliittimenä suunniteltiin käytettävän autoista tuttua niin sanottua tupakansytytinpistoketta, joka olisi helppo liittää veneen akkuun, ja joka myös kestäisi roiskevettä hyvin. Yksi tähän valintaan johtanut tekijä oli myöskin liittimen yleisyys. Lisäksi monissa veneissä on tupakansytytinliitäntä jo valmiiksi. Tällainen pistoke hankittiin, mutta sitä ei otettu käyttöön vielä testausvaiheessa, vaan johdinten päissä käytettiin myös muita kulloiseenkin tilanteeseen ja käytössä olleeseen akkuun sopivia liitintyyppejä.

Kuten luvussa 5.2.3 kerrottiin, jäi NMEA-liitännän rakentaminen tekemättä, koska sopivaa liitintyyppiä ei löydetty. Työn loppuvaiheessa ajatukseksi kuitenkin nousi NMEA-liitännän kahden paljaan piuhan vetäminen ilman minkäänlaista liitintä tupakansytytinpistokkeen yhteyteen nelinapaista johdinta käyttäen. Usein veneen kaikuluotain sijaitsee lähellä veneen akkua, ja takilan käyttäjä voisi itse yhdistää kaksi NMEA-johdinta kaikuluotaimensa liittimiin tai johtimiin parhaaksi näkemällään tavalla, esimerkiksi jonkinlaista riviliitintä käyttäen. Näin takilaan ei tarvitsisi valita vain yhdentyyppistä NMEA-liitäntää, joka ei välttämättä sopisi kaikkiin kaikuluotaimiin, vaan aktiivinen syvyysseurantaominaisuus olisi käytössä kaikille NMEA 0183 -standardia tukeville kaiuille, vaikka liitoksen tekemisestä tulisikin hankalampaa.

Eräs virransyöttöön liittyvä asia oli virransyöttöjohtimien vetäminen vesitiiviisti takilan ohjainkotelon seinämän läpi. Tähän tarkoitukseen suunniteltiin käytettävän vesitiivistä läpivientikumia, joka kiinnitettäisiin kotelon seinämään porattuun reikään. Tarvittava ve-

donpoisto saavutettiin joko käyttämällä kiristettävää läpivientikumia tai ohjainkotelon sisään lisättyä rakennetta, jota vasten johdin voitiin puristaa kiinni vaikkapa kahta ruuvia ja teräslevyä käyttäen.

Erilliselle virransyöttöpiiriin yhdistettävälle virtakytkimelle ei todettu olevan minkäänlaista tarvetta, sillä takila sai käynnistyä heti, kun tupakansytytinpistokkeen kytki kiinni veneen akkuun. Näin käyttöpaneeli myös pysyi sopivan yksinkertaisena, eikä siihen tullut kosketusnäytön lisäksi mitään ylimääräisiä kytkimiä tai painikkeita. Sopivan kokoista sulaketta kuitenkin suunniteltiin käytettävän, mutta sen tarkkaa mitoittamista ei vielä pystytty tekemään, sillä moottorin virrankulutusta ei tiedetty. 30 ampeerin sulakkeen arveltiin kuitenkin olevan varmasti riittävän suuruinen.

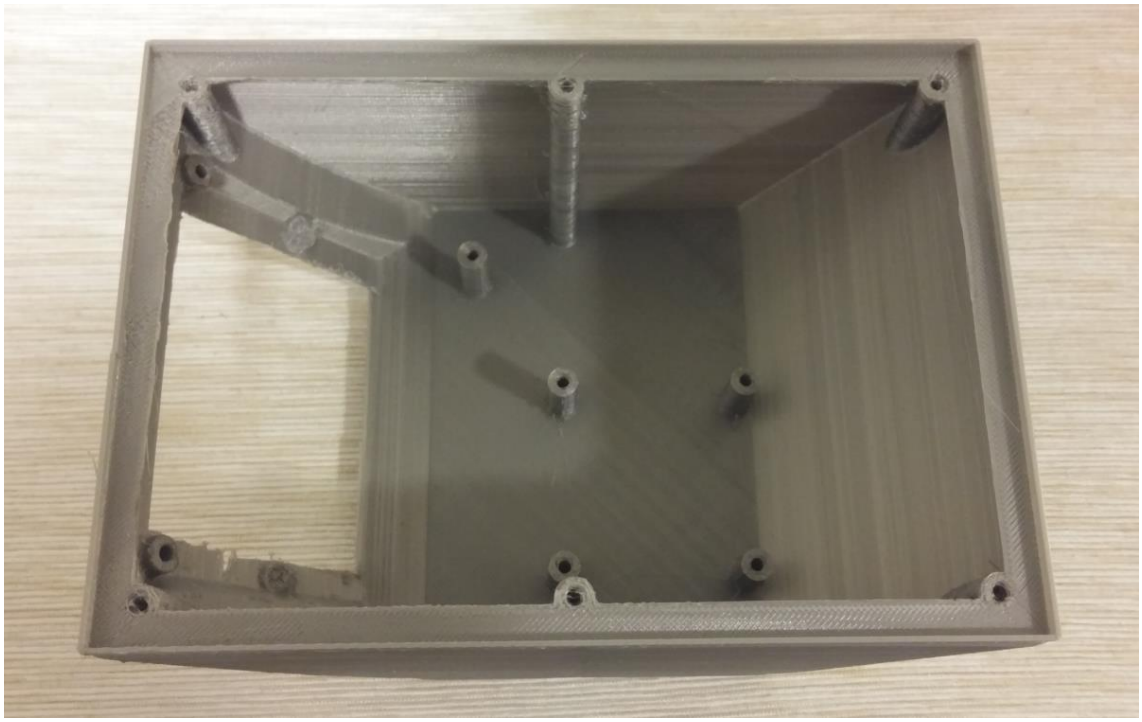
5.6 Ohjainkotelo

Koska sähkötakila suunniteltiin käyttötarkoituksensa vuoksi kosteisiin ympäristöihin, tuli kaikki elektroniikka sulkea vesitiiviiseen koteloon. Kaupallisuutta ajatellen kotelon tuli myös olla näyttävä ja sopia muotoilultaan hyvin osaksi takilan runkoa, johon se kiinnitettäisiin ruuviliitoksin mahdollisesti kumisia aluslevyjä käyttäen. Käyttöliittymän komponentit, eli näyttö ja mahdolliset painikkeet, tuli jotenkin kiinnittää näkyville kotelon ulkopintaan huolehtien edelleen kotelon vesitiiviydestä. Kotelon sisään oli mahduttava mikro-ohjain, H-silta, kauko-ohjaimen vastaanotin, näyttö sekä näiden komponenttien väliset johdotukset. Anturit ja mikrokytkin jäivät kotelon ulkopuolelle, joten niiden johtimet tuli virtajohdinten ja moottorin johdinten lisäksi viedä koteloon vesitiiviisti läpivientikumia käyttäen.

Koska kotelolle asetetut vaatimukset olivat etenkin sen muotoilun suhteen melko kovat, päätettiin se valmistaa 3D-tulostamalla TAMKin tulostimia käyttäen. Toisaalta tieto käytettävissä olevista tulostimista antoi vapauksia suunnitella kotelon geometria monimutkaisemmaksi, kuin mitä perinteisemmillä tuotantomenetelmillä olisi ollut mahdollista valmistaa. Kotelo 3D-mallinnettiin Autodesk Inventor -ohjelmiston avulla, ja tulostamista varten tiedosto muunnettiin .stl-tiedostomuotoon.

Aluksi kotelo suunniteltiin luvussa 5.4.1 kuvailtua LCD-näyttöä ja painonappeja varten. Sen muodon pohjaksi otettiin jo olemassa oleva takilan alumiinirungon muotoilu, jonka

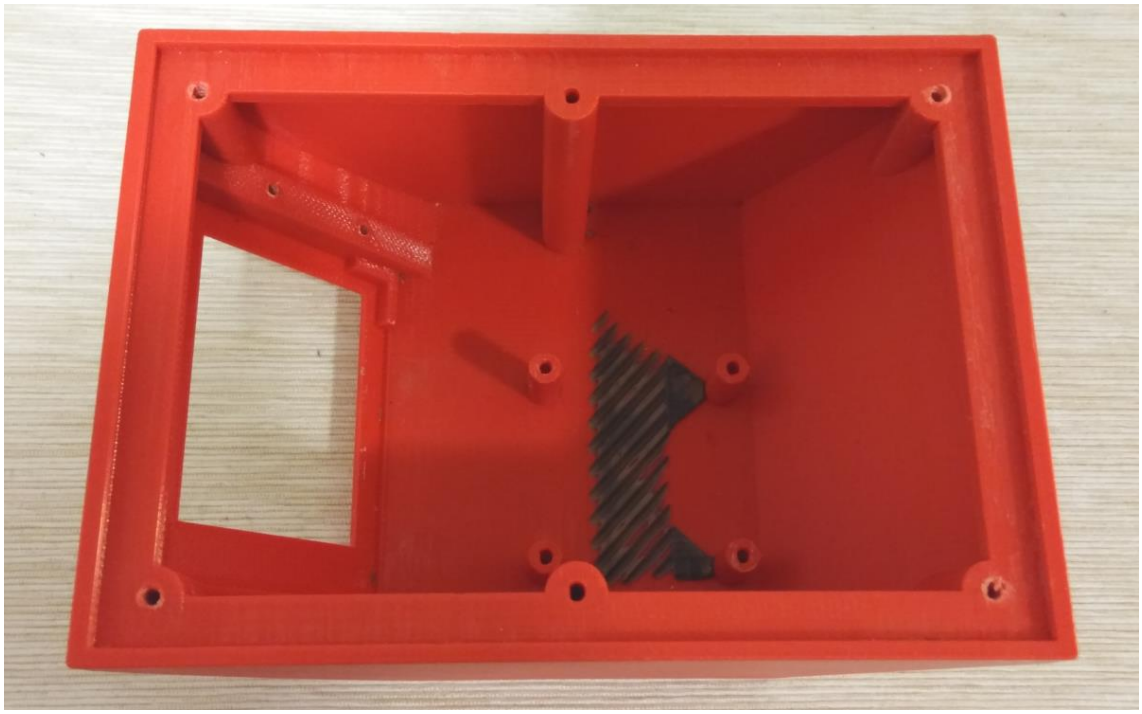
pohjalta kotelo sai kuvassa 26 näkyvän muotonsa. Se mitoitettiin sopimaan rungon pystyseinien väliin, ja sen yläpinnasta tehtiin kaareva. Tälle kaarevalle pinnalle tehtiin korotettu tasopinta, jolle tehtyihin aukkoihin LCD-näyttö ja painikkeet asetettiin. Kotelon sisään tehtiin kiinnikkeet H-sillan ja apuvirtapiirin kiinnittämistä varten. Komponenttien asentamista varten takaseinä jätettiin avoimeksi, ja se suljettiin erillisellä muovilevyllä ruuviliitoksin. Vesitiiviys saavutettiin takaseinämää ympäröivän lipan avulla, jonka muovilevyä vasten olevaan pintaan kiinnitettiin tiivistenauhaa. Kotelon sisään laitettiin myös silikageeliä, joka imi itseensä mahdollisen kosteuden.



Kuva 26. Takilan sähkökomponentteja varten 3D-tulostettu ensimmäinen ohjainkotelo.

Tämä ensimmäinen ohjainkotelo tulostettiin TAMKin Ultimaker-3D-tulostimella abs-muovista. Hintaa tulosteelle tuli materiaalikulujen verran, eli noin 15 €. Huomattiin kuitenkin, että tuloste ei ollut laadultaan riittävän hyvä, sillä kotelon seinämät olivat liian ohuet (alle 1 millimetri) mistä johtuen kotelo ei ollut tarpeeksi kestävä, eikä tulostusjälki ollut kovinkaan edustava vaan todella karhea. Myöskään LCD-näytön ja painikkeiden kiinnitystä ei ollut suunniteltu riittävän hyvin. Tätä koteloa ei päädyttykään juuri enempää testaamaan, vaan sitä käytettiin ainoastaan väliaikaisratkaisuna komponenttien kasassa pitämistä varten.

Kosketusnäyttöön siirtymisen jälkeen tuli ohjainkotelo suunnitella uudestaan. Tällä kertaa näytön kiinnityksen suunnitteluun panostettiin huomattavasti enemmän. Koska näytössä itsessään ei ollut minkäänlaisia kiinnitysreikiä (toisena vaihtoehtona ollut kiinnitysreiät sisältävä näyttö olisi tarvinnut liikaa I/O-pinnejä), piti se tukea kotelon sisältä päin näytölle tehdyn aukon reunoja vasten. Tämä tukeminen tehtiin liitteessä 3 sivulla 3 näkyvää laserleikattua teräslevyä käyttäen. Koteloon mallinnettiin levyä varten kiinnitysreiät siten, että levy oli helppo asettaa paikoilleen. Kotelon sisäpuolen geometriasta tehtiin sellainen, että näyttö asettui jämäkästi paikoilleen, ja levyn ja kosketusnäytön piirilevyn väliin laitettiin tiivistenauhaa pehmikkeeksi. Vesitiiviys varmistettiin liimaamalla kotelon aukon reunoihin älypuhelinten näyttöjä varten tarkoitettu kosketuskalvo. Näytön kiinnityksen suunnittelun lisäksi kotelon seinämäpaksuutta suurennettiin 3 millimetriin. Tässä vaiheessa apuvirtapiiri oli jätetty pois käytöstä, joten myös sen kiinnitysreikä voitiin poistaa 3D-mallista. Takilan nimi ”Reelin’ Downrigger” lisättiin kotelon etuseinään upotetuin kirjaimin. Muilta osin kotelo pidettiin samanlaisena, kuin ensimmäinenkin versio. Uusi kotelo näkyy kuvassa 27.



Kuva 27. Toinen ohjainkotelo. Kotelon sisäpinnassa näkyy vielä tummaa tukiainetta. Näytön kiinnitysgeometriaa näkyy kuvan vasemmalla puolella näytön aukon ympärillä.

Tämä tuloste saatiin ilmaiseksi TAMKin Stratasys Dimension Elite -tulostimella tulostaen. Tulostuksen jälkeen kappale käytettiin muutaman tunnin lipeäkylvyssä, joka poisti siitä tulostimen käyttämän tukimateriaalin. Välittömästi huomattiin, että tulostusjälki oli

paljon sileämpi ja edustavampi, ja että seinämäpaksuuden kasvattaminen tukevoitti kotelon rakennetta huomattavasti. Myös kosketusnäyttö sopi paikoilleen halutulla tavalla, ja teräslevy tiivistenauhoineen painoi sen tarpeeksi lujasti kotelon sisäpintaa vasten. Upotetun tekstin tulostaminen sen sijaan onnistui huonosti, sillä tekstistä olisi pitänyt tehdä suurempi paremman tulostusjäljen saavuttamiseksi. Tulosteen punainen väri suunniteltiin peitettäväksi mustaa spray-maalia ja lakkaa käyttäen, ja tätä ennen kotelon pinta piti hioa tasaisemmaksi. Pintakäsittelyä ei kuitenkaan päädytty tekemään, sillä tämän opinnäytetyön valmistumisen aikoihin takilasta suunniteltiin tässä raportissa esiin tulleiden puutteiden ja pohdintojen pohjalta kokonaan uusi versio, jota lähdettiin työstämään tässä kuvatun prototyypin täydellisen loppuunsaattamisen sijaan.

6 KOMPONENTTIEN TESTAUS JA OHJELMOINTI

Suuri osa tämän opinnäytetyön toteuttamisesta kului takilaa varten hankittujen komponenttien toimintaa testaten. Kaikki luvussa 5 kuvatut sähkökomponentit piti testata niiden toiminnan varmistamiseksi, ja samalla selvisi oikea tapa ohjata niitä koodin ja sopivan kytkentätavan avulla. Testiohjelmia koodatessa myös Arduinon kehitysympäristö tuli tutuksi yhdessä koodikielen kanssa.

Testeistä oli paljon hyötyä ohjelmoinnin oppimisen kannalta, sekä komponenttivalintojen tekemisessä. Esimerkiksi luvussa 5.2.1 kuvatut Hall-anturit todettiin huonoksi vaihtoehdoksi vasta, kun niitä oli testattu yhdessä Arduinon ja koodin kanssa. Osa testeistä sujui mutkattomasti, mutta jotkin komponentit aiheuttivat päänvaivaa, koska niistä ei ollut saatavilla kovin kattavaa dokumentaatiota ja oikeanlainen kytkentä- ja ohjelmointitapa piti itse päätellä vähin neuvoin. Komponenttitestejä tehtiin sekä opinnäytetyön tekijän tietokoneella että TAMKin automaatiolaboratorion tiloissa.

Testien pohjalta syntyi monia kooditiedostoja, jotka kaikki liittyivät jonkin yksittäisen komponentin tai muutaman komponentin kokonaisuuden toiminnan toteuttamiseen. Lopullinen käyttöliittymä syntyi testausvaiheen lopuksi, kun nämä osaohjelmat yhdistettiin suuremmaksi koko käyttöliittymän muodostavaksi kokonaisuudeksi. Aivan yksinkertaista tämä ei kuitenkaan ollut, vaan käyttöliittymän ohjelmointi vaati myös paljon uutta koodia ja yksittäisten osien hienosäätöä, jotta ohjelmasta tuli toimiva kokonaisuus. Valmista koodia ei ole sisällytetty tähän opinnäytetyöhön salassapidon vuoksi.

6.1 Alustavat komponenttitestit

Ennen varsinaisen käyttöliittymän ohjelmointia tehtiin monia yksittäisiä komponenttitestejä Arduino-aloituspakkauksen mukana tullutta Arduino Unoa, kytkentälevyä ja muita komponentteja käyttäen. Kaikki sähkökomponentit käytiin läpi yksitellen, osa jo paljon ennen Arduino MEGA -mikro-ohjaimen hankkimista. MEGA:a kuitenkin käytettiin hyödyksi etenkin shieldiä ja kosketusnäyttöä testattaessa. Esimerkiksi moottorinohjaukseen liittyvää koodia sen sijaan testattiin jo todella aikaisessa vaiheessa aloituspakkauksen mukana tullutta pientä 9 voltin sähkömoottoria käyttäen, ja sama koodi toimi hyvin myös myöhemmin hankitun tuulilasinpyyhkijän moottorin kanssa.

6.1.1 Arduino-mikrokontrolleri ja perusharjoitukset

Komponentti- ja ohjelmointitestit aloitettiin tekemällä Arduino-aloituspakkauksen mukana tulleessa ohjekirjassa esitettyjä esimerkkitehtäviä. Ne esittelivät Arduinon toimintaa erittäin selkeällä ja helposti ymmärrettävällä tavalla, ja opettivat mikro-ohjaimen perustoiminnallisuuksiin, kuten I/O-porttien tilojen lukemiseen, analogiakytkentöjen tekemiseen, PWM-pulssituksen käyttämiseen, yksinkertaisten painike- ja LED-kytkentöjen tekemiseen, moottorien ohjaamiseen ja LCD-näytön käyttämiseen liittyviä asioita. Moni asia oli tuttu jo TAMKin opintojaksoilta, mutta harjoituksia tehdessä kytkennät ja koodi tulivat käytännössä tutuiksi oikeassa asiayhteydessä.

