Marko Kauppinen

# 6X6 ROBOTIN PROTOTYYPPI

Tietotekniikan koulutusohjelma

2012

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

# 6X6 ROBOTIN PROTOTYYPPI

Kauppinen, Marko
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Kesäkuu 2012
Ohjaaja: Ylikoski, Mauri
Sivumäärä: 66
Liitteitä: 12

Asiasanat: 6x6, robotti, prototyyppi, etäohjaus, 3G, WLAN

Opinnäytetyön aiheena oli robotin prototyyppi. Tarkoituksena oli rakentaa jatkokehitystä varten runko ja ohjausjärjestelmä. Robotissa on kuusi vetävää rengasta. Runko on mahdollisimman stabiili kameroita varten ja siksi renkaat ovat kiinnitetty edessä teliakselilla ja takana keinuakselilla. Etäohjaus ja ohjelmoitavuus 3G-internet tai WLAN-yhteyden välityksellä ovat robotin olennaisia ominaisuuksia.

Robotin suunnitteluun vaikuttivat vahvasti käytettävissä olevat välineet, budjetti ja resurssit. Runko piti pystyä tekemään itse välineillä mitä oli saatavilla. Projektissa tavoitteena oli löytää edullisimmat vaihtoehdot, miten laite voidaan toteuttaa. Aikataulusyistä johtuen moottorin ohjainpiirissä oli tyydyttävä valmiiseen I/O-piiriin. Moottoriohjaimet on suunniteltu toimimaan myös PIC-kontrollerilla myöhempää testausta varten.

Työn tuloksena valmistui prototyyppirobotti josta on hyvä jatkaa kehittelyä ja tehdä käytännön testaamista. Lisäksi kustannukset pysyivät alhaisina, kuten oli tavoitteena.

6X6 ROBOT PROTOTYPE

Kauppinen, Marko
Satakunta University of Applied Sciences
Information Technology
June 2012
Supervisor: Ylikoski, Mauri
Number of pages: 66
Appendices: 12

Keywords: 6x6, robot, prototype, remote control, 3G, WLAN

The purpose of this thesis was the prototype of a robot. The intention was to build a frame and control system for further development. The robot has six tires which are capable to pull. The frame is as stable as possible for the cameras and that is why the tires are mounted as tandem on the front axle and swinging on the rear axle. Remote control and programmability via 3G Internet or WLAN connection are the essential properties of the robot.

The design of the robot was strongly influenced by the available equipment, budget and resources. The frame has to be done with the tools which were available. The objective of the project was to find the cheapest options by which the device can be implemented. For the reasons of the timetable one has to be satisfied with a ready-made I/O-circuit. The motor controllers are also designed to operate with the PIC-controller for later development.

As the result of this thesis the prototype of a robot was constructed which is a good basis to continue development and practical testing. Also the cost remained low which was the original goal.

# SISÄLLYS

# LYHENNELUETTELO

| | |
|---|---|
| 3G | 3rd generation mobile telecommunications |
| APT | Advanced Packaging Tool |
| ARM | Advanced RISC Machines |
| GPS | Global Positioning System |
| HALL-anturi | Hall effect sensor |
| I2C | Inter-Integrated Circuit |
| I/O | Input/Output |
| LAN | Local Area Network |
| PIC | Peripheral Interface Controller |
| PWM | Pulse-width modulation |
| N-FET | N-channel Field-Effect Transistor |
| RS232 | Recommended Standard 232 |
| TTL | Transistor-transistor logic |
| URL | Uniform Resource Locator |
| USB | Universal Serial Bus |
| WLAN | Wireless Local Area Network |

# 1 JOHDANTO

Opinnäytetyön tavoitteena on valmistaa prototyyppirobotti myöhempää kehitystyötä varten. Työssä on käytettävissä jonkin verran valmiina olevia komponentteja, joiden lisäksi hankitaan puuttuvat ja tehdään runko, moottorinohjaimet sekä ohjelmisto. Ensimmäinen vaihe jota tämä työ käsittelee on saada robotti rakennettua ja toimimaan alkeellisesti etäohjattuna. Toisessa vaiheissa robottia kehitetään lisää, mutta sitä tämä opinnäytetyö ei käsittele. Työssä on kuitenkin otettava huomioon jatkokehittely.