Eräs testattava asia, mikä ei liittynyt Arduinon ulkoisiin komponentteihin vaan itse mikro-ohjaimeen, oli Arduinon EEPROM-muistin käyttäminen. Normaalisti Arduinon sammuessa sen työmuistissa olevien muuttujien sisältämät arvot katoavat virran katkeamisesta johtuen. Jotkin arvot, kuten vaikkapa valittu kieliasetus tai syvyyden mittayksikkö, on kuitenkin tarve pitää kontrollerin muistissa myös sähkökatkon ajan. Tätä varten useimmissa Arduino-piirilevyissä on niin sanottua EEPROM-muistia, johon voidaan tallentaa ohjelman suorituksen aikana arvoja, jotka jäävät muistiin myös sähkökatkon ajaksi. EEPROM-muistin käyttämiseen tulee hyviä esimerkkiohjelmia Arduino IDE:n mukana, ja niitä tutkimalla selvisi, että muuttujien tallentamiseksi tuli koodiin sisällyttää EEPROM-kirjasto, sekä käyttää read()- ja write()-komentoja aina, kun muistiin tallennettua muuttujaa haluttiin lukea tai kirjoittaa uudelleen. Lisäksi tuli käyttää osoitemuuttujaa, jolla viitattiin oikeaan muistipaikkaan:

```
#include <EEPROM.h>

int address = 0;

void setup()
{
    EEPROM.write(address, 123);    // 123 tallentui muistiin.
    EEPROM.read(address);          // Tämä palauttaa arvon 123.
}
```

Nämä koodirivit tulivat tarpeeseen ohjelmoitaessa takilan asetusvalikkoa, sillä käyttäjän tekemien valintojen haluttiin pysyvän muistissa myös takilan ollessa sammuksissa. Myös kelaussyvyys tallennettiin muistiin, vaikka hyvin suurella todennäköisyydellä pintatason kalibroinnin joutuikin tekemään uudelleen aina uudelleenkäynnistyksen jälkeen.

6.1.2 Moottorinohjaus ja H-silta

Suuri osa komponenttitesteistä ja ohjelmasta koostui moottorin ohjaamiseen liittyvistä asioita ja H-sillan käyttämisestä. Nämä testit aloitettiin selvittämällä H-sillan oikea kytkentätapa. Selvisi, että komponentti tuli kytkeä sen riviliittimistä kahdella johtimella moottorin napoihin ja toisella johtoparilla jännitelähteeseen eli 12 voltin akkuun. Tätä kautta saatiin moottorille sen tarvitsema ohjausjännitettä suurempi käyttöjännite. Ohjausta varten tuli H-sillan GND-pinni kytkeä Arduinon nollapinniin, sekä 5 voltin ohjausjännitettä vastaanottavat EN1- ja EN2-pinnit joihinkin Arduinon I/O-pinneistä. Näihin pinneihin tulevan signaalin perusteella H-silta pystyi ohjaamaan moottorille kulkevaa virtaa siten, että toiseen pinniin tuleva signaali pyöritti moottoria myötäpäivään ja toiseen tuleva signaali vastapäivään. Jos signaali tuli yhtä aikaa molempiin pinneihin, moottori pysähtyi. EN-pinnejä pystyi myös PWM-pulssittamaan, jolloin moottorin pyörimisnopeus muuttui pulssisuhteen mukaan.

Koodin puolella moottorin pyörimissuunnan ja -nopeuden muuttaminen tapahtui analogWrite()-komennon avulla. Tällä komennolla pystyi antamaan ohjausjännitettä mistä tahansa Arduinon PWM-pinnistä 0 ja 5 voltin väliltä. Komennolle annettiin parametreiksi ulostulopinnin numero, sekä analogia-arvo väliltä 0 - 255. Käytännössä moottorin ohjaaminen tapahtui kutakuinkin seuraavalla tavalla if-rakennetta käyttäen:

```
if(currentSpeed > 0){
  analogWrite(motorPinA, currentSpeed * 255);
  analogWrite(motorPinB, 0);
}
else if (currentSpeed < 0){
  analogWrite(motorPinA, 0);
  analogWrite(motorPinB, -currentSpeed * 255);
}
else {
  analogWrite(motorPinA, 0);
  analogWrite(motorPinB, 0);
}
```

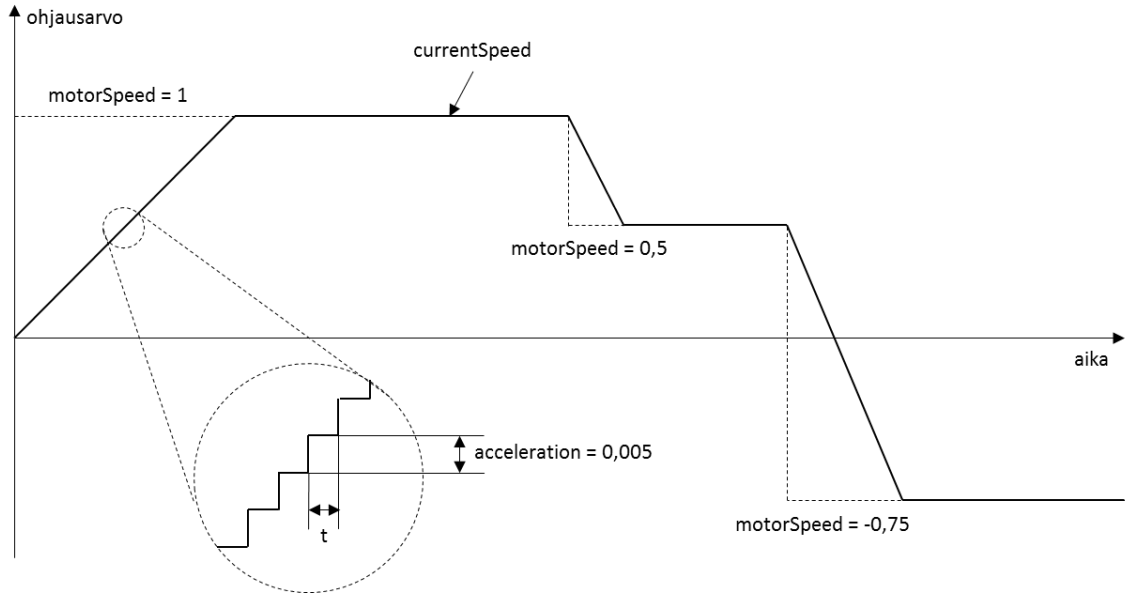
Tätä koodinpätkää kutsuttiin loop()-funktiosta kerran jokaisella ohjelmakierrolla, jolloin koodi kirjoitti H-sillan pinneille (motorPinA ja motorPinB) arvot siten, että moottori pyöri currentSpeed-muuttujan ilmaisemaan suuntaan. Muuttujan arvo oli väliltä -1 - 1, jolloin arvo 1 vastasi moottorin pyörimisnopeutta 100 % nopeudella toiseen pyörimissuuntaan ja -1 100 % nopeudella päinvastaiseen suuntaan. Arvon pysyminen tällä välillä

varmistettiin erillisellä koodirivillä. Koodissa muuttujan arvo myös skaalattiin analogWrite()-komennon vaatimalle 0 - 255 arvovälille. Moottorinohjaukseen liittyvät muuttujat on lueteltu muuttujan tyyppeineen ja arvoväleineen seuraavassa taulukossa.

Taulukko 3. Moottorinohjaukseen liittyvät muuttujat ja niiden kuvaukset.

Muuttuja	Tyyppi	Arvoväli	Kuvaus
motorPinA	int	vakio	H-sillan EN1 kytkentäpinni
motorPinB	int	vakio	H-sillan EN2 kytkentäpinni
motorSpeed	float	-1 - 1	tavoiteltava pyörimisnopeus
currentSpeed	float	-1 - 1	hetkellinen pyörimisnopeus
acceleration	float	0 - 1	kiihtyvyys (nopeuden muutos yhden ohjelmakierron aikana)

Moottorin nopeuden kiihdyttäminen tai hidastaminen haluttuun nopeuteen tehtiin joka ohjelmakierrolla kutsuttavassa motorControl()-nimisessä funktiossa muuttamalla currentSpeed-muuttujan arvoa acceleration-muuttujan verran kohti tavoiteltavaa nopeutta, joka tallennettiin motorSpeed-nimiseen muuttujaan. Käytännössä acceleration-muuttujan arvot olivat pieniä, noin 0,005 kokoluokkaa, sillä yksi ohjelmakierto oli ajallisesti niin lyhyt, että sen aikana nopeuden piti muuttua vain vähän, jotta moottorin kiihdytys- ja hidastuvuusrampeista tuli havaittavan pituiset. Muuttamalla motorSpeed-muuttujan arvoa muualla koodissa esimerkiksi käyttäjän syötteen perusteella koodi automaattisesti pyrki muuttamaan moottorin todellisen pyörimisnopeuden kohti tätä tavoitearvoa kiihdytysramppia seuraten. Tarvittaessa myös kiihdytysnopeuden arvoa pystyi muuttamaan, mikäli moottorin haluttiin reagoivan hitaammin tai nopeammin eri tilanteissa. Tätä ohjaimen toimintaa on havainnollistettu kuvassa 28.



Kuva 28. Periaatekuva moottorinohjaimen toiminnasta. Moottorin todellinen pyörimisnopeuden ohjausarvo currentSpeed pyrkii kaiken aikaa muuttumaan kohti tavoiteltavaa arvoa motorSpeed. Kiihtyvyyden arvo acceleration muuttaa nopeutta ajan t välein, eli kerran ohjelmakierron aikana.

Moottorin ohjaamista varten kirjoitettiin myös muita hyödyllisiä funktioita. Näistä startMotor() ja stopMotor() tekivät juuri sen, mitä funktioiden nimistä voi päätellä, eli ne käynnistivät tai sammuttivat moottorin funktiolle annettua kiihtyvyysarvoa käyttäen. Lisäksi startMotor()-funktiolle annettiin tavoiteltava nopeusarvo, johon se lähti moottoria kiihdyttämään. Lisäksi käytettiin motorSlowdown()-nimistä funktiota, jonka tehtävänä oli hidastaa moottorin kelausnopeutta silloin, kun takila oli kelaamassa kuulaa haluttuun syvyyteen, ja kun kuula oli tietyn lähestymisetäisyyden päässä tavoiteltavasta syvyydestä. Kyseessä oli siis eräänlaisen loppuhidastuksen toteuttava funktio, jolla pyrittiin parantamaan syvyyteenkelaustoiminnon tarkkuutta. Haluttuun syvyyteen kelaus toteutettiin useampaa tätä varten kirjoitettua, hieman monimutkaisempaa logiikkaa sisältävää, funktiota käyttäen.

6.1.3 Kauko-ohjaus

Kauko-ohjaimen ohjelmoiminen aiheutti muihin komponentteihin verrattuna ehkäpä eniten päänvaivaa, sillä sen toimintaan saattamisessa ilmeni ongelmia useassa kohtaa työn suorittamista. Komponentin toimintaa lähdettiin testaamaan netistä sille löydettyä valmistajan esimerkkikoodia käyttäen ja muokaten. Esimerkkikoodi saatiin toimimaan pienten

ponnisteluiden jälkeen, ja siitä saatuja oppeja käytettiin apuna oman koodin kirjoittamisessa. Ongelmia tuotti vastaanottimen liittäminen kauko-ohjaimen pariksi vastaanotinpiirin reset-painiketta painamalla, mitä ei aluksi ymmärretty tehdä. Tästä syystä vastaanotin ei tunnistanut käytetyn ohjaimen signaalia ollenkaan, vaikka koodi toimikin oikein. Kun liittäminen oli tehty, saatiin myös koodi toimimaan ja kauko-ohjaimen painikkeiden painallukset näkymään Arduino IDE:n serial-näytöllä.

Koodissa tuli radiovastaanottimen 6 pinniä (joista 1 ei ollut välttämätön) asettaa joko input- tai output-pinneiksi. Yksi pinni kirjoitettiin output-pinniksi, ja se asetettiin antamaan jatkuvaa 5 voltin käyttöjännitettä vastaanottimelle. 4 pinniä asetettiin input-pinneiksi, ja niiden kautta saatiin tieto kauko-ohjaimen painikkeiden painalluksista. Kutakin painiketta vastasi siis yksi pinni. Painikkeiden lukeminen tehtiin radioControl()-nimisessä joka ohjelmakierrolla kutsuttavassa funktiossa digitalRead()-komentoa käyttäen, jolla luettiin for-silmukkarakenteessa jokaisen painikkeen tila yksitellen muistiin radioStates[]-nimiseen neljäpaikkaiseen taulukkomuuttujaan:

```
void radioControl()
{
  for(int i = 0; i < 4; i++){
    radioStates[i] = digitalRead(radioPins[i]);

    if(radioStates[i] == HIGH && radioStatesPrev[i] == LOW){
      radioActions(i, true);
    } else if(radioStates[i]==LOW && radioStatesPrev[i]==HIGH){
      radioActions(i, false);
    }

    radioStatesPrev[i] = radioStates[i];
  }
}
```

Vastaanotinpiirin kauko-ohjaimen painikkeita vastaavien digitaalipinnien numerot luettiin digitalRead()-komentoa varten radioPins[]-taulukkomuuttujasta, johon ne ohjelman alustuksessa oli talletettu. Funktiossa painikkeiden tilojen luennan jälkeen tiloja verrattiin edellisellä ohjelmakierrolla luettuihin tiloihin, ja jos signaalissa havaittiin nouseva tai laskeva reuna, siirryttiin radioActions()-nimiseen funktioon, joka vastasi kauko-ohjaimen painikkeiden painallusten linkittämisestä eri toimintoihin ja jolle tätä varten syötettiin havaitun napinpainalluksen järjestysnumero sekä boolean-tyyppinen arvo kertomaan laskevasta tai nousevasta reunasta. Aivan yllä kuvatun funktion lopuksi painikkeiden tilat tallennettiin vielä toiseen radioStatesPrev[]-nimiseen taulukkomuuttujaan, jotta niitä voitiin

tämän muuttujan kautta verrata seuraavalla ohjelmakierrolla luettuihin uusiin tila-arvoihin.

6.1.4 Näytöt

Ennen kosketusnäytön hankkimista käytössä oli Arduino-aloituspakkauksen mukana tullut kaksirivinen LCD-näyttö, jota testattiin paljon ja jolle koodattiin kokonainen alustava käyttöliittymä valikkorakenteineen ja toimintoineen (kuten liitteessä 4). Kuten luvussa 5.4.1 kerrottiin, tarvittiin näytön kytkemiseen 12 I/O-pinniä. Näytön käyttämisessä ei ilmennyt varsinaisia ongelmia, mutta koska kytkentä oli melko monimutkainen, piti oikeanlaisen kytkennän tekemisessä olla tarkkana. Näyttö ei toiminut, mikäli kytkentä oli tehty millään lailla väärin. Varsinkin ennen luvussa 5.5 kuvatun apupiirilevyn käyttöönottoa kytkentävirheen tekeminen oli melko yleistä.

LCD-näytön ohjelmoimiseen liittyi useita käytettäviä komentoja, joilla näyttö ensinnäkin alustettiin ja otettiin käyttöön ohjelman käynnistyessä ja joilla sille saatiin kirjoitettua ASCII-merkistön mukaisia symboleja. Näytön alustusta varten tarvittiin seuraavat koodirivit:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(33, 31, 29, 27, 25, 23);

void setup()
{
  lcd.begin(16, 2);
}
```

Arduinon sisäinen LiquidCrystal.h-koodikirjasto tuli sisällyttää ohjelmaan, ja sitä kautta koodissa pystyi käyttämään näytön ohjelmoimiseen tarkoitettuja komentoja. Näyttömuuttuja luotiin lcd()-komentoa käyttäen, ja samalla ohjelmalle kerrottiin näytön kytkentäpinnit. Tämän jälkeen setup()-funktiossa näyttö otettiin käyttöön tietyn kokoisena (16 merkkiä leveä, 2 riviä).

Kaikki näytölle tulostettavat merkit käsiteltiin screenControl()-nimisessä funktiossa, joka aina kutsuttaessa päivitti näytölle kutakin tilannetta vastaavan valikkonäkymän. Toisin kuin vaikkapa moottorinohjaukseen liittyvää funktiota, tätä funktiota ei kutsuttu joka oh-

jelmakierrolla, sillä tällöin näyttö vilkkui jatkuvasti sen hitaasta päivitystaajuudesta johtuen. Funktio kutsuttiinkin muualta kuin loop()-funktiosta aina, kun tehtiin jokin muutos, joka muutti myös näytön näkymää.

Merkkien tulostaminen näytölle tapahtui todella yksinkertaisesti. Aluksi näytön ”kursori” tuli asettaa haluttuun kohtaan näyttöä lcd.setCursor()-komentoa käyttäen, minkä jälkeen käytettiin lcd.print()-komentoa, joka tulosti kursorin kohdalle ja siitä oikealle komennolle syötetyn tekstin tai muuttujan sisältämän datan. Näytön sai myös tyhjennettyä kaikista merkeistä käyttämällä lcd.clear()-komentoa, kuten tässä typistetyssä screenControl()-funktiossa on esitetty:

```
void screenControl()
{
  lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print("merkkeja");
  writeString(languages[selectedLanguage][selectedMenu]);
}
```

Hankaluuksia tuotti ä-, ö- ja å-kirjainten sekä joidenkin erikoismerkkien, kuten nuolten, tulostaminen näytölle, sillä ne eivät kuulu alkuperäiseen 7-bittiseen ASCII-kirjastoon (ASCII Codes Table 2016). Selvisi kuitenkin, että LCD-näytön koodikirjasto tukee joitakin ASCII-merkistön ulkopuolisia symboleja, joihin takilan käyttöliittymässä tarvittavista lukeutuivat ainoastaan ä- ja ö-kirjaimet. Ne pystyi tulostamaan näytölle käyttämällä komentoja lcd.print(char(225)) ja lcd.print(char(239)). Å-kirjain sekä tarvittavat 4 nuolta piti määritellä ohjelmalle hankalammalla tavalla, mutta nekin saatiin lopulta näkyviin. Kaikkien näiden merkkien tulostaminen näytölle tehtiin itse kirjoitettua writeString()-funktiota käyttäen, jolle pystyi syöttämään näitä merkkejä sisältävän tekstimuuttujan ja joka osasi tarvittavia komentoja käyttäen myös tulostaa ne näytölle. Valikon eri kielivalinnat mahdollistettiin käyttäen kaksiulotteista languages[][]-taulukkomuuttujaa, johon oli tallennettu kaikki valikoissa käytetyt sanat eri kielillä. Oikea sana oikealla kielellä valittiin kieliasetuksen ja valitun valikkosivun perusteella.

Kosketusnäytön testaamisessa käytettiin apuna Adafruit.com -sivustolla kyseiselle näytölle annettuja esimerkkikoodeja. Sieltä ladattiin myös näytön ohjelmoimiseen tarvittavat koodikirjastot, jotka sisälsivät muun muassa kosketusominaisuuteen ja graafisten muoto-

jen piirtämiseen liittyvää valmista koodia. Kuten arvata saattoi, liittyi tämän kosketusnäytön ohjelmoimiseen paljon enemmän asiaa kuin LCD-näytön ohjelmoimiseen. Näyttö tuli ensinnäkin alustaa seuraavilla koodiriveillä:

```
#include <Adafruit_GFX.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_ILI9341.h>
#include <Adafruit_STMPE610.h>