Rungon rakentamista varten käytettävissä on yleistyötila, jossa on hitsauskone, kulmahiomakone, metallisorvi sekä kattava valikoima muita työkaluja ja työkoneita. Elektroniikan rakentamiseen käytettävissä on juotosasema, oskilloskooppi, yleismittareita, kaasupoltin, yms. harrastajan tarvikkeita. Piirilevyt suunnitellaan itse tilaten suunnitelmista tai yksinkertaisemmissa kohteissa kasaten prototyyppilevyille.

# 2 PROTOTYYPPI

## 2.1 Taustat

Prototyyppirobotti on pidempiaikainen projekti, joka kehittyy oppimisen ja havaittujen epäkohtien mukaisesti eteenpäin. Ennen opinnäytetyön aloittamista on projekti alkanut radio-ohjattavista roboteista, joita kutsuin radio-ohjattaviksi autoiksi. Radio-ohjattaviin robotteihin olin hieman lisännyt yksinkertaista robotiikkaa, kuten viiksianturien havainnosta ennalta ohjelmoituja väistöjä.

Aiempi robotti oli yhteistyötä koneistaja opiskelijan kanssa ja yhteistyön päättyessä puuttui robotilta mekaniikka. Samalla projektin suuntaan tuli muutos että robotit eivät ole enään pelkästään radio-ohjattavia. Mittakaava pieneni tuolloin kuusijalkaiseen kävelevään robottiin. Valmiin mekaniikan yhdistin servo-ohjaimeen ja linuxilla pyörivään ARM-korttiin. Pahaksi ongelmaksi muodostui mittakaava. Paino-ongelmat rajoittivat akun kokoa sekä anturien määriä. Ja isompi mittakaava servoilla olisi maksanut aivan liikaa.

## 2.2 Tavoite

Projektilla on kaksiosainen rooli toimien esittelynä, sekä uusien innostavien asioiden testaamisen alustana. Projektin suurimpana haasteena on sen matala budjetti. Kustannusten takia ei useinkaan voida käyttää mieleisintä komponenttia ja tästä syystä joudutaan etsimään vaihtoehtoa.

Opinnäytetyönä tehtävän prototyypin on tarkoitus lopettaa nämä ongelmat ja tarjota uusi alusta projektille kohti autonomisuutta. Mittakaava on tarpeeksi iso, jotta robotilla kykenee kulkemaan maastossa sekä lisäämään ongelmitta toiminnallisuuksia.

# 3 VAATIMUSMÄÄRITTELY

## 3.1 Mekaaniset vaatimukset

Rakenteellisesti robotin pitää olla suhteellisen yksinkertainen ja mahduttava kulkemaan 80cm vapaan leveyden oviaukosta, joka nykyisten rakennusmääräysten mukaan pienin sallittu oviaukko. Robotin on päästävä kohtuullisten esteiden yli ja olla mahdollisimman stabiili alusta kameroille. Tähän päästään yksinkertaisimmin telirakenteella sekä 6x6 vedolla.

Robotissa on oltava työkalun kiinnitys järjestelmä, mihin kytkettävissä työntölevy tai nostin. Työntölevyn tai nostimen on kyettävä nostamaan vähintään 40kg kuorman 20cm korkeudelle. Tämä korkeus riittää asumuksien esteiden ylittämiseen, koska 19cm korkeus on asuinrakennuksen portaiden suurin sallittu yhden askelman nousu ja kynnyksien suurin sallittu korkeus on 20cm.

## 3.2 Toiminnalliset vaatimukset

Robotin etäohjaus yksi tärkeimmistä ominaisuuksista tässä projektin vaiheessa. Robotti on kyettävä täysin uudelleen ohjelmoimaan etänä. Ohjelmointiin käytetään C/C++ -kieltä.