Adafruit_STMPE610 ts = Adafruit_STMPE610(8);
Adafruit_ILI9341 tft = Adafruit_ILI9341(10, 9);

void setup(void)
{
  tft.begin();
  if (!ts.begin()) Serial.println("Unable to start
touchscreen.");
  else Serial.println("Touchscreen started.");

  tft.setRotation(1);
}
```

Näyttö vaati siis toimiakseen viiden ulkoisen koodikirjaston sisällyttämisen koodiin. Lisäksi näyttöä varten esiteltiin kaksi muuttujaa, joista noista käytettiin kontrolloimaan näytölle piirrettävää geometriaa ja toista painallusten havaitsemiseen. Näyttö ja sen kosketusaisti olivat siis ikään kuin kaksi erillistä objektia. Esittelyn jälkeen kumpikin näistä objekteista tuli ”käynnistää” setup()-funktiossa begin()-komentoa käyttäen. Näytön kiertokulmaa pystyi myös säätämään 90° välein setRotation()-komennon avulla.

Näytön painallusten lukeminen ja käsittely tehtiin joka ohjelmakierrolla loop()-funktiosta kutsutussa touchscreenControl()-funktiossa seuraavasti:

```
void touchscreenControl()
{
  // This empties the touch buffer.
  while (ts.bufferSize() > 0) ts.getPoint();

  if (!ts.bufferEmpty() && ts.touched())
  {
    TS_Point p = ts.getPoint();
    p.x = map(p.x, TS_MINY, TS_MAXY, 0, tft.height());
    p.y = map(p.y, TS_MINX, TS_MAXX, 0, tft.width());
    y = tft.height() - p.x;
    x = p.y;

    buttonActions();
  }
}
```

Näyttö toimi siten, että aina kun se havaitsi painalluksen, painallus kirjoitettiin koordinaatteineen puskurimuistiin. Kun ohjelmakierron aikana muistissa havaittiin painallus, sen koordinaatit haettiin talteen p-nimiseen pistetyyppiseen muuttujaan. Tämän jälkeen pisteen koordinaatit skaalattiin map()-funktiolla näytön käännettyyn koordinaatistoon, sillä kosketusobjektin koordinaatisto ei sitä alun perin vastannut. Lopuksi siirryttiin buttonActions()-nimiseen funktioon, jossa koordinaattien ja valittuna olleen valikon perusteella toteutettiin painalluksen sijaintiin liittyvät toiminnot. Eli jos koordinaatit osuivat vaikkapa painonapin kohdalle, toteutettiin napin toiminto.

Painonappien ohjelmoiminen oli monimutkaisempi tehtävä, sillä niitä oli niin paljon useissa eri koordinaateissa erikokoisina, -värisinä ja eri teksteillä ja toiminnoilla. Kaikki painikkeiden piirtämiseen tarvittavat tiedot päädyttiin tallentamaan taulukkomuuttujiin, joita määriteltiin ohjelman alussa kaksi kutakin painiketta (tai muuta näytön elementtiä) kohti. Toinen sisälsi tiedot painikkeen grafiikasta ja toinen sen tekstistä eri kielillä. Painikkeiden piirtämistä varten kirjoitettiin funktio, joka piirsi painikkeen sille syötettyjen taulukkomuuttujien sisältämien tietojen perusteella. Näin jokaisen painikkeen piirtämistä ei tarvinnut erikseen koodata ohjelmaan. Funktiossa painikkeiden piirtämiseen käytettiin kosketusnäytön koodikirjastoiden mukana tulleita valmiita komentoja. Esimerkiksi pyöristetyn suorakulmion muotoisen painikkeen piirtäminen tehtiin seuraavalla tavalla:

```
void drawBtn(int a[], char* aTxt[])
{
  if (a[4] == 0)
  {
    // Draw rounded rectangle.
    tft.fillRoundRect(a[0], a[1], a[2], a[3], rnd, a[5]);
    tft.setCursor(a[6], a[1] + a[3] * 0.5 - 6);
    tft.setTextColor(a[7]);
    tft.setTextSize(2);
    tft.println(aTxt[language]);
  }
}
```

Kirjastojen mukana tuli fillRoundRect()-komento sekä muita vastaavia komentoja, jotka piirsivät näytölle valmiin geometrisen muodon sille annettujen koordinaattien ja värin perusteella. Yllä funktio lukee koordinaatit ja muut tiedot sille syötetystä a[]-taulukkomuuttujasta. Tekstin tulostamisessa oli käytössä myös LCD-näytössä käytetty kursoriominaisuus, eli kursori tuli ensin siirtää haluttuun kohtaan, minkä jälkeen teksti piirrettiin sen kohdalle halutulla värillä ja kirjasinkoolla.

Näytöllä eri valikkoihin siirryttäessä näytölle piirrettiin kaikki valikkoon kuuluvat painikkeet tätä drawBtn()-funktiota useita kertoja kutsumalla syöttäen sille aina eri painikkeen taulukkomuuttujat. Joissakin tilanteissa painike piti hävittää näytöltä, ja tätä varten kirjoitettiin eraseBtn()-niminen funktio, joka ainoastaan piirsi uuden painikkeen vanhan päälle käyttäen valikoiden taustaväriä, hävittäen näin painikkeen näkyvistä. Koko näytön värjääminen jollakin värillä tapahtui tft.fillScreen()-komennon avulla, jolle syötettiin ainoastaan käytettävä väri.

6.1.5 Pulssianturi ja muu bittitieto

Kaikuluotaimen NMEA-väylää lukuun ottamatta kaikki anturitieto saatiin digitaalisessa muodossa, joten pulssianturin induktiivisten antureiden, mikrokytkimen sekä LCD-näytön kanssa käytettyjen painikkeiden lukeminen tehtiin koodissa lähes samalla tavalla kaikissa näissä tapauksissa. Keskeisin koodin komento oli digitalRead(), jolle annettiin syötteeksi ainoastaan luettavan sisääntuloportin järjestysnumero. Komento sen jälkeen palautti joko arvon tosi tai epätosi riippuen siitä, tuliko porttiin 5 voltin jännite vai ei.

Tämän bittitiedon käsittelyssä oli joitakin eroja tilanteesta riippuen. Käyttöliittymän ensimmäisissä versioissa ja painiketesteissä yksinkertaisin koodi oli mikrokytkimellä:

```
if(digitalRead(inputPins[6]) == HIGH && motorSpeed > 0){
  stopMotor(0.05);
}
```

Yllä inputPins[6] vastaa mikrokytkimen kytkentäpinniä. Tämä if-rakenne suoritettiin joka ohjelmakierrolla, ja se sai moottorin pysähtymään, mikäli moottori oli kelaamassa kuulaa ylöspäin mikrokytkimen kytkeytyessä päälle.

LCD-näytön kanssa käytettyjen painikkeiden koodi oli hieman edellistä monimutkaisempi, sillä painalluksista haluttiin tunnistaa sekä nousevat että laskevat reunat. Niiden tunnistus tehtiin seuraavalla tavalla:

```
for(int i = 0; i < 4; i++){
  buttonStates[i] = digitalRead(inputPins[i]);

  if(buttonStates[i] == HIGH && buttonStatesPrev[i] == LOW){
    buttonActions(i, true);
    Serial.println("sdf");
    } else if(buttonStates[i] == LOW &&
              buttonStatesPrev[i] == HIGH){
    buttonActions(i, false);
  }

  buttonStatesPrev[i] = buttonStates[i];
}
```

Painikkeiden pinnien numerot tallennettiin ohjelman alustuksessa inputPins[]-taulukkomuuttujaan. Lisäksi painikkeiden tilat luettiin talteen buttonStates[]- ja buttonStatesPrev[]-nimisiin taulukkomuuttujiin yksitellen joka ohjelmakierrolla for-silmukan avulla. Vertaamalla ohjelmakierrolla luettua painikkeen tilaa edellisellä kierroksella luettuun tilaan voitiin if-else-rakenteen avulla havaita, oliko painike painettu pohjaan vai oliko siitä päästetty irti. Tämän perusteella voitiin buttonActions()-funktiolle kertoa, minkä painikkeen painallus oli kyseessä ja oliko reuna nouseva vai laskeva.

Induktiivisten antureiden lukeminen ja kelan pyörimissuunnan selvittäminen vaati edelliseen verrattuna vielä hieman monimutkaisempaa koodia, sillä yhden signaalin nousevan ja laskevan reunan tunnistamisen sijaan piti tunnistaa, kumpi kahdesta signaalista oli ensin päällä. Koodi kuitenkin muistuttaa paljon nousevan ja laskevan reunan tunnistusta:

```
void sensorControl()
{
  sensorStates[0] = digitalRead(sensorPins[0]);
  sensorStates[1] = digitalRead(sensorPins[1]);

  // When both inductive sensors are on...
  if(sensorStates[0] == HIGH && sensorStates[1] == HIGH){
    // ...go to action depending on which one was on first.
    if(sensorStatesPrev[0] == HIGH && sensorStatesPrev[1] ==
LOW){
      sensorActions(0);
    } else if(sensorStatesPrev[0] == LOW &&
              sensorStatesPrev[1] == HIGH){
      sensorActions(1);
    }
  }

  sensorStatesPrev[0] = sensorStates[0];
  sensorStatesPrev[1] = sensorStates[1];
}
```

Tässä koodissa verrataan jälleen joka ohjelmakierrolla induktiivisilta antureilta luettuja arvoja edellisellä ohjelmakierrolla luettuihin arvoihin. If-rakennetta käyttäen kutsuttiin

sensorActions()-funktiota aina silloin kun molemmat anturit antoivat signaalin, ja kun toinen niistä oli edellisellä kierrolla epätosi. Pyörimissuunta saatiin selville siitä, kumman anturin arvo oli epätosi, ja tämän perusteella sensorActions()-funktiossa tiedettiin joko pienentää tai kasvattaa syvyyslukemaa.

6.1.6 Kaikuluotaimen syvyystiedon parsiminen

Kaikuluotaimelta saatavan syvyystiedon vastaanottamista päästiin testaamaan vain hetkellisesti, sillä tarkoitukseen sopiva kaikuluotain pystyttiin lainaamaan testejä varten vain vähäksi aikaa. Kaikuluotaimen palauttamisen jälkeen uutta ei löydetty tilalle, eikä sellaista viitsitty ostaa kalliiseen hintaan pelkkiä testejä varten. Testeissä kuitenkin onnistuttiin vastaanottamaan NMEA 0183 -standardin dataa Arduinolla, joka näytti datan Arduino IDE:n serial monitorissa.

Kaikuluotaimen NMEA-piuhassa oli yhteensä 4 johdinta, joiden oikean kytkentätavan selvittäminen tuotti aluksi paljon ongelmia. Lopulta tiedonhaun jälkeen selvisi, että tavoiteltava kytkentä oli yksinkertainen: yksi johtimista kytkettiin Arduinon gnd-pinniin ja yksi sen rx-porttiin, eli pinniin 0, joka yhdessä tx-portin 1 kanssa on tarkoitettu Arduinon ja muiden laitteiden väliseen kommunikointiin. Kun kaikuluotaimen serial-asetuksista baudinopeus ja muut asetukset asetettiin oikein, lähetti kaiku dataa Arduinolle luettavassa muodossa.

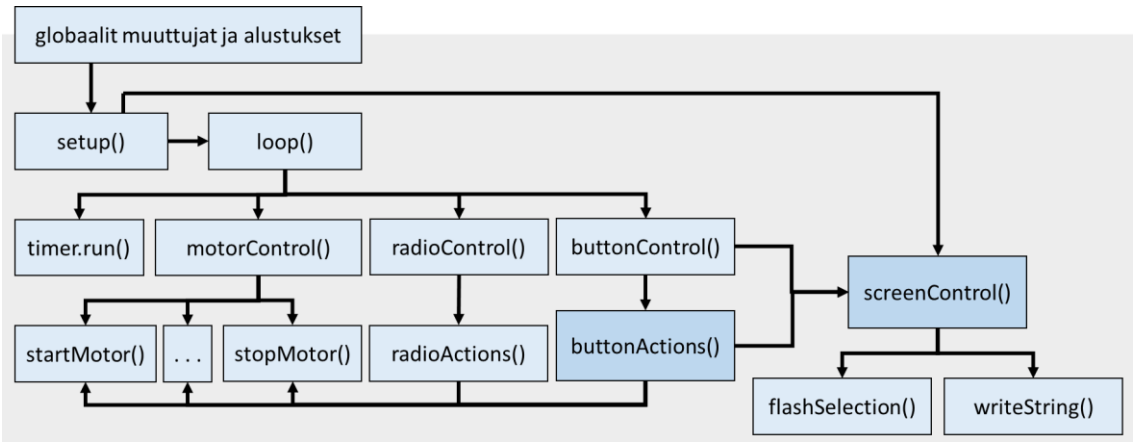
Kaikuluotain lähetti siihen käynnistetyn demo-ohjelman aikana dataa serial-ikkunaan tekstirivin kerrallaan tasaisin väliajoin. Rivi sisälsi monia numero- ja kirjainarvoja pilkuin eroteltuna, ja niiden joukosta tuli etsiä järven syvyyttä vastaava lukuarvo. Se tunnistettiin melko helposti liikuttelemalla kaikuluotaimen lähetintä pystysuunnassa lattiaan osoitettuna seuraten samalla lukujen muutoksia. Kun syvyyslukeman paikka tekstijonosta oli tunnistettu, jäljelle jäi sen parsiminen erilleen muusta datasta ja arvon tallentaminen muuttujaan. Parsimisen tekevää koodia ei kuitenkaan päästy kirjoittamaan kokonaan loppuun kaikuluotaimen puuttuessa.

6.2 Käyttöliittymän ohjelmointi

Sähkötakilan käyttöliittymien koostaminen aloitettiin, kun kaikki niitä varten tarvittavat komponenttitestit ja komponentteihin liittyvät ohjelman osat oli saatu valmiiksi. Kuten jo aiemmin on todettu, ohjelmoitiin takilaa varten kaksi käyttöliittymää, joista ensimmäinen tehtiin LCD-näytölle ja painikkeille sekä toinen kosketusnäytölle. Käyttöliittymien ohjelmoimisessa käytettiin apuna luvussa 5.4.3 kuvattuja Excel-tiedostoja. Kaikki visuaalinen suunnittelu tehtiin niiden avulla, eikä värejä, eri elementtejä tai niiden sijoittelua tarvinnut juurikaan miettiä enää ohjelmointivaiheessa, jolloin ajan pystyi käyttämään tehokkaammin hyödykseen pelkkiä koodiin liittyviä asioita pohtien ja selvittäen. Seuraavissa luvuissa kerrotaan, miten kumpikin käyttöliittymä rakennettiin ja miten niissä huomioitiin luvussa 3.5 kuvailtuja hyvään käyttöliittymään liittyviä asioita.

6.2.1 Ensimmäinen käyttöliittymä ja toiminnot

LCD-näytölle tarkoitettu käyttöliittymä koostettiin, kun moottorinohjaukseen, kauko-ohjaimeen, LCD-näyttöön sekä painikkeiden painallusten ja antureiden digitaalisignaalien vastaanottamiseen liittyvät osakoodit oli saatu hahmoteltua. Nämä osakoodit yhdistettiin yhteen ohjelmaan, jonka rakennetta on havainnollistettu kuvassa 29. Tässä vaiheessa otettiin tavaksi kirjoittaa sisääntulosignaalien käsittely ja toimintoja toteuttavat ohjelmarivit omiksi erillisiksi funktioikseen. Esimerkiksi painikkeiden painallusten havaitseminen tehtiin buttonControl()-nimisessä funktiossa, josta kutsuttiin buttonActions()-nimistä funktiota aina, kun painallus tunnistettiin. Tunnistetun painikkeen järjestysnumero välitettiin buttonActions()-funktiolle, joka sen perusteella yhdisti oikean moottorinohjaukseen tai valikossa liikkumiseen liittyvän toiminnon kyseiseen painikkeeseen. Vastaavat funktiot kirjoitettiin myös kauko-ohjaimelle ja antureiden signaalien aiheuttamille toiminnoille, mutta ne vaikuttivat lähinnä moottorinohjaukseen, eivätkä juurikaan näytön ohjaukseen.



Kuva 29. Ensimmäisen käyttöliittymän ohjelmarakenne pääpiirteissään.

Käyttöliittymän valikkorakenne syntyi buttonActions()- ja screenControl()-funktioiden yhteistoiminnan tuloksena. Globaaleja muuttujia käyttäen välitettiin tietoa esimerkiksi valittuna olleesta valikosta eri funktioiden välillä. Aina painiketta painettaessa toteutettiin jokin toiminto, mikä usein tarkoitti valikosta toiseen siirtymistä. Tällöin näytölle tulostetut merkit päivitettiin uusiin screenControl()-funktiota käyttäen. Näihin kahteen funktioon kirjoitettiin kumpaankin pitkät if-else-rakenteet, joissa määriteltiin kuhunkin valikon sivuun liittyvät toiminnot jokaiselle painikkeen painallukselle sekä logiikka oikeiden tekstijonojen ja merkkien tulostamiselle. Näyttöä käyttävää funktiota ei voitu kutsua loop()-funktiosta joka ohjelmakierrolla, koska tällöin näyttö olisi päivittynyt jatkuvasti ja alkanut vilkkumaan sen huonosta päivitystaajuudesta johtuen. Tästä syystä näytön päivittäminen tehtiin kutsumalla screenControl()-funktiota vain silloin, kun jokin toiminto aiheutti näytölle muutoksen.

Ohjelman käynnistyessä suoritettiin setup()-funktio, jonka yhteydessä näytölle tulostettiin takilan nimen ja versionumeron näyttävä käynnistymisruutu. Kun tämän jälkeen siirryttiin muutaman sekunnin viiveellä muun muassa moottorinohjauksen käynnistävään loop()-funktioon, siirryttiin aloitusnäytöstä valikkorakenteen päävalikkoon (liitteessä 4 vasen yläkulma, syvyyden asetus). Tässä funktiossa päivitettiin myös ajastinta, jota käytettiin joissakin valikoissa valittuna olleiden elementtien vilkuttamiseen päälle ja pois flashSelection()-funktiota käyttäen. Funktio writeString() tarvittiin ä-, ö- ja å-kirjainten tulostamista varten.

Syvyysmuuttujan lukuarvon päivittäminen pulssianturin signaalin perusteella tehtiin seuraavalla tavalla:

```
void updateDepth(int sign)
{
  if(sign != 0){
    depth += sign * circumference / pulses;
  }
}
```

Funktiolle syötettiin joko luku 1 tai -1, jonka perusteella syvyysarvoa muutettiin joko pienemmäksi tai suuremmaksi. Syvyysarvon suuruuden muutos laskettiin kelan halkaisijan ja yhtä kierrosta vastaavan pulssimäärän perusteella, jotka kummatkin olivat alustuksessa määriteltyjä vakioita. Tämän funktion lisäksi syvyystietoon liittyi moottorinohjausfunktioissa käytetty maksimisyvyysvakio, jota alemmas moottori ei suostunut kuulaa laskemaan, automaattikelauksen hidastusetäisyys, jonka päässä tavoiteltavasta syvyydestä moottori alkoi kelaamaan narua hitaammalla vauhdilla, sekä yksikönmuunnoksia varten käytetty muunnoskertoimet sisältävä taulukkomuuttuja, josta valitun yksikön perusteella valittiin oikea muunnoskerroin ennen syvyyslukeman tulostamista näytölle.

Käyttöliittymän valikot päätettiin jo aikaisessa vaiheessa toteuttaa usealla eri kielellä; suomeksi, englanniksi ja ruotsiksi. Jokaisella valikon sivulla oli näytön ylemmällä rivillä otsikkoteksti, jonka tuli näkyä oikealla kielellä kieliasetuksesta riippuen. Näiden otsikkorivien tulostaminen tehtiin joustavasti käyttäen ohjelman alussa alustettua kaksiulotteista taulukkomuuttujaa:

```
char** languages[] = {finnish, english, swedish};
```

Languages[][]-taulukkomuuttujaan tallennettiin siis kolme muuta taulukkomuuttujaa, joista jokainen sisälsi valikkojen otsikkorivit aina yhdellä kielellä kirjoitettuna. Nämä alemman tason taulukot määriteltiin myöskin ohjelman alussa, ennen languages[][]-muuttujan määrittelyä. Kun myöhemmin screenControl()-funktiossa haluttiin kirjoittaa valittuna olleen valikon osoiterivi näytölle, käytettiin seuraavaa koodiriviä:

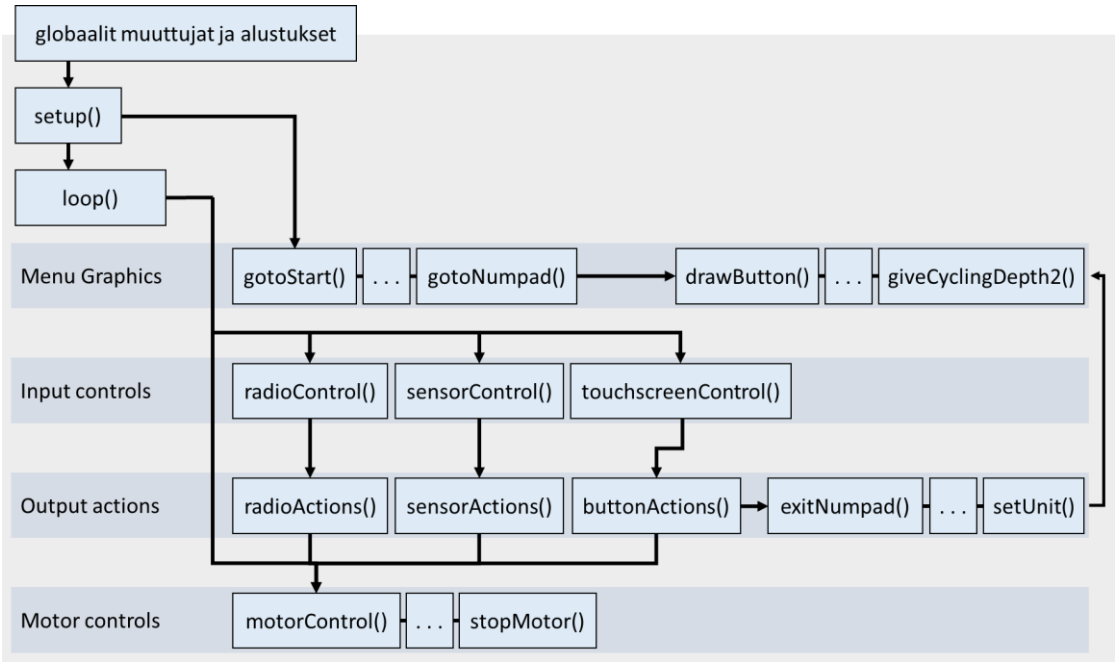
```
writeString(languages[selectedLanguage][selectedMenu]);
```

Tässä siis haetaan languages[][]-muuttujasta teksti, joka vastaa valittuna olevaa valikkoa (selectedMenu) oikealla kielellä kirjoitettuna (selectedLanguage). Teksti sitten annetaan writeString()-muuttujalle, joka kirjoittaa sen näytölle huomioiden ä-, ö- ja å-kirjaimet.

Kieli- ja yksikköasetuksen sisältävät muuttujat tallennettiin Arduinon EEPROM-muistiin, jossa ne säilyivät tallessa myös sähkökatkon yli.

6.2.2 Käyttöliittymän ohjelmointi kosketusnäytölle

Kosketusnäytölle ohjelmoitu lopullinen käyttöliittymä oli suoraa jatketta edelliselle käyttöliittymälle, joten sen ohjelma (kuva 30) muistutti rakenteeltaan paljon edellä kuvassa y kuvattua LCD-näytön käyttöliittymää. Rakenne kasvoi kuitenkin hieman monimutkaisemmaksi osittain kosketusnäytön vaativan monimutkaisemman koodin takia.



Kuva 30. Lopullisen käyttöliittymän ohjelmarakenne pääpiirteissään. Ohjelman funktiot on jaettu neljään kategoriaan niiden toiminnallisuuden perusteella koodin selkeyttämiseksi.

Globaaleja, koko ohjelman kattavia, muuttujia käytettiin jälleen paljon, jotta tieto esimerkiksi valittuna olleesta käyttöliittymän valikosta saatiin useiden funktioiden luettavaksi. Tällä kertaa valikkorakenne ei syntynyt ainoastaan kahden funktion yhteistoiminnan tuloksena kuten aikaisemmin, vaan ohjelman alustuksessa määriteltiin sivujen painikkeiden sijainnit, tekstit, värit ja muut tiedot, joiden perusteella kunkin valikon sivun piirtämiseen tarkoitetut funktiot (esimerkiksi gotoStart()) pystyivät niitä kutsuttaessa piirtämään näytölle juuri halutun valikon. Ohjelma oli näin paljon edellistä joustavampi ja selkeämpi.

Nämä valikkofunktiot tarvitsivat toimiakseen lisäksi useita apufunktioita, joilla yksittäiset painikkeet, niiden tekstit sekä muuttuvia lukuarvoja piirtävät tekstit piirrettiin näytölle.

Sisääntulosignaalien käsittely tehtiin jälleen niille tarkoitetuissa omissa funktioissaan. Nyt aikaisemmin käytetty buttonControl()-funktio korvattiin ja jaettiin kahdeksi funktioksi. Kosketusnäytön painallusten käsittely tehtiin touchscreenControl()-funktiossa, ja antureiden ja mikrokytkimen signaalit tunnistettiin sensorControl()-funktiolla. Moottorinohjauksen funktiot jaettiin omaksi suuremmaksi kokonaisuudekseen, jota useat muut ohjelman funktiot kutsuivat.

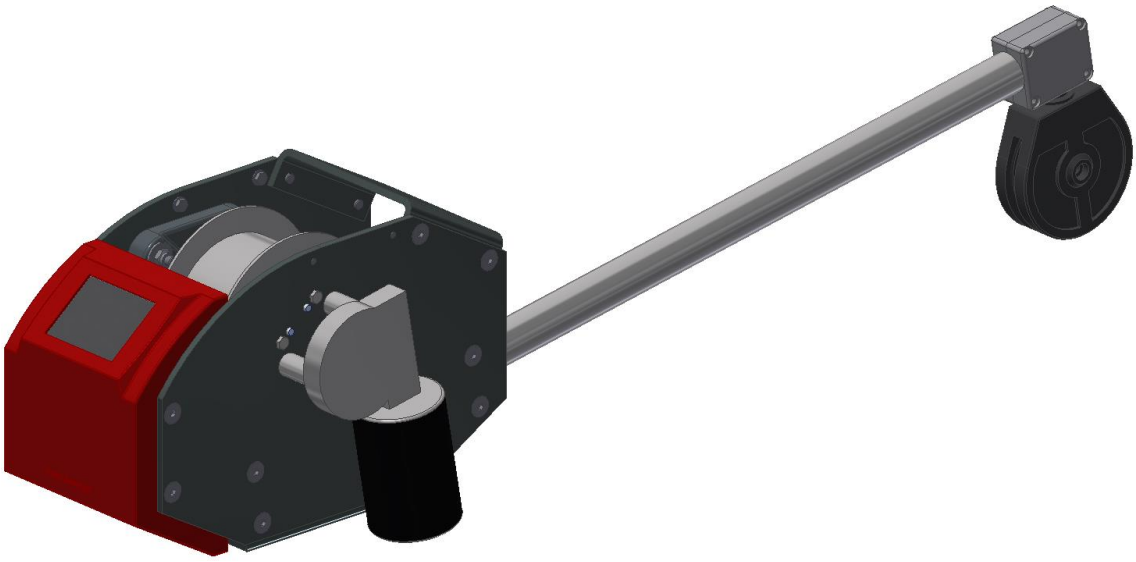
Sisääntulosignaalien aiheuttamien toimintojen toteuttaminen tehtiin output controls -ryhmään jaettujen funktioiden avulla, jotka linkittivät sisääntulosignaalit oikeisiin toimintoihin. Nämä funktiot siis kutsuivat sekä moottorinohjauksen että näytön grafiikoiden piirtämiseen tarkoitettuja funktioita. Muita toiminnallisuuksia, kuten kieli- tai yksikköasetuksen muuttamista, varten kirjoitettiin joitakin apufunktioita.

Eräs käyttöliittymään lisätty uusi toiminnallisuus oli numeronäppäimistösivun lisääminen ohjelmaan. Edellisessä käyttöliittymässä numeroarvoja tai muuta käyttäjän syötettä antaessa tuli siirtyä kutakin arvoa varten ohjelmoituun valikkoon, jolloin esimerkiksi kieliasetus, yksikköasetus ja haluttuun syvyyteen kelauksen lukuarvo annettiin jokainen omassa tätä tarkoitusta varten erikseen ohjelmoidussa valikossa. Nyt kaikkien numeroarvojen syöttäminen tehtiin yhtä numpad-nimistä sivua käyttäen, joka sen ok-painiketta painettaessa palautti siihen syötetyn arvon siihen muuttujaan, jonka arvoa numeronäppäimistöllä oltiin muuttamassa. Yhdellä sivulla siis pystyttiin muokkaamaan useita lukuarvoja joustavasti. Asetussivun kieli- ja yksikköasetukset eivät olleet lukuarvoja, joten niiden muokkaaminen toteutettiin vielä toistaiseksi erillään numeronäppäimistöstä.

Automaattisen syvyysseurannan ohjelmoiminen jouduttiin jättämään pois tästä opinnäytetyöstä, sillä tämän toiminnallisuuden testaamista varten välttämätöntä NMEA 0183 -standardia tukevaa kaikuluotainta ei saatu ajoissa mistään käyttöön. Syvyysseurantaa ja syklistä syvyysseurantaa varten kuitenkin ohjelmoitiin omat valikkonsa, joihin syvyysseurannan toiminnallisuus voidaan myöhemmin yhdistää.

7 TULOSTEN POHDINTA

Tämän opinnäytetyön tuloksena syntyi moottorinohjaus ja käyttöliittymä alla olevan kuvan 3D-mallin mukaiseen sähkötakilaan. Lopullisen ohjausjärjestelmän kytkentäkaavion sivut ovat liitteenä 8. Vaikka kaikkia vaatimusluettelossa listattuja tavoitteita ei erinäisistä syistä saavutettukaan, tuli ohjauksesta pääpiirteissään toimiva ja vaatimusten mukainen. Sen pohjalta takilaa on nyt helpompi lähteä työstämään kaupalliseksi versioksi, jota voidaan valmistaa ja myydä useampia kappaleita voittoa tehden.



Kuva 31. Lopputuloksena syntyneen sähkötakilan 3D-malli.

Suurin syy joidenkin ominaisuuksien ja rakenteiden keskeneräiseksi jäämiselle oli, että ennen prototyypin valmiiksi saattamista päätettiin, että myytäväksi tarkoitettu versio sähkötakilasta tullaan suunnittelemaan lähes täysin uudelleen alusta saakka, jotta siinä voidaan heti alusta alkaen ottaa huomioon mahdollisimman monet tätä prototyyppiä rakentaessa ilmi tulleet ongelmat ja puutteet. Työn lähestyessä loppuaan kaupallisen version suunnittelu asetettiinkin tärkeysjärjestyksessä tämän opinnäytetyön kohteena olleen prototyypin valmiiksi saattamisen edelle, eikä kaikkien kesken jääneiden ominaisuuksien rakentamista haluttu jatkaa, koska kyseessä olisi ollut vain turhaa työtä. Näiden keskeneräisten ominaisuuksien hyödyt ja haitat oli tiedostettu, vaikka ominaisuuksia ei valmiiksi saatukaan. Myös kaikista valmiiksi saaduista ominaisuuksista otettiin opiksi analysoimalla niiden hyviä ja huonoja puolia kaupallisen version näkökulmasta.

7.1 Tavoitteiden täyttyminen ja parannusehdotukset

Pääpiirteissään sähkötakilan käyttöliittymän ja moottorinohjauksen toteuttaminen onnistui todella hyvin, ja etenkin moottoria käyttävästä ohjelmasta tuli todella joustava ja käyttöliittymästä selkeä ja visuaalisesti näyttävä. Ohjainkotelosta tuli kestävä ja punaiseksi jäänyttä väriä lukuun ottamatta edustavan näköinen. Myös itse rakennettu pulssianturi osoittautui kohtuullisen toimintavarmaksi, vaikka sen toimimisesta olikin alkuun pieniä epäilyksiä. Automaattinen pintaan kelaus, syvyysnäyttö ja kauko-ohjaus toimivat juuri halutulla tavalla. Etenkin ohjauksen rakentamiseen käytettyyn rahamäärään nähden työ onnistui erittäin hyvin.

Sähkötakilasta lähdetään kehittämään myytäväksi kelpaavaa tuotetta, ja yksi tämän opinnäytetyön tärkeimpiä tavoitteita oli rakentaa ohjauksen prototyyppi ja löytää siitä oleellisimmat viat, sekä suunnitteluun ja rakentamiseen liittyvät haasteet. Kuten liitteenä 1 olevan vaatimusluettelon merkinnöistä voi huomata, eivät kaikki vaatimukset toteutuneet halutulla tavalla. Huomattavin puute oli aktiivisen ja syklisen syvyysseurannan puuttuminen, jota ei testaamiseen tarvittavan kaikuluotaimen puuttuessa pystytty kunnolla toteuttamaan. Näihin toimintoihin liittyviä asioita kuitenkin suunniteltiin melko yksityiskohtaisesti vähintäänkin ajatuksen tasolla, ja mikäli Arduinolle olisi saatu syvyysdataa, olisi toimintojen ohjelmoiminen ollut melko yksinkertainen tehtävä. Standardoidun NMEA-liitännän puuttumiseen liittyneet ongelmat päätettiin ratkaista jättämällä liitin kokonaan pois, jolloin NMEA-tiedon vastaanottamiseen tarvittavat 2 johdinta jäisivät käyttäjän kytkettäväksi hänen haluamallaan tavalla. Oikeasta kytkentätavasta olisi luonnollisesti ohjeistus takilan käyttöohjeessa. Tästä huolimatta takilaa tulisi myös etukäteen pystyä testaamaan useilla eri kaikuluotainmalleilla.

Vaikka käyttöliittymä olikin pääosin onnistunut, oli siinä kuitenkin joitakin puutteita. Suurin puute lienee se, että käyttöliittymä ei juurikaan anna käyttäjälleen palautetta toimintojen onnistumisesta tai epäonnistumisesta. Esimerkiksi kuulan saapuessa tavoiteltuun kelaussyvyyteen olisi käyttäjälle voinut näyttää viestin, joka kertoo kelauksen olleen onnistunut.

Enimmät ohjaukseen liittyneet ongelmat liittyivät kuitenkin ohjauksen toteuttaviin komponentteihin, kuten ohjainkotelon kiinnitysratkaisuihin, antureiden kiinnityksiin ja komponenttivalintoihin. 3D-tulostetusta ohjainkotelosta tuli vahva, mutta kosketusnäytön

kiinnitys ei ollut tarpeeksi tukeva eikä varmuudella vedenpitävä. Vaikka ohjainkotelo suunniteltiin kiinnitettävän takilan runkoon neljällä ruuviliitoksella, osoittautui tämä kiinnitystapa ongelmalliseksi kotelon huonon kokoonpantavuuden takia, sillä ruuviliitokset oli lähes mahdotonta kiristää kotelon sisäpuolelta muiden tiellä olleiden osien takia. Tulevassa versiossa ohjainkotelo tullaan toteuttamaan täysin erilailla myös siksi, että 3D-tulostaminen on huono valmistusmetodi suurempia kappalemääriä ajatellen.

Ohjainkotelossa riitti juuri tila kaikkia siihen suljettavia komponentteja varten. Vaikka tila käytettiinkin tehokkaasti hyödyksi, etenkin siirryttäessä Arduino Unosta MEGA:an, oli tämä ongelmana ohjainkoteloa kasattaessa, sillä komponenttien välisiä johtimia joutui asettelemaan paikoilleen huolellisesti. Tehtävää vaikeutti myös käytettyjen komponenttien suurehko määrä. Ratkaisuna tähän voisi olla jonkinlainen mikrokontrollerin ja kosketusnäytön yhdistävä komponentti. Kauko-ohjaimen vastaanotin ja H-silta sen sijaan ovat melko välttämättömiä ohjauksen kannalta. Shield-piirilevy oli toimiva ratkaisu prototyyppikäytössä, mutta kaupalliseen versioon vastaavaa piirilevyä ei todennäköisesti tulla sen hankalan valmistettavuuden takia lisäämään, ainakaan nykyisessä muodossaan, vaan kytkentöjä joko vähennetään järkevämmillä komponenttivalinnoilla tai piirilevy tullaan valmistamaan jotenkin muuten kuin käsityönä kolvaamalla.

Väkipyörään kiinnitettävä mikrokytkin osoittautui erityisen ongelmalliseksi, sillä sitä ei pystytty kiinnittämään väkipyörän runkoon järkevällä tavalla siten, että kytkimen varsi olisi saatu riittävän lähelle takilan narua ja vieläpä siten, että naruun kiinnitetty osa olisi sitä riittävän varmasti liikuttanut. Todettiinkin, että mikrokytkin on parempi jättää kokonaan pois käytöstä, jolloin kuulan kelaaminen liian ylös jäisi täysin käyttäjän vastuulle. Tästä pitäisi olla varoitus myöhemmin kirjoitettavassa käyttöohjeessa.

Kelan rakenne onnistui hyvin, ja siihen yhdistetty pulssianturin pulssikiekko oli toimiva idea. Kelasta ei kuitenkaan tullut erityisen näyttävän näköistä, mutta tämä ongelma olisi mahdollisesti korjattu pintakäsittelyllä ja maalauksella. Kela oli kuitenkin jonkinasteinen turvallisuusriski, sillä käyttäjän oli mahdollista jättää kätensä kelan ja takilan rungon väliin. Tämä voitaisiin estää takilan yläosan aukon kohdalle kiinnitettävällä muovilevyllä, joka samalla tekisi takilasta näyttävämmän. Kelan sivua vasten kiinnitetyt induktiiviset anturit jäivät hieman huteriksi eikä niiden sijainti ollut säädettävissä. Tämä voitaisiin korjata jonkinlaisella ruuvein säädettävällä säätöpalalla, johon anturit kiinnitettäisiin todella tukevasti ja jonka ruuveja kiertämällä antureiden sijaintia pystyisi säätämään tarkasti.

7.2 Uudet jatkosuunnitelmat

Takilaprototyypin tultua lähes valmiiksi teetettiin asiakaskysely kyselynetti.com -sivustolla, johon kerättiin vastaajia kalastajapiireistä sosiaalisen median kautta. Kyselyllä pyrittiin selvittämään, mitkä prototyypissä olleet ominaisuudet olivat tärkeitä potentiaalisten ostajien mielestä, ja tätä tietoa pyrittiin käyttämään hyödyksi takilan kaupallista versiota suunniteltaessa. Vastaajia pyydettiin kertomaan asteikolla 1 - 5 kuinka tärkeänä he pitivät kutakin takilasta listattua ominaisuutta. Kyselyn tulokset on taulukoitu alle, ja vastauksista on laskettu painotetut keskiarvot kunkin kysymyksen kohdalla.

Taulukko 4. Asiakaskyselyn tulokset.

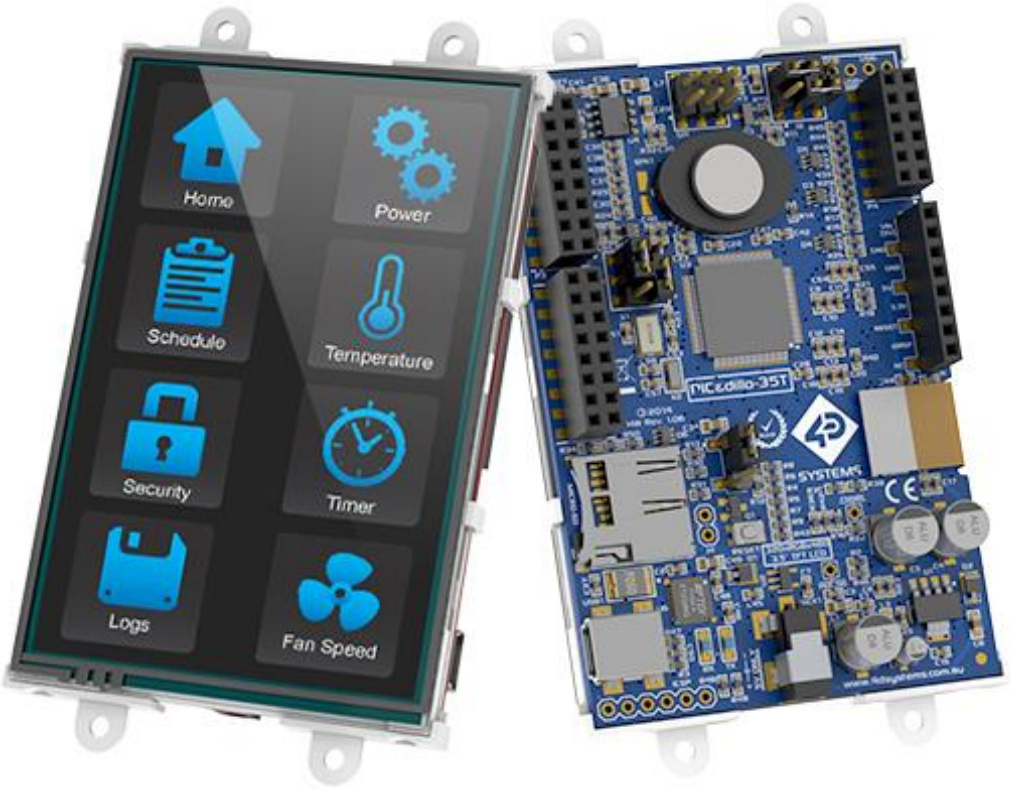
	Vastausten lukumäärä					
	Todella tärkeä				Ei yhtään tärkeä	
Ominaisuus	5	4	3	2	1	**keskiarvo**
Rakenteen yksinkertaisuus	10	10	5	2	1	3,9
Rakenteen purettavuus	3	12	6	4	3	3,3
Kelausnopeus	11	13	3	0	1	4,2
Kauko-ohjaus	8	11	3	5	1	3,7
12V tupakansytytinliitäntä	10	8	1	3	4	3,7
Aktiivinen syvyysseuranta	12	7	5	2	0	4,1

Keskiarvoista huomataan, että kaikki takilassa jo olevat ohjauksen ominaisuudet ovat vähintäänkin kohtalaisen tärkeitä, ja että kelausnopeutta ja aktiivista syvyysseurantaa pidetään todella tärkeinä ominaisuuksina. Näiden tulosten perusteella etenkin takilan kelausnopeutta päätettiin parantaa tekemällä voimansiirron rakenteeseen suuria muutoksia.

Varsinkin työn loppuvaiheessa syntyi paljon uusia ideoita siitä, miten sähkötakilasta voisi tehdä huomattavasti paremman myös täysin uusia ratkaisuja hyödyntäen. Nämä ideat eivät kaikki liittyneet käyttäjäkyselyssä kysyttyihin asioihin, vaan ne olivat kokonaan uusia. Osa niistä kuitenkin liittyi myös kyselyssä listattujen takilan ominaisuuksien parantamiseen. Näitä uusia ideoita syntyi niin mekaniikkaan kuin ohjaukseenkin liittyen, mutta seuraavissa luvuissa keskitytään lähinnä ohjauksen parantamiseen niin kokonaan uusien ideoiden kuin käyttäjätutkimuksen tulostenkin suhteen.

7.2.1 Kosketusnäyttö ja mikro-ohjain

Oleellisin ohjausta koskeva ajatus oli siirtyminen Arduinosta laadukkaampaan ja toimintavarmempaan mikrokontrolleriin ja kosketusnäyttöön. Vaikka Arduino onkin joustava ja monipuolinen työkalu prototyyppien rakentamiseen, ei sitä ole tarkoitettu laajempaan tuotantoon. Tästä syystä sitä pidettiin liian epävarmana valintana takilan kaupallisen version mikrokontrolleriksi. Korvaajaksi löydettiin 4D Systems -nimisen yrityksen valmistama 3,5 tuuman Picadillo-35T resistiivinen kosketusnäyttö (kuvassa 32), jossa on yhdistettynä myös mikrokontrolleri sekä tarvittava määrä I/O-pinnejä ja PWM-lähtöjä.



Kuva 32. 4D Systemsin Picadillo-35T -kosketusnäyttö (4D Systems 2016).

Picadilloa käyttämällä saataisiin nykyisen ohjauksen kaksi merkittävintä komponenttia, Arduino MEGA ja kosketusnäyttö, korvattua yhdellä ainoalla komponentilla, joka olisi vieläpä helppo kiinnittää takilan kuoreen siinä valmiina olevista kiinnitysrei'istä. H-siltaa ja kauko-ohjaimen vastaanotinta sen sijaan tarvittaisiin edelleen, mutta vastaanottimen saisi kiinnitettyä tukevasti suoraan Picadillon portteihin ja H-silta on helppo kiinnittää siinä olevista rei'istä vaikkapa takilan runkoon. Picadillossa on myös pieni sisäänrakennettu kaiutin, joka mahdollistaa käyttöliittymän palautteen antamisen myös äänen muodossa. Komponentissa on lisäksi enemmän sisäistä muistia kuin Arduinossa, ja siinä on

valmiiksi microSD-muistikorttipaikka lisämuistia varten. Päivitystaajuudeltaan tämä näyttö on huomattavasti takilaprototyypissä käytettyä kosketusnäyttöä nopeampi. Huono päivitystaajuus olikin yksi prototyypin näyttävyyttä eniten heikentävistä asioista. Hintaa Picadillolla on hieman enemmän, kuin mitä Arduino MEGA ja Adafruitin kosketusnäyttö yhteensä maksavat, mutta sen edut ovat takilan kannalta selvästi suuremmat. (4D Systems 2016.)

7.2.2 Usean takilan samanaikainen käyttö