Etäohjaus tapahtuu joko 3G-verkon tai WLAN-verkon välityksellä. Robottia voi ohjata kamerakuvan tai GPS-kordinaattien ja myöhemmin kompassi-anturin avulla. Kahdella kameralla voidaan muodostaa stereokuva. Paikallisesti robottia voidaan ohjata verkkokaapelilla tai bluetooth-yhteydellä. Paikallisesti ohjaukseen käytetään myös Zeemote JS1 -ohjainta tai muuta ohjainta.

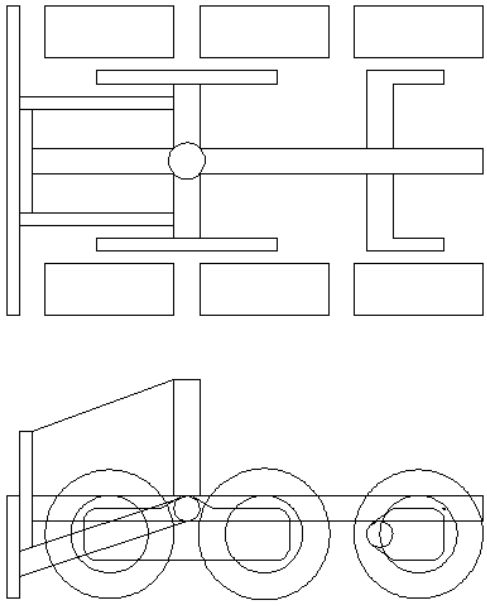
Robottiin pyritään myös lisäämään vähintään englanninkielinen puhe ja äänien toiston mahdollisuus, jotta voidaan antaa ihmisille palautetta. Myöhemmässä vaiheessa robottiin lisätään moottoreihin HALL-anturit autonomista toimintaa varten. Toiminta-ajat on robotilla on oltava yli 30 minuuttia. Tähän pyritään pääsemään akustolla. Pienen aggregaatin mahdollisuus myös otetaan huomioon.

# 4 LAITTEISTON TOTEUTUS

## 4.1 Mekaaninen toteutus

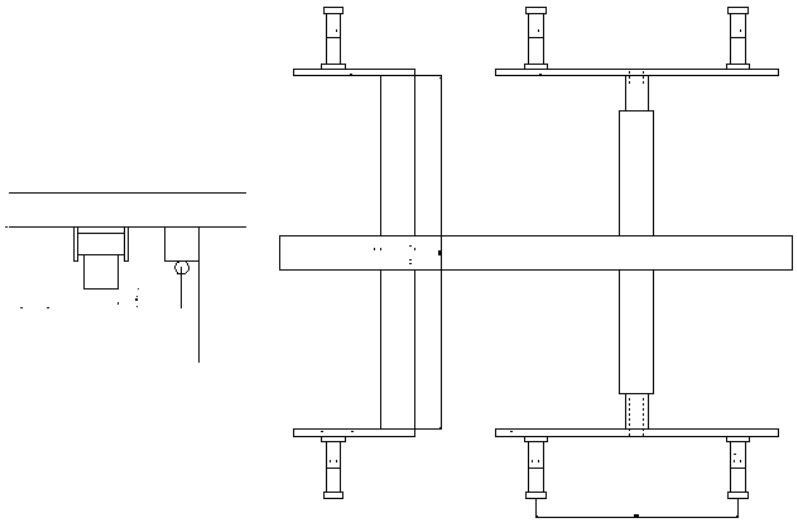
Yksinkertaiselta tavalta toteutta runko, vaikutti ristinmallinen runko telillä. Tällainen runkoratkaisu on vaatimusten mukaisesti stabiili alusta kameroille. Suunnitteluun käytettiin QCad-ohjelmistoa, joka suunnittelun aikana muuttui LibreCAD nimiseksi.

Karkeassa suunnitelmassa (kuva 1.) aluksi hahmoteltiin rungon ulkonäköä ja toiminnallisuutta. Moottorien ollessa pyörätuolien moottoreita voitiin renkaat kiinnittää suoraan moottorinakseleihin. Suunnitelmissa alussa oli työntölevy, joka myöhemmässä vaiheessa muuttui trukkinostimeksi.