Uuden kosketusnäyttötyypin käyttöönoton lisäksi suurin käyttöliittymään liittyvä lisäys olisi tuki kahden (tai useamman) yhtäaikaisesti käytössä olevan sähkötakilan ohjaukselle samalta näytöltä. Ajatus tämän ominaisuuden kehittämiselle lähti siitä tosiasiasta, että monesti veneessä on yksi takila veneen kummallakin laidalla. Nämä kaksi takilaa voitaisiin yhdistää toisiinsa joko kaapelilla tai langattomasti siten, että kumpaakin takilaa pystyisi ohjaamaan kummalta tahansa näytöltä. Ohjaus voitaisiin toteuttaa joko isäntä-orja -periaatteella siten, että toinen ohjaimista ottaa komennon kummastakin takilasta, tai siten, että kummalta tahansa näytöltä pystyisi tekemään muutoksia sekä paikalliseen että etäohjattavaan takilaan. Paikalliselta takilalta etätakilaan tehdyt muutokset päivittyisivät etätakilan omalle näytölle reaaliajassa.

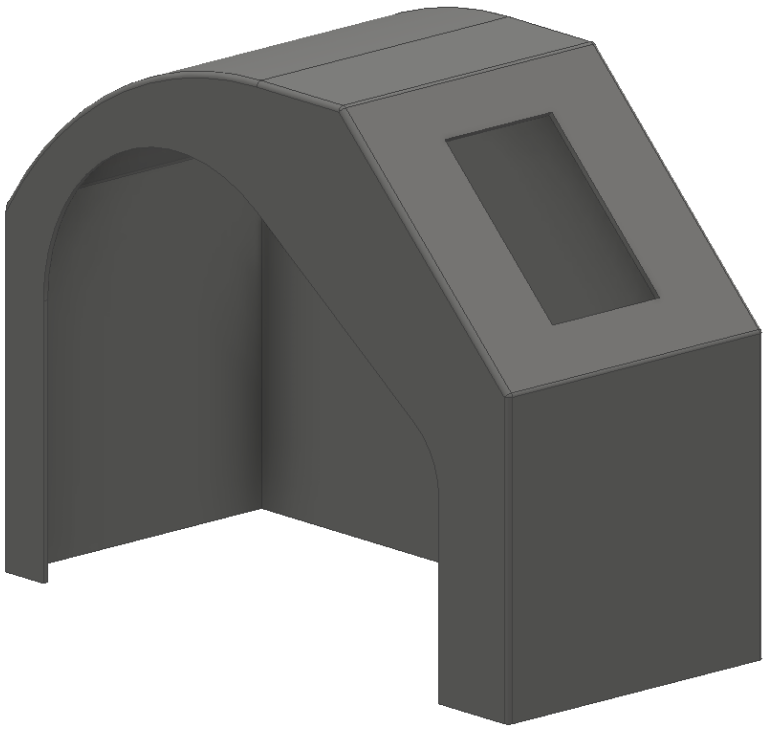
Tällä ominaisuudella saavutettaisiin se hyöty, että kalastajan ei tarvitsisi liikkua niin paljon edes takaisin veneessä kahden takilan välillä, vaan hän voisi tehdä kumpaakin takilaa koskevia toimintoja vain yhtä päätettä käyttäen. Tämä ominaisuus voidaan ajatella yhdeksi myyntivaltiksi, sillä vastaavaa ominaisuutta ei kaupallisista takiloista vielä löydy. Sähkötakilasta tehdyn käyttäjäkyselyn tulosten mukaan tällainen ominaisuus olisi haluttu potentiaalisten asiakkaiden keskuudessa.

Käytännössä ominaisuuden toteuttaminen vaatisi useampinapaisen johtimen takiloiden väliin, mikä olisi asennuksen kannalta hankalampi, mutta varmasti toimintavarmempi, vaihtoehto. Siistimpi ratkaisu olisi käyttää langatonta yhteyttä, mutta yhteyden tulisi toimia varmasti noin 5 metrin säteellä, millä on suuri vaikutus tiedon lähetin- ja vastaanotinpiirien valintaan. Langattoman yhteyden vaatimat komponentit tulisivat myös varmasti paljon langallista yhteyttä kalliimmaksi vaihtoehdoksi. Tämän ominaisuuden kehittely vaatiikin siis vielä paljon eri asioiden selvittämistä.

7.2.3 Uusi runkorakenne ja ohjaimen kotelointi

Vaikka prototyyppiin 3D-tulostettu ohjainkotelo olikin toimiva ratkaisu, ei sitä voida käyttää takilan markkinoille kelpaavassa versiossa, koska kyseisenlaisen kotelon tulostaminen tulisi aivan liian kalliiksi. Lisäksi valmistusmenetelmä on todella hidas. Tästä syystä ohjaimen kotelointi tuleekin suunnitella kokonaan uudelleen. Kuten seuraavassa luvussa kerrotaan, on myös takilan voimansiirtoon päätetty tehdä huomattavia muutoksia. Näistä syistä takilan runkorakenne tullaan suunnittelemaan uudelleen siten, että se mahdollistaa sekä uuden voimansiirron vaatimat rakenteet että moottorinohjaimen yksinkertaisemman koteloinnin.

Alustavasti runko on ajateltu muuttaa L-kirjaimen malliseksi siten, että takilan kela tulisi rungon ulkopuolelle pystysivuun kiinnitettynä, jolloin rungon ympärille voidaan rakentaa lasikuidusta vesitiivis kotelointi (kuten kuvassa 33) esimerkiksi puusta valmistettua rungon muotoon tehtyä muottia käyttäen. Kotelo värjättäisiin mustaksi ja siihen tehtäisiin kosketusnäytölle aukko, joka tiivistettäisiin liimaamalla sen päälle kosketuskalvoa samaan tapaan kuin takilan prototyypin ohjainkotelon kohdallakin tehtiin. Kotelon avoin reunus kiinnitettäisiin teräslevystä kantattua runkolevyä vasten ruuviliitoksin, ja liitos tiivistettäisiin tiivistysnauhalla.

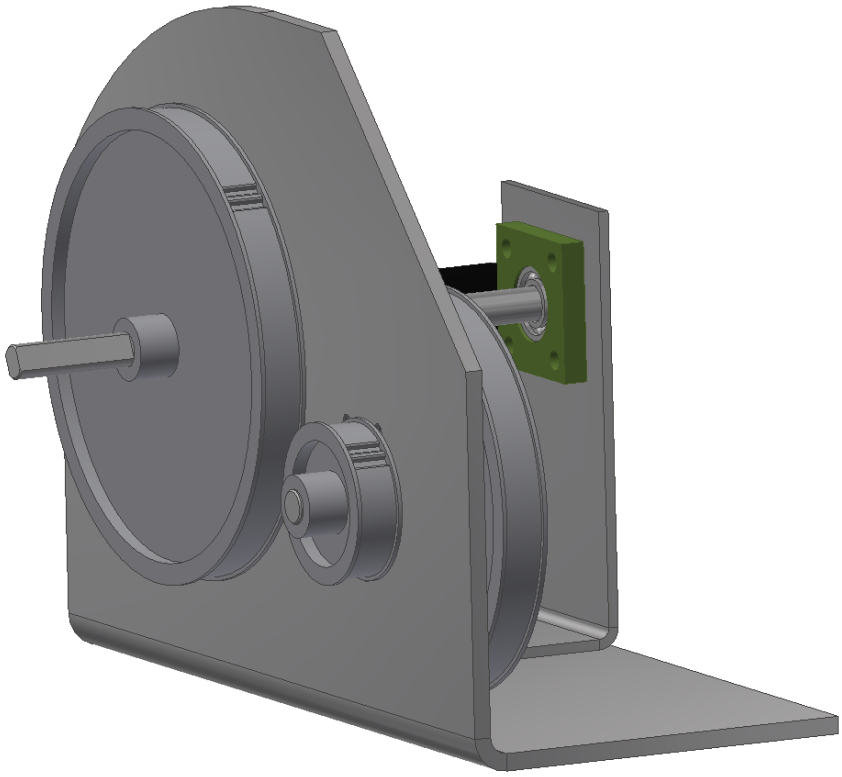


Kuva 33. Suunniteltu uusi kotelorakenne. Kosketusnäyttö käännetään pystyyn, jotta takilasta saadaan mahdollisimman kapea.

Tähän rakenteeseen päädyttiin, koska siitä voidaan tehdä kohtuullisen helposti vesitiivis, ja sen muotoilu tulee olemaan melko yksinkertaista. Koska kotelo liu'utetaan ylhäältä takilan rungon ympärille, jää kotelon yläpinta yhtenäiseksi ja siitä on sopivaa muottia vasten helppo tehdä näyttävän muotoinen. Kotelo myös ratkaisee takilan käyttöturvallisuutta koskevan ongelman, sillä se peittäisi vaihteiston kokonaan siten, ettei käyttäjä saisi sormiaan minkään liikkuvien osien väliin.

7.2.4 Voimansiirto, jarru ja pyörimisen anturointi

Kuten jo edellä mainittiin, tullaan myös takilan voimansiirtoon tekemään huomattavia muutoksia. Suoran välityksen sijaan kelan pyörimisliike tullaan siirtämään moottorilta kelan akselille kahta hihnavaihdetta käyttäen (kuten kuvassa 34). Näin saavutetaan parempi vääntömomentti ja pyörimisnopeus pienemmällä moottorilla. Hihnavälityksen komponentit kiinnitetään L-kirjaimen muotoiseen runkolevyyn, joka on prototyypissä käytettyyn U-kirjaimen muotoiseen runkoon verrattuna yksinkertaisempi ja helpompi valmistaa.



Kuva 34. Aikainen 3D-malli takilan uudesta runkorakenteesta ja voimansiirrosta. Toinen hihnavaihde tullaan sijoittamaan rungon ulkopuolelle tilan säästämiseksi, mutta vaihde koteloidaan erillisellä kotelorakenteella siten, ettei siihen pääse käsiksi.

Nämä muutokset eivät koske ainoastaan takilan mekaniikkaa, vaan niillä tulee olemaan vaikutuksia myös ohjauksen toimintaan. Koska hihnavälityksen pitomomentti on prototyypin moottorissa olleen matovaihteen pitomomenttiin verrattuna lähes olematon, tulisi yhteen hihnapyöristä rakentaa jonkinlainen jarrumekanismi, jota ohjattaisiin ohjausjärjestelmän I/O-porttien kautta päälle tai pois. Jarru voisi olla yksinkertaisinta toteuttaa jonkinlaisella solenoidikäyttöisellä mekanismilla, jossa solenoidi painaa hihnaa vasten hihnaprofiilipalan, joka pysäyttää hihnan liikkeen. On olemassa myös suoraa sähkömoottorin päätyyn kiinnitettäviä jarruja, mutta sellaista ei haluta käyttää, koska se vaatisi takilan ulkomittojen liiallista suurentamista. Tyypiltään jarrun tulisi olla normaalisti suljettu, jotta takilan kuula ei lähtisi putoamaan sähkökatkoksen aikana. Tämä tarkoittaisi sitä, että aina kelaa liikutettaessa tulisi ohjaimelta antaa sekä moottorin pyörimissignaali että jarrun vapauttava signaali.

Uusi vaihteisto mahdollistaisi myös syvyystiedon anturoinnin muuttamisen paremmaksi. Koska käytettävän moottorin suhteen ei nyt olisi niin kovia vaatimuksia, voitaisiin etsiä moottori, jossa olisi yhdistettynä myös pulssianturi. Valmiin pulssianturin resoluutio olisi huomattavasti itse rakennettua parempi, mikä tarkoittaa tarkempaa syvyyslukemaa kosketusnäytöllä. Pulssianturi olisi myös toimintavarmempi ja se yksinkertaistaisi takilan rakennetta. Mikäli sopivaa pulssianturin sisältävää moottoria ei kuitenkaan löydetä, voitaisiin prototyypin pulssikiekkoa vastaava rakenne suunnitella helposti myös osaksi uutta vaihteistoa, ja induktiiviset anturit voitaisiin sulkea uuden ohjainkotelon sisään jossa ne eivät olisi lainkaan käyttäjän nähtävillä.

7.3 Markkinointi ja kaupallistaminen

Kustannukset, näyttävyys ja kaupallinen arvo ovat asioita mitkä pidettiin mielessä kaiken aikaa sähkötakilaprototyyppiä rakentaessa, sillä kuten luvussa 4 todettiin, on tavoitteena jo melko aikaisesta vaiheesta ollut tehdä takilasta myytävä tuote. Kaupallisen menestymisen saavuttamista on tavoiteltu ja tullaan tavoittelemaan monin tavoin. Takilaan on esimerkiksi kaiken aikaa pyritty lisäämään sellaisia uusia ominaisuuksia, joita ei markkinoilla olevissa sähkötakiloissa vielä ole, mutta jotka kuitenkin jollakin tavalla parantaisivat tuotteen käytettävyyttä. Tärkeimpiä tällaisia ominaisuuksia ovat takilan kauko-ohjattavuus ja käyttöliittymän älykkäät toiminnot. Edellä kuvattu usean sähkötakilan tuki tulee olemaan yksi tällainen myyntivaltti.

Sähkötakilan rakentaminen lähti alun perin käyntiin kustannussyistä. Koska markkinoilla tarjolla olevat sähkötakilat ovat kalliita, pyrittiin tästä takilasta tekemään mahdollisimman halpa karsimalla siitä pois kaikki ylimääräiset ominaisuudet ja turhat osat, ja keskittymällä oleellisten toimintojen saavuttamiseen yksinkertaisilla ja halvoilla ratkaisuilla. Takilaprototyypin ohjauksen lopullisiksi kustannuksiksi tuli arvioiden noin 170 €, kuten taulukossa 5 on laskettu. Koko takilan hinnaksi runko, voimansiirto ja muu mekaniikka mukaan lukien tuli noin 300 €, mikä oli todella hyvä saavutus asetettuun hintavaatimukseen nähden.