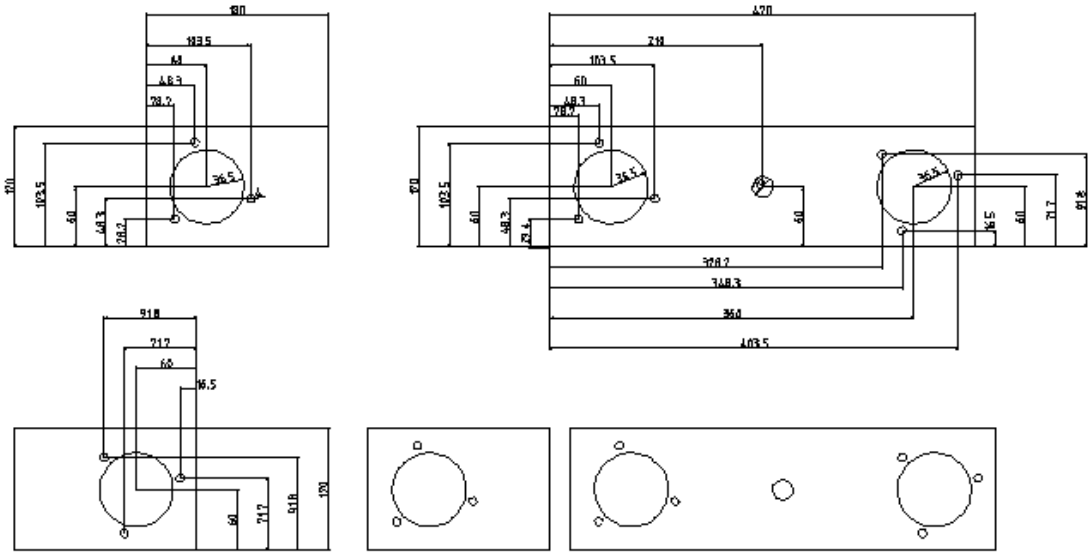
*Kuva 1. Alustava rungon suunnitelma*

Suunnitelma eteni alustavasta suunnitelmista yksityiskohtia kohti. Akseliston rakennetta hahmoteltaessa saatiin tarkempia mittoja (kuva 2.) ja päästiin suunnittelemaan telinakselin rakenteita. Teliakselin heiluriin hitsataan sorvatut liukuholkit joissa rasvanipat. Keinut lukitaan akseliin sokkatapeilla. Taka-akseliin hitsataan myös sorvattu liukuholkki jossa rasvanipat. Runkoon hitsataan korvakkeet ja taka-akseli kiinnitetään liukutapilla. Renkaiden kiinnitys moottoreihin suunniteltiin samalla. Renkaiden reikää piti hieman porata isommaksi sekä taltalla muotoilla kiila-akselin kiilan kolo. Moottoreiden akseleiden pituus ei kuitenkaan riittänyt suoraan kiinnitykseen pelkällä akseleiden päissä olevalla ruuvilla. Joten moottoreiden akseleiden jatkoksi sorvattiin holkit, jotka kiinnitettiin akselin päihin pultilla.



*Kuva 2. Telirungon yksityiskohtia*

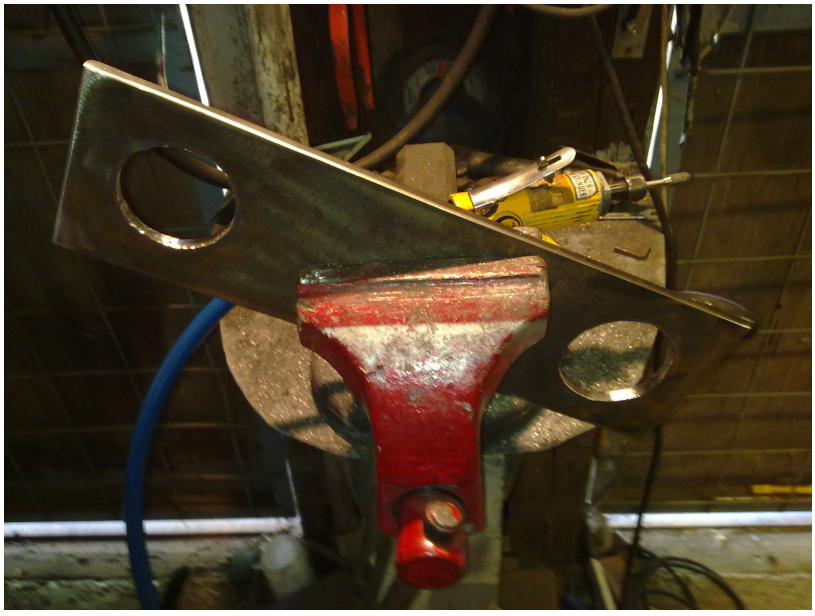
Telin heiluri ja taka-akselin moottorien kiinnitykset (kuva 3.) myös mittasuunniteltiin tietokoneella etukäteen. Moottoreita kallistettiin hieman toisiaan kohti ettei ne telin kääntyessä ottaisi kiinni taka-akseliin tai etuosan nostolaitteisiin. Myös taka-akselin moottoreita kallistettiin, jotta akselin kallistuessa menisivät molemman puolen moottorit samaan aukkoon. Moottorit kun eivät ole symmetrisiä kulmavaihteeltaan. Mittasuunnitelma tulostettiin paperille luonnollisessa koossa jolloin reikien paikat oli näin helppo merkitä tarkasti metalliin paperin läpi pistepuikolla.



*Kuva 3. Moottoreiden kiinnityksien mitoitus*

Aivan kokonaan tietokoneella ei voitu tehdä suunnitelmia, koska osa tarvikkeista oli vasta tulossa eikä kaikkien tarvikkeiden saatavuudesta ollut täyttä varmuutta. Akkutelineet ja nostolaitteen kiinnitykset oli tällaisia joiden suunnittelu oli pakko jättää myöhemmäksi.

Työn tekeminen alkoi telin heilurin rakentamisesta (kuva 4.) Reiät leikattiin karkeasti auki ja hiottiin karahiomakoneella oikeisiin mittoihin. Metalli on 10mm vahvuista levyä, joten hiominen vei aikaa.

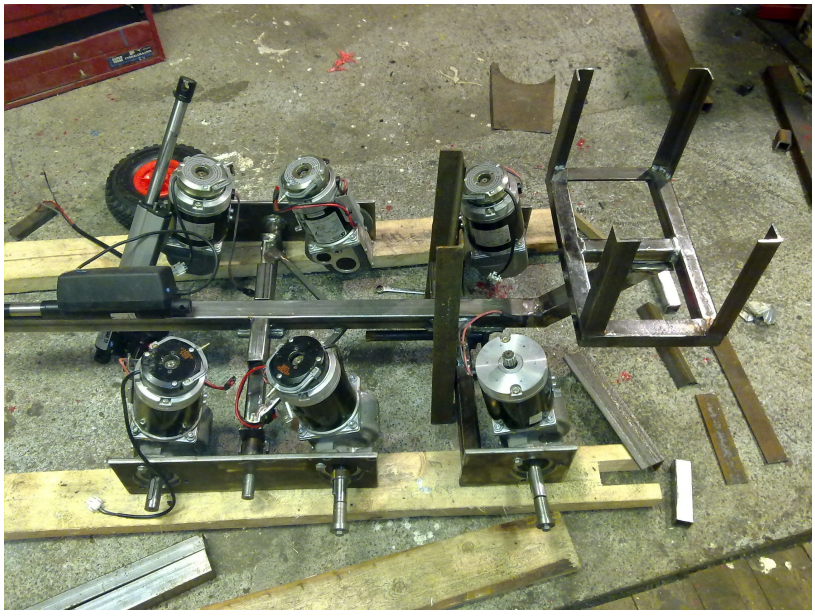


*Kuva 4. Telin heilurin moottorien kiinnityksen reikien teko*

Muut reiät saatiin porattua suoraan pylväsporakoneella ja päästiin moottorien sovittamiseen heiluriin (kuva 5.). Samoihin aikoihin saatiin myös sorvattua akselien päiden renkaan kiinnityksen holkit sekä sovitteet.