Taulukko 5. Takilaprototyypin ohjausjärjestelmän rakentamisesta aiheutuneet kustannukset. Listasta on jätetty pois komponentit, joita takilassa ei päädytty käyttämään.

Osa tai palvelu	**Hinta n. €**
Tuulilasinpyyhkijänmoottori	25,00 €
Laserleikkeet (kela + näytön kiinnike)	30,00 €
Kosketusnäyttö	50,00 €
Arduino MEGA (Kiina)	10,00 €
H-silta	10,00 €
Kauko-ohjain + vastaanotin	10,00 €
Ohjainkotelo *	0 €
Sekalaiset	25,00 €
yht.	**160,00 €**

*Ilmaiseksi TAMKilta, todellinen hinta noin 300 €

Ohjaus onnistuttiin siis rakentamaan todella halvalla, mikä on loistava asia takilan kaupallistamisen kannalta. Takilan myyntihinta on haluttu pitää alle 1000 €:ssa, mieluiten noin 800 € alueella, mikä vaikuttaisi prototyypin valmistuskustannuksiin nähden helposti saavutettavalta tavoitteelta. Myyntiin tarkoitetun version ohjauksen valmistuskustannukset tulevat kuitenkin hieman kasvamaan prototyypin kustannuksista, ja niitä on arvioitu taulukossa 6.

Taulukko 6. Arvio takilan kaupallisen version ohjauksen valmistuskustannuksista.

Osa tai palvelu	**Hinta n. €**
150 W tasavirtamoottori	30,00 €
Pulssianturin osat	30,00 €
Picadillo-35T	100,00 €
H-silta	10,00 €
Kauko-ohjain + vastaanotin	10,00 €
Ohjainkotelo	30,00 €
Jarrun komponentit	30,00 €
Sekalaiset	20,00 €
yht.	**260,00 €**

Suurimmat ohjauksen hintaa kasvattavat tekijät tulevat olemaan 4D Systemsin kosketusnäytön suurempi hinta, sekä pulssianturin ja ohjainkotelon hankkimisesta aiheutuvat kustannukset, joita prototyyppiä valmistaessa ei juuri ollut. Lisäksi erillisen jarrun valmistaminen tulee olemaan kokonaan uusi kulu, mutta sen hinta saadaan luultavasti pysymään noin 30 €:ssa. Vaikka myös mekaniikkaan liittyvät kulut tulevat nousemaan, tulevat takilan valmistuskustannukset varmasti pysymään alle 500 €:ssa.

Jotta takilaa voidaan laillisesti myydä EU:n alueella, tulee sen täyttää ainakin CE-merkinnän koneelle asettamat turvallisuusvaatimukset. Merkin saamista varten tulee takilan uudesta versiosta suunnitella riittävän turvallinen, eikä käyttäjän saa olla mahdollista esimerkiksi saada takilasta sähköiskua tai jättää sormiaan sen pyörivien osien väliin. Takilasta tulee myös kirjoittaa käyttöohje, joka kuvaa sen toimintaa kattavasti.

Takilan markkinointiin tulee kiinnittää jatkossa enemmän huomiota, sillä sen tunnettavuuden kasvattaminen etenkin kalastajapiireissä lisää tuotteen myyntiä. Tätä markkinointia tullaan aluksi tekemään ainakin sosiaalisen median kautta, sanallisesti takilaa mainostamalla sekä mahdollisesti jakamalla mainoslehtisiä erilaisissa kalastustapahtumissa. Myöhemmässä vaiheessa markkinointiin voitaisiin keskittää enemmän rahaa esimerkiksi internet- tai lehtimainonnan muodossa, sekä antamalla takiloita arvosteltavaksi eri medioihin esimerkiksi lehtiarvostelun vastineeksi.

LÄHTEET

Arduino. 2015. Arduino Software (IDE). Päivitetty 7.9.2015. Luettu 7.2.2016. https://www.arduino.cc/en/Guide/Environment

Arduino. 2016. Getting Started with Arduino on Windows. Päivitetty 15.2.2016. Luettu 7.2.2016. https://www.arduino.cc/en/Guide/Windows

Arduino. 2016. Language Reference. Luettu 7.2.2016. https://www.arduino.cc/en/Reference/HomePage

Arduino. 2016. Libraries. Luettu 7.2.2016. https://www.arduino.cc/en/Guide/Libraries

Arduino. 2016. Shields. Luettu 7.2.2016. https://www.arduino.cc/en/Main/arduinoShields

Arduino. 2016. What is Arduino? Luettu 6.2.2016. https://www.arduino.cc/en/Guide/Introduction#

ASCII Codes Table. 2016. Standard Characters. Päivitetty 2016. Luettu 16.3.2016. http://ascii.cl/

Betke, K. 2000. The NMEA 0183 Protocol. Julkaistu 5.2000. Päivitetty 8.2000. Luettu 28.3.2016. http://www.tronico.fi/OH6NT/docs/NMEA0183.pdf

Brain, M. 2006. How does a brushless electric motor work? Julkaistu 15.12.2006. Luettu 11.2.2016. http://electronics.howstuffworks.com/brushless-motor.htm

Brown, J. 1998. Brief H-Bridge Theory of Operation. Julkaistu 4.1998. Päivitetty 9.2002. Luettu 11.2.2016. http://www.dprg.org/tutorials/1998-04a/

Bruce, J. 2013. Arduino vs Raspberry Pi – Which is the Mini Computer for You? Julkaistu 24.5.2013. Luettu 22.2.2016. http://www.makeuseof.com/tag/arduino-vs-raspberry-pi-which-is-the-mini-computer-for-you/

Choudhary, H. 2012. How to program a microcontroller | How to burn a microcontroller. Päivitetty 2012. Luettu 5.2.2016. http://www.engineersgarage.com/tutorials/how-to-program-a-microcontroller

Circuits Today. 2013. Microcontroller – Invention History and Story Behind the Scenes. Päivitetty 23.10.2013. Luettu 5.2.2016. http://www.circuitstoday.com/microcontroller-invention-history

CVEL. 2016. Brushed DC Motors. Luettu 11.2.2016. http://www.cvel.clemson.edu/auto/actuators/motors-dc-brushed.html

Earl, B. 2015. Brushed DC Motor Control. Julkaistu 21.5.2014. Päivitetty 4.5.2015. Luettu 11.2.2016. https://learn.adafruit.com/adafruit-motor-selection-guide/dc-motor-control

forum.arduino.cc. 2011. TinyFAT Library. Julkaistu 27.2.2011. Luettu 7.2.2016. http://forum.arduino.cc/index.php?topic=53742.0

Hirzel, T. 2016. PWM. Päivitetty 2016. Luettu 11.2.2016. https://www.arduino.cc/en/Tutorial/PWM

Hutri, J. & Hutri, T. 2016. Takilan käyttö - takilalla saat uistimet oikealle syvyydelle. Luettu 3.2.2016. http://www.hongkong.fi/fi/info/kalastusvinkit-takilan_kaytto_takilalla_uistimet_oikealle_syvyydelle.html/

Jouppila, V. 2015. Mechatronics: Engineering Design Process. Mechatronics_EngineeringDesignProcess.pdf. Päivitetty 8.9.2015. Tuntiopettaja.

Motonet. 2016. Cannon Easi-Troll ST Takila. Luettu 3.2.2016. http://www.motonet.fi/fi/tuote/561761/Cannon-Easi-Troll-ST-takila

Motonet. 2016. Scotty High Performance Tele 5-2116 sähkötakila. Luettu 3.2.2016. http://www.motonet.fi/fi/tuote/567485/Scotty-High-Performance-Tele-5-2116-sahkotakila

Mäkelä, M., Soininen, L., Tuomola, S. & Öistämö, J. 2012. Tekniikan kaavasto. 10. painos. Tampere: Tammertekniikka.

Mäkelä, S. 2014. Anturitekniikka ja koneautomaation komponentit. Anturitekniikka ja koneautomaation komponentit.pptx. Kevät 2014. Tuntiopettaja.

NC State University. 2015. What is Mechatronics? Päivitetty 12.11.2015. Luettu 4.2.2016. http://www.engr.ncsu.edu/mechatronics/what-mech.php

OEM Automatic. 2016. Pulssianturien teoriaa. Luettu 10.2.2016. http://www.oem.fi/Tuotteet/Anturi/Pulssianturit/Yleista/Pulssianturien_teoriaa/825723-526144.html

Savolainen, J. 2011. Pulssianturi. Päivitetty 28.4.2011. Luettu 10.2.2016. https://wiki.metropolia.fi/display/koneautomaatio/Pulssianturi

Techexplainer. 2012. Resistive vs Capacitive Touchscreen. Julkaistu 2.4.2012. Luettu 26.2.2016. https://techexplainer.wordpress.com/2012/04/02/resistive-vs-capacitive-touchscreen/

Usability.gov. 2016. User Interface Design Basics. Luettu 12.2.2016. http://www.usability.gov/what-and-why/user-interface-design.html

WiseGEEK. 2016. What is a microcontroller? Luettu 5.2.2016. http://www.wisegeek.org/what-is-a-microcontroller.htm

4D Systems. 2014. Picadillo-35T. Luettu 11.4.2016. http://www.4dsystems.com.au/product/Picadillo_35T/

LIITTEET

Liite 1. Vaatimusluettelo

Sähkötakilan ohjausjärjestelmän rakentamiseksi koottiin seuraava vaatimusluettelo. Luettelosta on jätetty pois mekaniikkaan liittyviä ja muita ohjauksen ulkopuolelle jääviä kohtia. Oikeanpuoleiseen sarakkeeseen on merkitty vaatimuksen täyttyminen.

	Vaatimus on täytetty.
	Vaatimuksen täyttämisessä on puutteita.
	Vaatimusta ei ole täytetty.

Mahdollisimman halpa	
Koko prototyypin valmistuskustannukset korkeintaan 500 €	