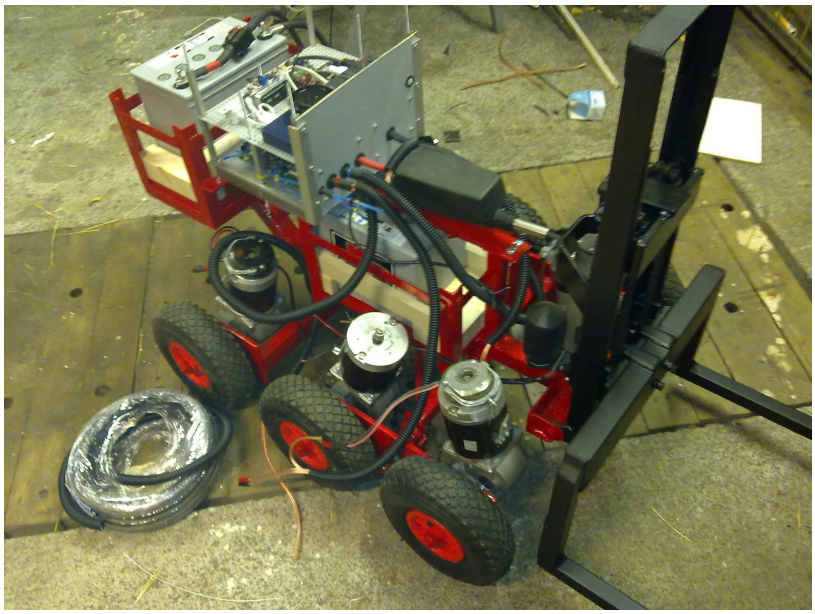
*Kuva 5. Moottorien sovitusta kiinnityksiin*

Runkoa pääsi lisää hahmottelemaan sitä mukaan kun tarvikkeita saapui. Nostimien ja akkutelineiden sopivuus osumatta renkaisiin tai moottoreihin piti tarkistaa kokeilemalla (kuva 6.). Samalla miettiä mistä johdotukset kulkee parhaiten.



*Kuva 6. Rungon alusta kasattuna ja akkutelineiden hahmottelua*

Akkutelineistä tuli lopulta turhan suuria kun käyttöön tuli ulkomitoiltaan pienemmät akut mitä telineitä tehdessä oli ajateltu. Loppuvaiheita työssä oli johtojen asennusta vetäen ne suojaputkeen (kuva 7.).



*Kuva 7. Runko loppusuoralla*

## 4.2 Sähköinen toteutus

### 4.2.1 Akusto ja virransyöttö

Akuston vaatimuksena oli 30 minuutin toiminnallinen aika. Tästä hyvin karkeasti pahin mahdollinen sallittu kuormitus laskemalla kuuden ajomoottorin yhteistehoista saaden 2100W, joka kerrotaan 0.5 tunnilla. Eli energian tarve pahimmillaan on 1050Wh. Akuston ollessa 24V tarkoitti tämä akkujen vaatimuksena noin 44Ah kapasiteettia 88A purkuvirralla.

Akuissa lopulta päädyttiin MK Power 8G24FT 73Ah/63Ah akkuihin. Rakenteeltaan ne ovat geeliakkuja, joten niiden kallistelusta ei ole haittaa. Kapasiteetti on määritelty olevan 73Ah 0.73A purkuvirralla ja 48.5Ah 73A purkuvirralla, jotka riittävän lähellä pahinta mahdollista tilannetta.

Akusto ja sen johdotus suojattiin kahdella resetoitavalla 100A johdonsuojakytkimellä. Johdonsuojakytkimet asennettiin molempien akkujen positiiviselle navalle suojaamaan siten molempia akkuja oikosuluilta. Alkuperäisenä suunnitelmana oli kytkeä johdonsuojakytkimet erikseen oikean ja vasemman puolen moottoreille, mutta todettiin että 100A riittää maksimiksi virrankulutukseksi. Virransyötön johdotuksena käytettiin akkujen välillä sekä maajohtona 50mm² johdinta. Virransyöttönä moottoriohjaimille käytettiin 25mm² johdinta.

Ohjauselektroniikkaa varten on 10A sulake moottoriohjainten yhteydessä. Sen taakse on kytketty kameroiden 12V 3A, kytkimen 6V 3A, USB-hubin 5V 3A sekä NSLU2 varten 5V 3A hakkurivirtalähteet.

### 4.2.2 Moottoriohjain

Moottoreiden ohjaamiseksi aluksi katselin valmiita ohjaimia, mutta niiden hinta oli melkoinen. Projekti olisi pysähtynyt moottoriohjainten puutteeseen. Tästä syystä suunnittelin edullisemman ohjaimen, joka myöhemmässä vaiheessa korvataan paremmin suunniteltuihin. Ohjaimista puuttuu esimerkiksi hyvinkin tarpeellinen virranrajoitus. Myös ohjauspiirinä on myöhemmässä versiossa tarkoituksena käyttää PIC16F690-kontrolleria /1/, jonka ohjelmiston testaaminen otettiin suunnitellussa ohjaimessa huomioon. Piirin vaihto tapahtuu muuttamalla jumpperien asentoa.

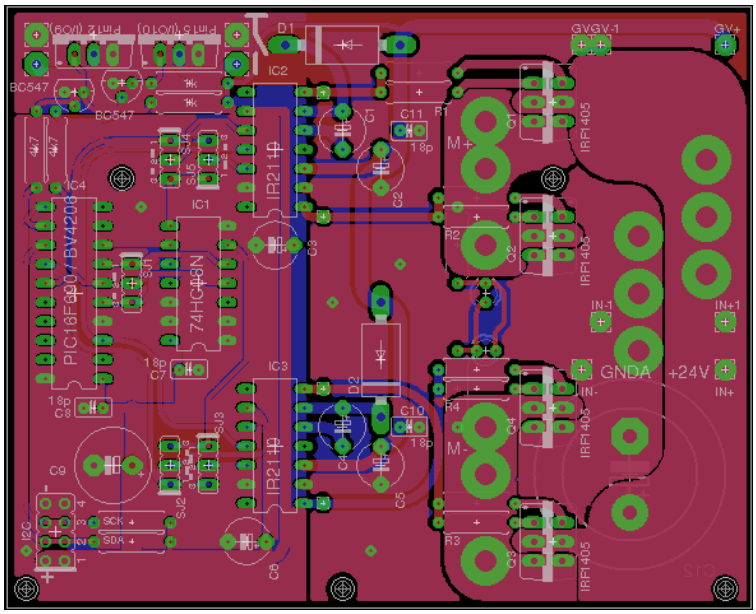
Ohjaimet tyypillinen H-silta rakenne /2/. Nopeudensäätö tapahtuu ylempien IRF1405 N-fettien /3/ pulssimoduloidulla signaalilla. Fettien ohjaukseen käytetään IR2110-fettiohjainta /4/. N-fettien tarvitsema 12V ohjaujännite tehdään erillisellä hakkurivirtalähteellä. Ohjaimen rajoituksena on sen ylempien N-fettien avausjänniteen lataaminen kondensaattoriin ylempien N-fettien toiminnasta. Ylemmät N-fetit ei tästä johtuen voi olla eivätkä pysy kokoaika päällä ilman moduloitua signaalia. Joten ihan täyttä 100% ajoa ei voida tehdä. Niin lähelle täyttä ajoa kuitenkin päästään, ettei nähty tarpeelliseksi tehdä erillistä piiriä. Joka joko varaisi kondensaattoria tai tuottaisi 12 V moottoreiden käyttöjännitettä suuremman jännitteen.