Ohjaus alle 200 €	
Yksinkertainen rakenne	
Vähän osia	
Ohjaus on toteutettu vähillä komponenteilla	
Yksinkertainen valmistaa	
Ohjaus koostuu valmiista komponenteista	
Nopea kuulan ylös kelaus	
Moottori kelaa vähintään 50 rpm	
Jaksaa nostaa 10 kg massan	
Moottorilla riittävän suuri vääntömomentti	
Mahdollisimman pienikokoinen	
Komponenttien sijoittelu tilaa tehokkaasti käyttäen	
Ohjainkotelon tila on tehokkaasti käytetty	
Anturointi ei kasvata takilan ulkomittoja	
Ulkonääöltään näyttävä	
Kelan osat ovat näyttävät	

(jatkuu)

Ohjainkotelo on hyvän näköinen	
Tyylikäs muotoilu	
Mahdollisimman vähän painikkeita ja kytkimiä	
Johdotukset ovat siistit ja piilossa	
Helppo käyttää	
Selkeä käyttöliittymä	
Suuret painikkeet ja teksti	
Selkeä asettelu	
Näytöllä ei ylimääräisiä elementtejä	
Selkeät toiminnot	
Etäohjattava	
Monipainikkeinen kaukosäädin	
Helppo huoltaa	
Runkorakenne on helppo purkaa	
Ohjainkotelo on helppo purkaa	
Käyttöliittymä usealle kielelle	
Suomi, englanti ja ruotsi	
Lisää kieliä tarpeen mukaan helposti lisättävissä	
Syvyyden näyttö usealla eri yksiköllä	
Metrit ja jalat	
Säänkestävä	
Ohjauskotelo on vesitiivis	
Vettä kestävä materiaali	
Näytön aukko on tiivistetty	
Kotelon taka-aukko on tiivistetty	
Johtojen läpiviennit ovat tiiviitä	
Anturit ja mikrokytkin ovat vesitiiviitä	
Turvallinen käyttää	
Ylös kelauksen automaattinen pysäytys	
Väkipyörässä anturointi kuulan pysäyttämistä varten	
Kelauksen hätäpysäytys	
Narussa on narunkatkaisija	
Ei mahdollista jättää sormiaan kelan ja rungon väliin	

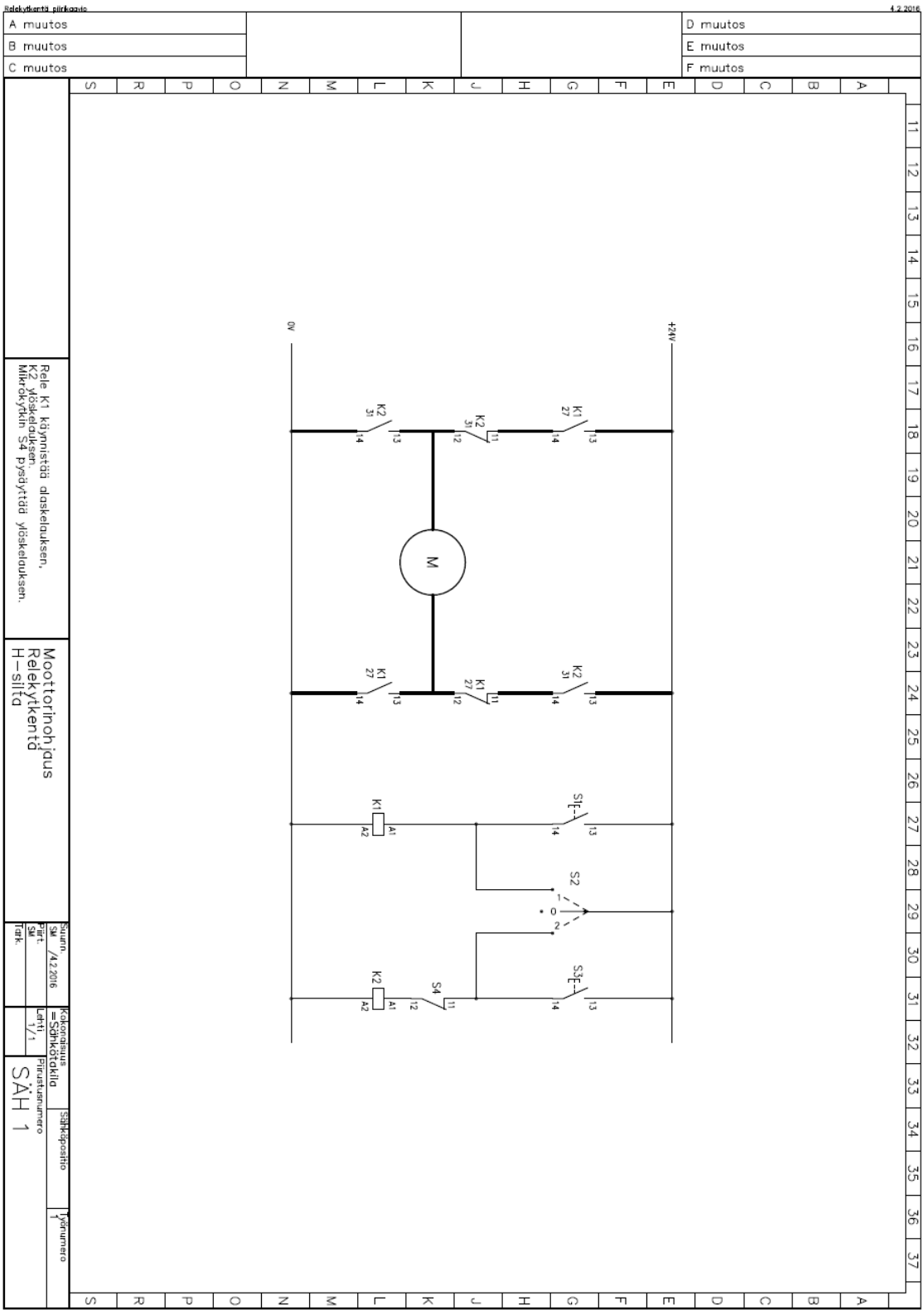
(jatkuu)

Ei ylikuumene	
Moottori	
Sähkökomponentit	
Toimiva moottorinohjaus	
Kuulan kelaaminen ylös ja alas manuaalisesti	
Näytöltä	
Kaukosäätimestä	
Kelauksen hätäpysäytys	
Näytöltä	
Kaukosäätimestä	
Kulloiseenkin tilanteeseen sopivat moottorin kiihdytysrampit	
Hitaammassa hienosäätökelauksessa suuri kiihtyvyys	
Hätäpysäytyksessä suuri kiihtyvyys	
Nopeassa syvyyteen kelauksessa pienempi kiihtyvyys	
Automaattinen pintaan kelaus	
Kelaussyvyyden anturointi	
Pintatason kalibrointi	
Näytöltä	
Kaukosäätimestä	
Automaattinen haluttuun syvyyteen kelaaminen	
Kelaussyvyyden anturointi	
Numeroarvon syöttäminen näytöltä	
Kelausnopeuden hidastus syvyyteen saavuttaessa	
Aktiivinen syvyysseuranta	
Järven syvyyden anturointi	
Seurantaetäisyyden asetus	
Syklinen syvyysseuranta	
Järven syvyyden anturointi	
Seurantaetäisyyden asetus x 2	
Syklin aikavälin asetus	
Kestävä	
Johdotukset eivät irtoile	
Liitinvalinnat ovat kestäviä	

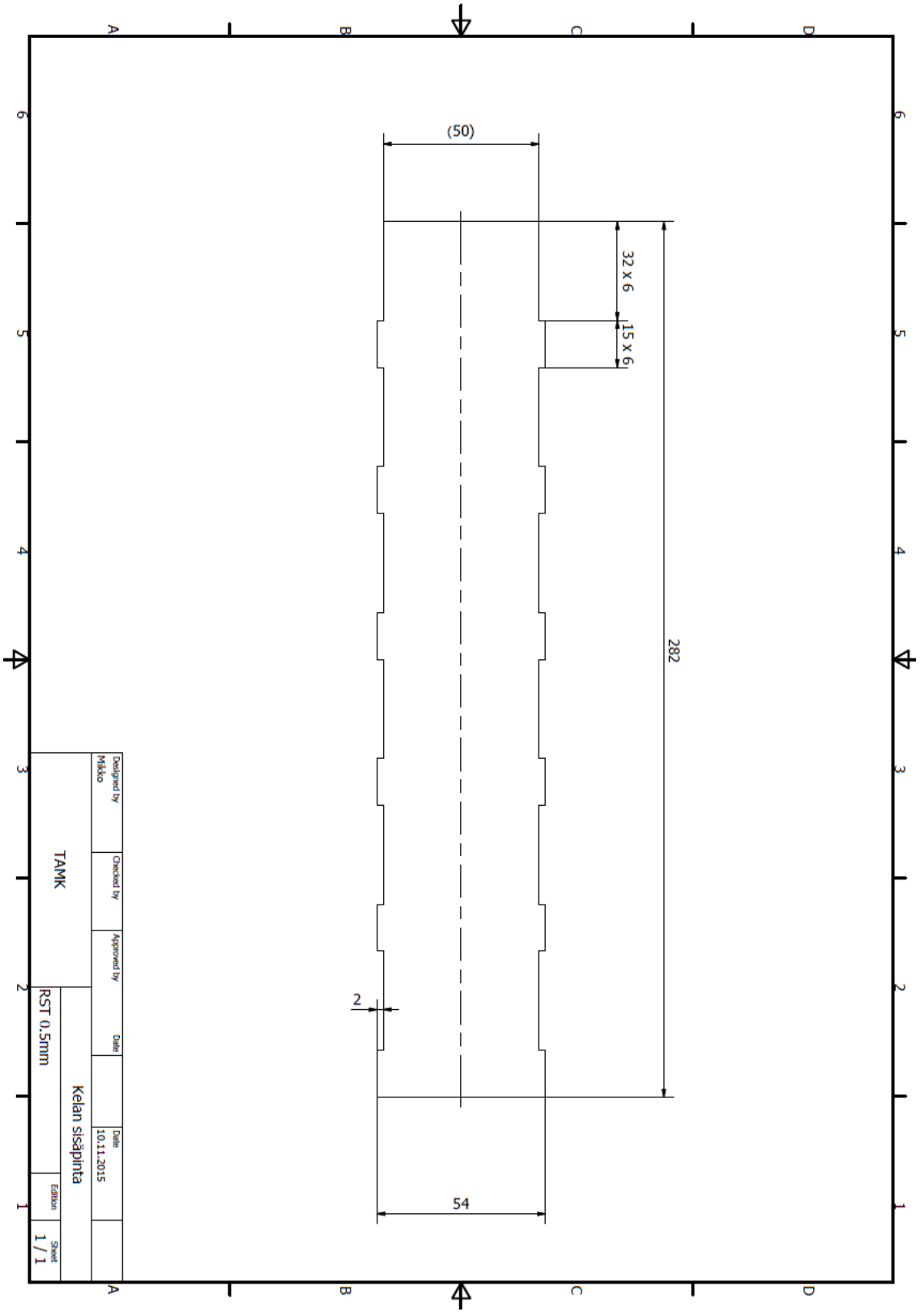
(jatkuu)

Ohjainkotelo on kestävä	
Kotelon rakenne on vahva	
Kotelo tukevasti kiinni rungossa	
Näyttö on hyvin kiinnitetty	
Anturit on kiinnitetty tukevasti	
Mikrokytkin	
Induktiiviset anturit	

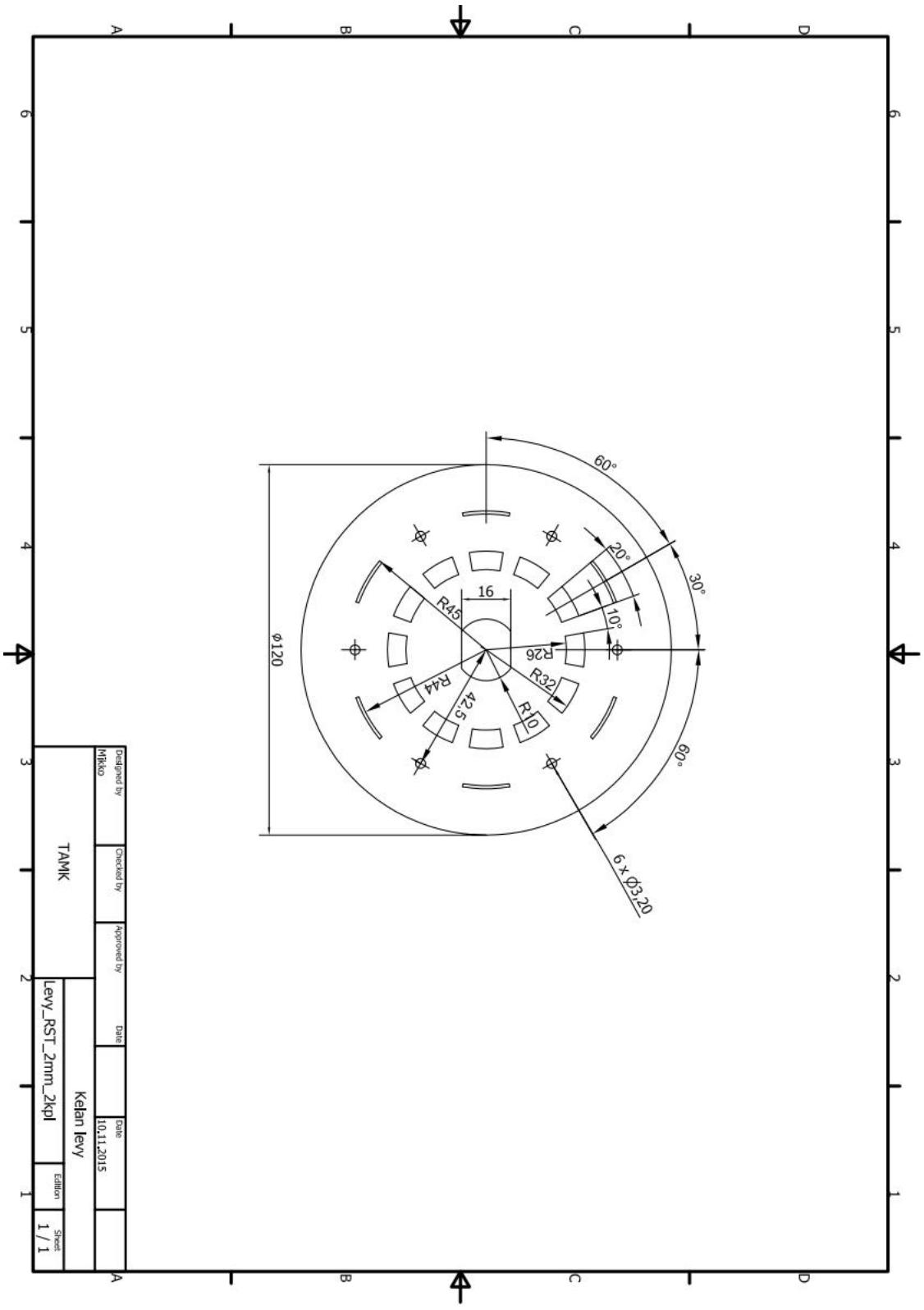
Liite 2. Ensimmäisen prototyypin releohjauksen piirikaavio



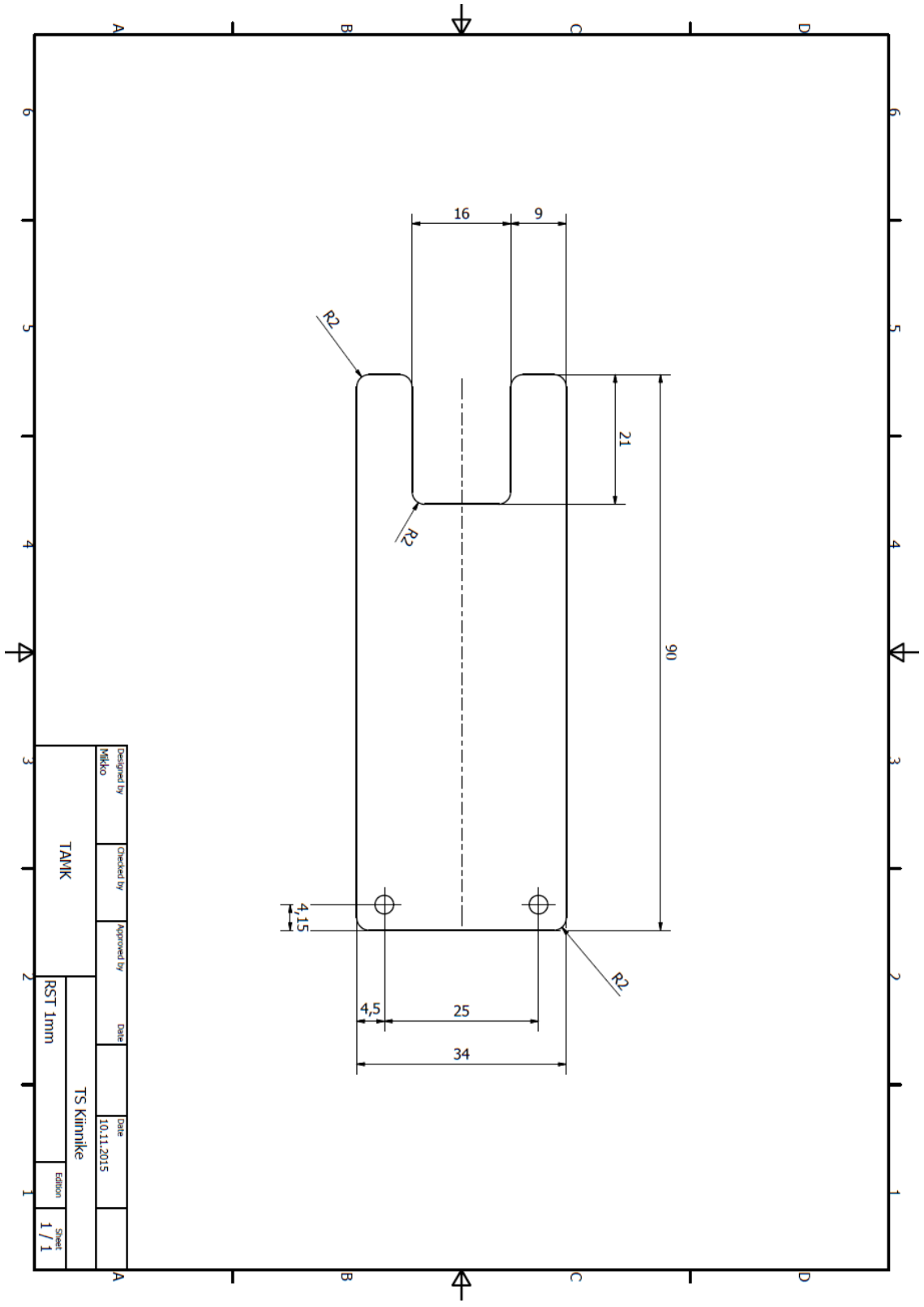
Liite 3. Laserleikkeiden valmistuspiirustukset



(jatkuu)

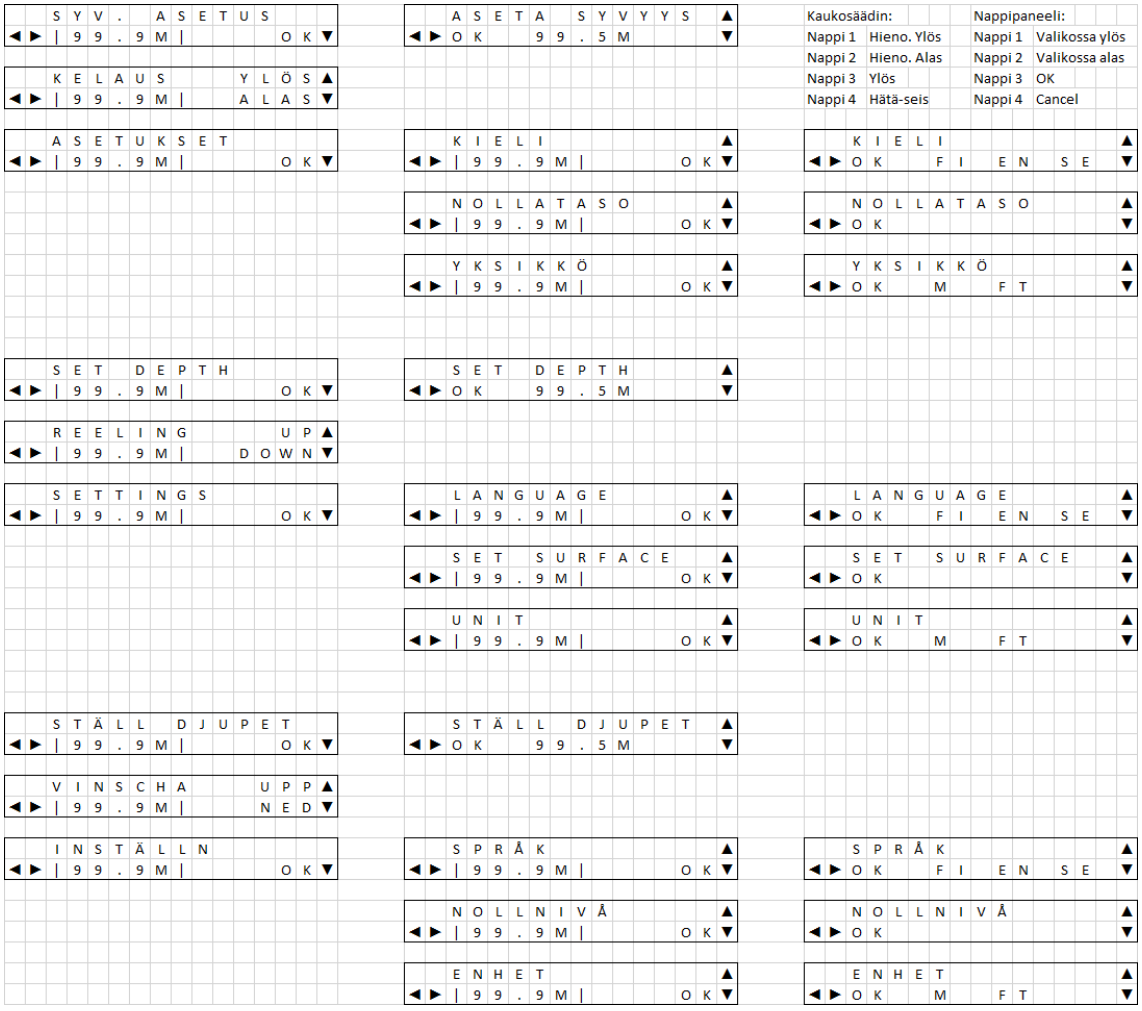

60°
20°
30°
10°
16
R45
R26
R32
R44
42,5
R10
ø120
60°
6 x Ø3,20
Designed by
Mikko
Checked by
Approved by
Date
Date
10.11.2015
TAMK
Kelan levy
Levy_RST_2mm_2kpl
Edition
Sheet
1 / 1

(jatkuu)


16
9
R2
21
R2
90
4,15
R2
4,5
25
34
TAMK
Designed by
Mikko
Checked by
Approved by
Date
Date
10.11.2015
RST 1mm
TS Kiinnike
Edition
Sheet
1 / 1

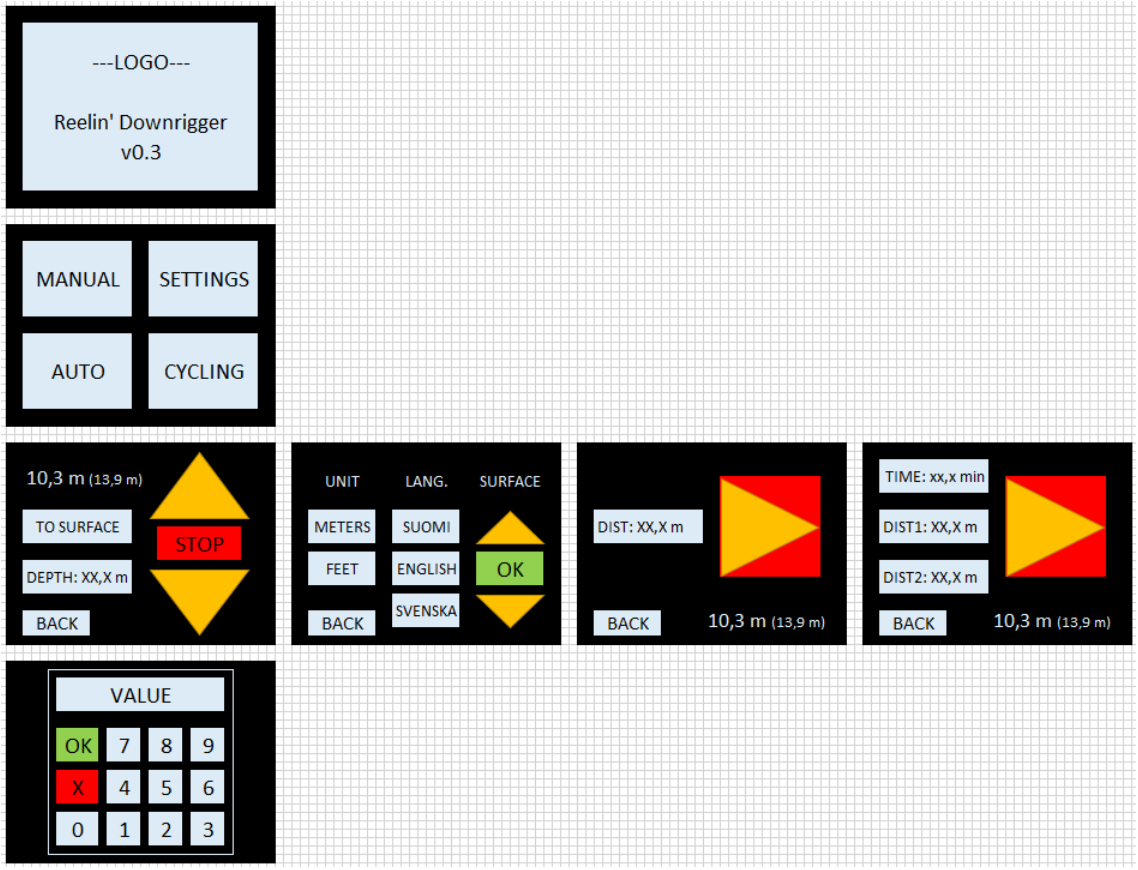
Liite 4. Ensimmäinen valikkorakennesuunnitelma

Alla on LCD-näyttöä varten tehdyn valikkorakenteen suunnitelma. Vasemmanpuolimmaisimmassa sarakkeessa ovat 3 ylimmän tason valikkoa. Niistä pääsee keskimmäisen sarakkeen alivalikkoihin, joista edelleen pääsee oikealla oleviin alimman tason asetusvalikkoihin. Saman tason valikoiden välillä voi liikkua kahdella näppäinpaneelin näppäimellä. Kaksi muuta näppäintä on tarkoitettu valikon tasojen välillä liikkumista ja asetusten tekemistä varten. Kuvassa yksi ruutu vastaa yhtä LCD-näytölle mahtuvaa merkkiä. Samalla suunniteltiin kaukosäätimen painikkeiden toiminnot.



Liite 5. Lopullinen valikkorakennesuunnitelma

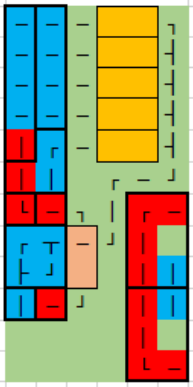
Alla on käyttöliittymän lopullista valikkorakennetta varten tehty graafinen suunnitelma. Kuvassa ylin ruutu on sähkötakilan käynnistymisruutu, joka näkyy muutaman sekunnin ajan, kun takilaan kytketään virta. Siitä siirrytään toisella rivillä näkyvään päävalikkoon, jonka neljästä painikkeesta pääsee jokaiseen 3. rivillä olevista valikoista. Näihin valikkoihin on jaoteltu kaikki takilan toiminnot. Jotkin näistä valikoista sisältävät tekstikenttiä, joihin käyttäjä voi syöttää lukuarvoja. Arvojen syöttäminen tapahtuu alimpana näkyvällä numeronäppäimistösivulla. Kuvassa yksi ruutu vastaa 10 pikseliä kosketusnäytöllä. Tästä oli suuri hyöty ohjelmointivaiheessa eri elementtejä paikoittaessa.



Liite 6. Apupiirilevyn Excel-kaavio

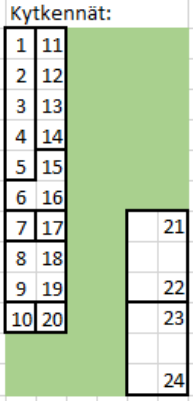
Alla on sähkötakilan kytkentöjen helpottamista varten valmistetun apupiirilevyn periaatekuva. Piirilevy tehtiin reikäpiirilevylle, ja kuvassa yksi ruutu vastaa yhtä piirilevyn reikää. Ylempään kaavioon piirretyt viivat kuvaavat tinattuja johteita eri reikien välillä. Vasemmalla oleva 2 x 10 ruudun kokoinen alue koostuu 1 x 5 kokoisista liitinrimoista, joihin näppäinpaneelin, LCD-näytön, Arduinon ja mikrokytkimen johtimet kiinnitettiin. Oikealla olevat 2 x 3 ruudun kokoiset alueet ovat 2 riviliitintä, joiden kautta jaettiin virtaa eri komponenteille.

Virran kulku:



	10 kΩ vastus, ruskea-musta-oranssi-kulta
	220 Ω vastus, punainen-punainen-ruskea-kulta
	+5V
	GND

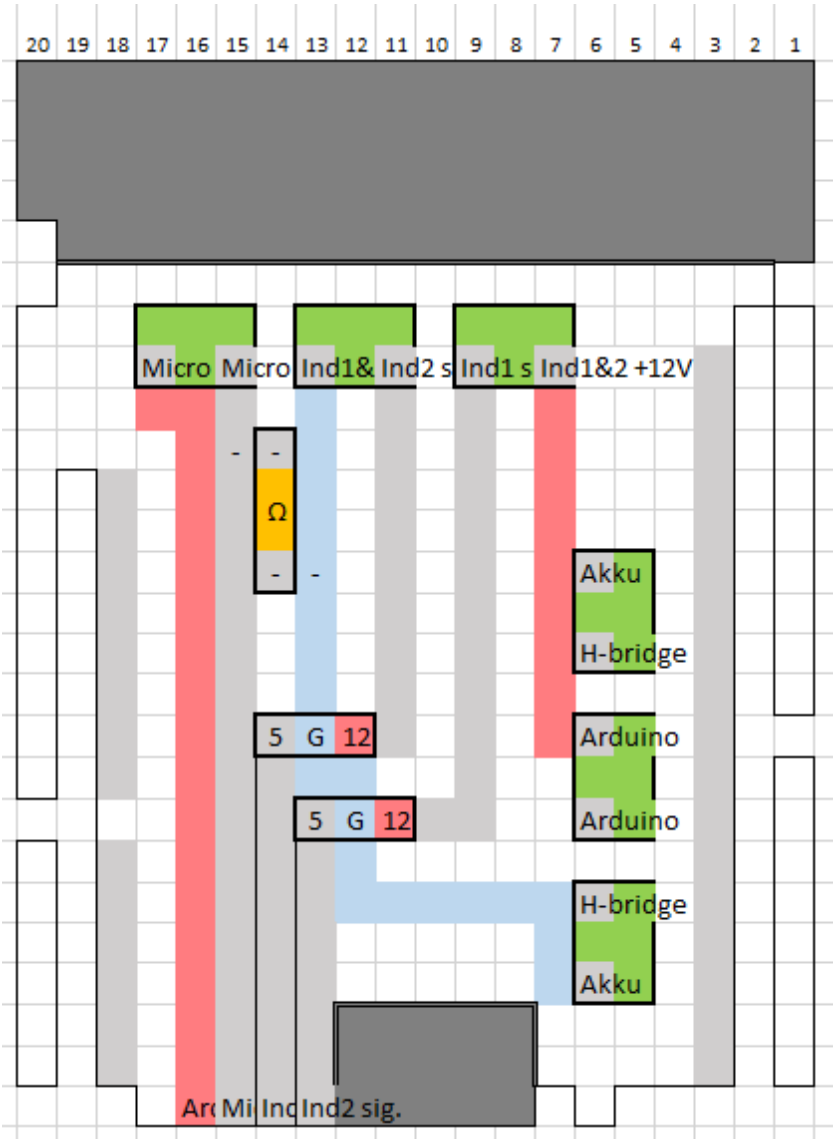
Kytkennät:



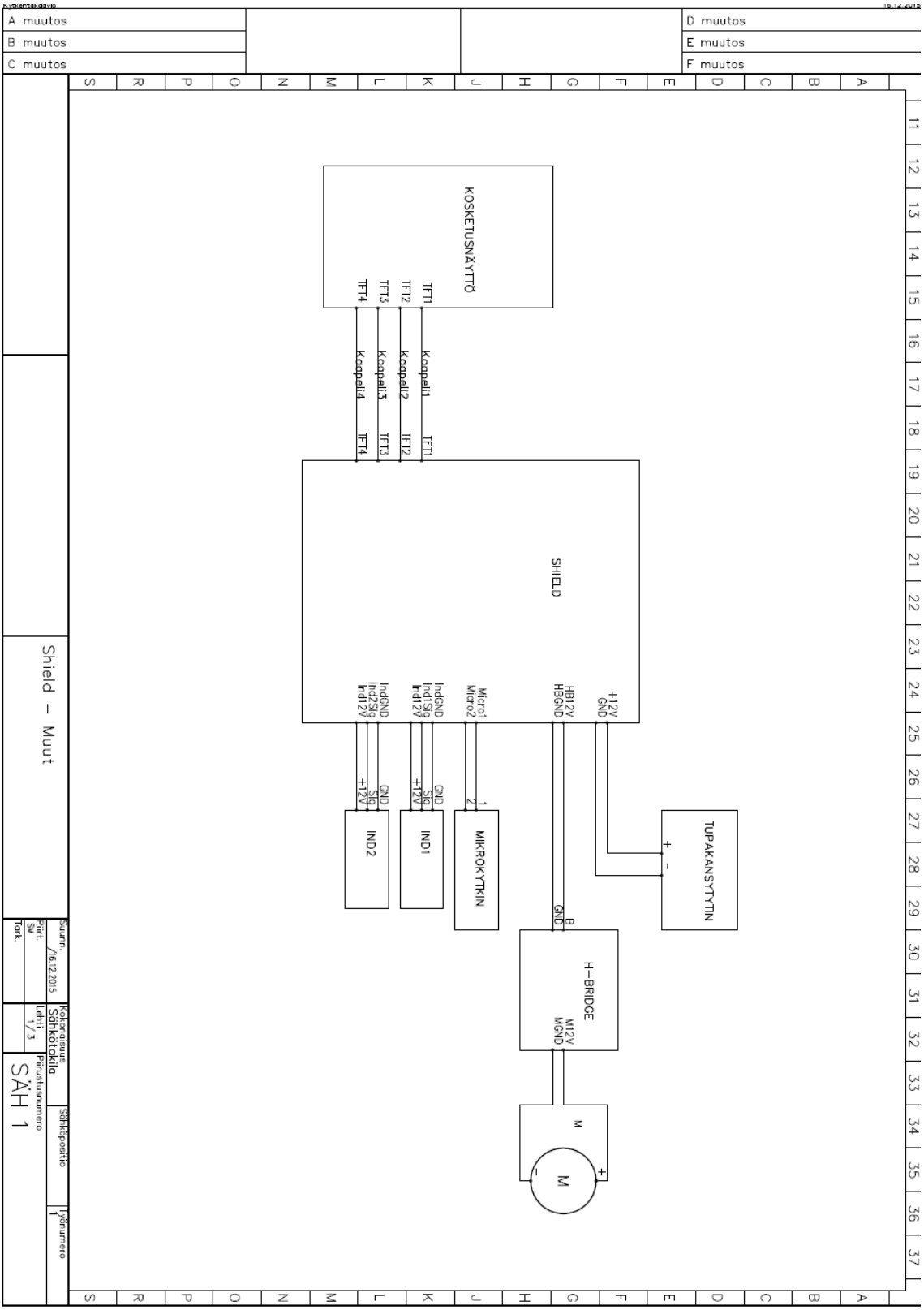
1	Näppäin 3	11	Näppäin 3 Arduinolle 53
2	Näppäin 4	12	Näppäin 4 Arduinolle 51
3	Näppäin 1	13	Näppäin 1 Arduinolle 49
4	Näppäin 2	14	Näppäin 2 Arduinolle 47
5	Näppäinpaneelin +5V	15	Mikrokytkin Arduinolle 45
6	Mikrokytkin +5V	16	Mikrokytkin GND
7	LCD +5V	17	Arduinolta +5V
8	LCD GND	18	LCD GND
9	LCD GND	19	LCD GND
10	Arduinolta GND	20	LCD +5V vastuksella
21	Riviliitin 1 +5V	22	Riviliitin 1 GND
23	Riviliitin 2 GND	24	Riviliitin 2 +5V

Liite 7. Shield-piirilevyn Excel-kaavio

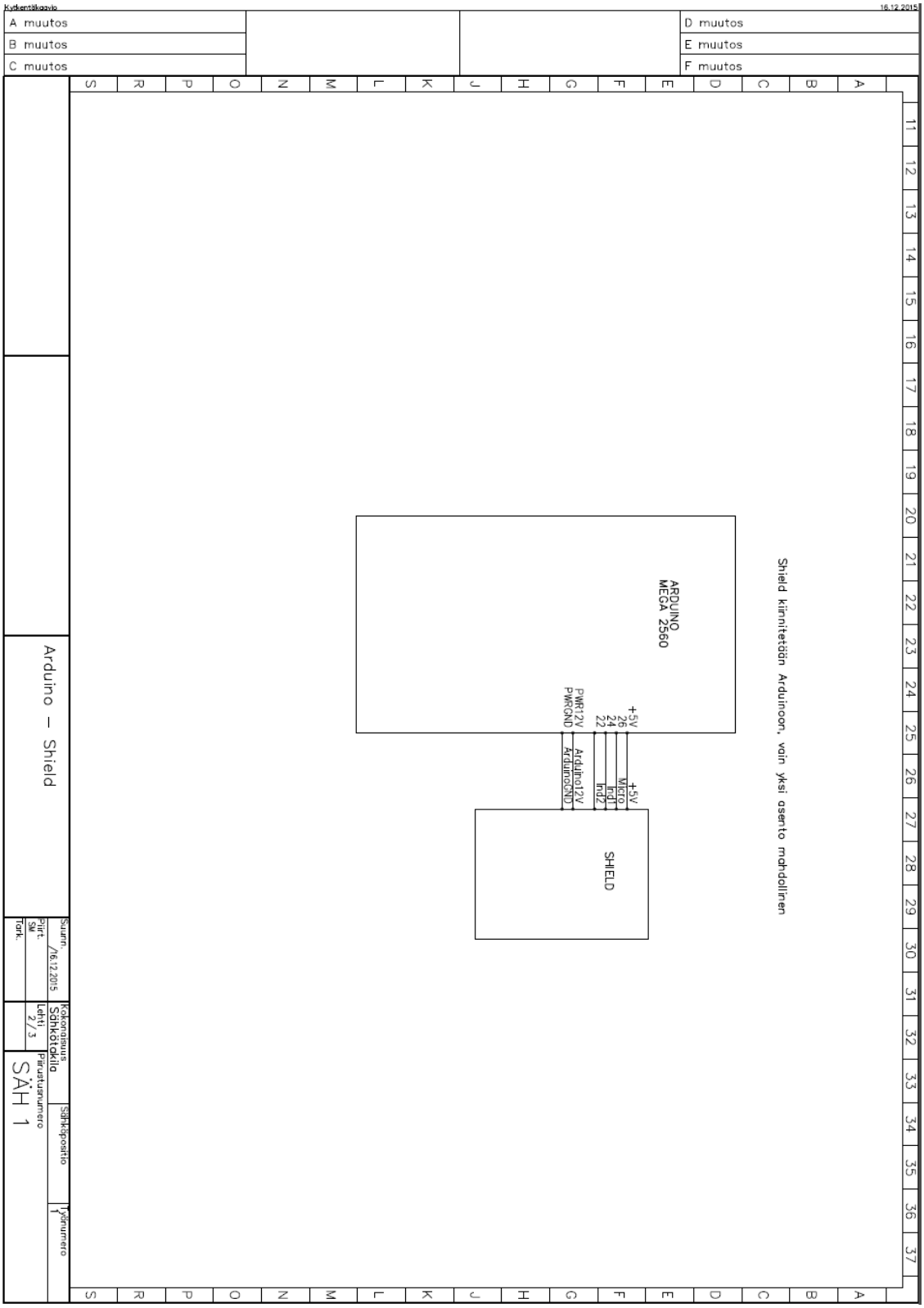
Alla on takilan moottorinohjaimen johdotuksia varten rakennetun shield-piirilevyn periaatekuva. Shield rakennettiin reikäpiirilevylle, ja kuvassa yksi ruutu vastaa yhtä piirilevyn reikää. Tummanharmaat alueet leikattiin pois piirilevystä. Vaaleanharmaat, siniset ja punaiset linjat ovat tinattuja johteita. Sinisessä johteessa kulkee 0 volttia, oikeanpuolimmaisessa punaisessa akulta tuleva 12 voltin jännite ja vasemmanpuolimmaisessa punaisessa Arduinolta tuleva 5 voltin jännite. Keskellä on kaksi (kolmijalkaista) 7805-regulaattoria. Vastuksen arvo on 10 kilo-ohmia. Vihreät alueet ovat riviliittimiä, joissa kussakin on kaksi liitintä. Sivuilla pystyssä olevista harmaista alueista lähtevät johtimet kosketusnäytölle. Alareunan ruuduista lähtee neljä johdinta Arduinolle.



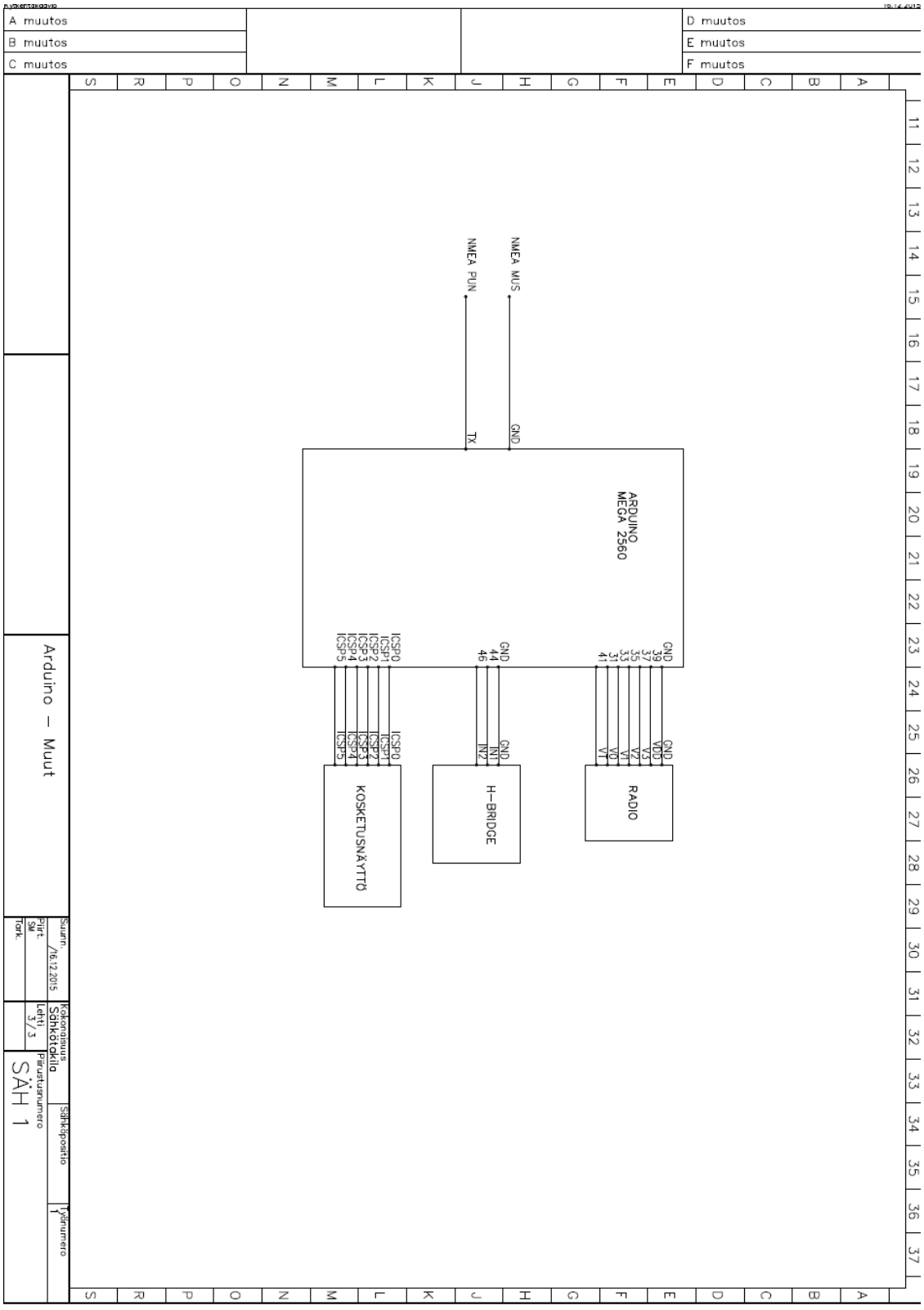
Liite 8. Ohjausjärjestelmän kytkentäkaaviot.



(jatkuu)


Kytkentäkaavio
16.12.2015
A muutos
B muutos
C muutos
D muutos
E muutos
F muutos
Arduino – Shield
Suunn. /16.12.2015
Piirt.
SM
Tark.
Kokoonpanoyksikkö
Sähkötakilla
Lehti
2 / 3
Piirustusnumero
SÄH 1
Sähköposti
Työnumero
1
ARDUINO
MEGA 2560
+5V
26
24
22
PWR12V
PWRGND
SHIELD
+5V
Mcto
Ind1
Ind2
Arduino12V
ArduinoGND
Shield kiinnitetään Arduinoon, vain yksi asento mahdollinen

(jatkuu)


A muutos
B muutos
C muutos
D muutos
E muutos
F muutos
NMEA MUS
NMEA PUN
GND
TX
ARDUINO MEGA 2560
GND 39 37 35 33 31 41
GND 44 46
ICSP0 ICSP1 ICSP2 ICSP3 ICSP4 ICSP5
GND VDD V3 V2 V1 V0 VT
GND IN1 IN2
KOSKETUSNÄYTTÖ
H-BRIDGE
RADIO
Arduino – Muut
SÄH 1