Fettiohjaimia ohjaa BV4206-piiri joka I2C-I/O piiri /5/. Piiri on jalka yhteensopiva PIC16F690-kontrollerin kanssa. BV4206 ei sisällä kuin yhden pulssimoduloidun lähdön, joten sen jakamiseen oikealle fettiohjaimelle ohjaimessa on 74HC08-logiikkapiiri /6/. Käytettäessä itse ohjelmoitua PIC16F690-kontrolleria logiikkapiiri ei ole käytössä vaan käytetään mikrokontrollerin datalehden mukaista H-sillan kytkentää. Poikkeuksena että ylempien ja alempien N-fettien paikkaa on vaihdettu. Tämä aikasemmin mainitusta fettiohjaimen ylempien N-fettien auki pitämisen tarvittavan jännitteen latauksen rajoitteen takia.

Suunnitteluun käytettiin Eagle -ohjelmaa. Vaihtoehtona pohdinnassa oli myös ollut KiCAD, mutta Eagle oli ennestään tutumpi. Kummassakaan vaihtoehdossa ei ollut kaikkia komponentteja suoraan saatavilla, joten niitä joutui itsekin tekemään.

Ensimmäiseksi tehdään piirikaavio (LIITE 1). Piirikaavion suunnittelussa Eagle:ssa on joitain ikäviä piirteitä. Kuten se että käytettävä komponentti ja sen kotelointi pitäisi valita jo siinä vaiheessa. Useasti Eaglen kanssa joutuu palaamaan takaisin piirikaavioon ja vaihtamaan komponenttia pelkästään syystä ettei komponenttia saanut juuri kyseisellä kotelolla.

Piirikaavion jälkeen suunnitellaan piirilevy. Eagle:n ilmaisissa ei kaupalliseen käyttöön olevassa versiossa on ikävä piirilevyn kokoa rajoittava rajoitus. Piirilevystä suunniteltiin kaksipuoleinen ja samalla suurin mitä Eagle:n ilmainen versio salli (kuva 8.). Virransyöttö pyrittiin pitämään mahdollisimman lyhyinä ja leveinä. Elektroniikan ja moottorien maatasot pidettiin myös erillään.



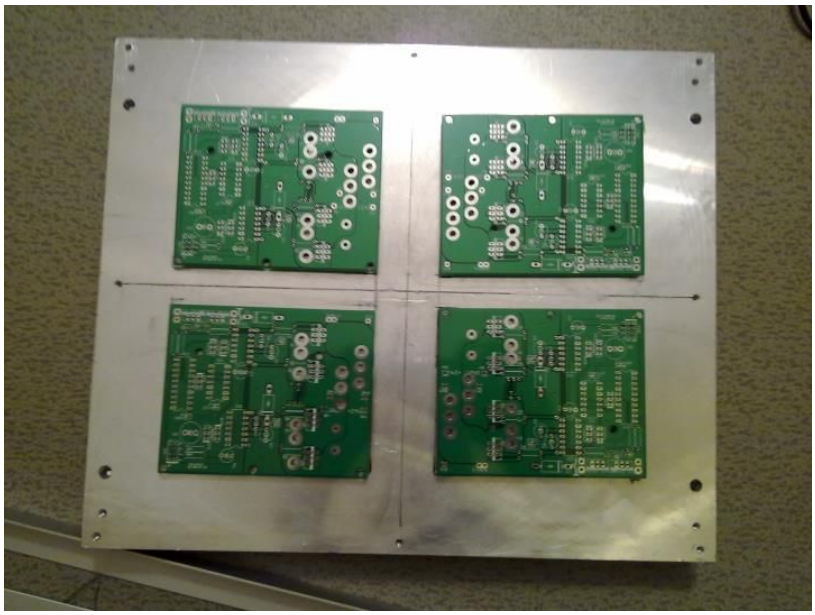

*Kuva 8. Moottoriohjaimen piirikortti*

Piirilevyn suunnitelmat kun oli tehty valmisteltiin ne ITead Studio:n /7/ haluamaan muotoon. ITead Studio on tehnyt asian helpoksi ja heiltä piirilevyn tilauksen yhteydessä löytyy Eagle:n DRC- ja CAM-tiedostot joista jälkimmäinen tuottaa

tarvittavat Gerber-tiedosto. Tiedostot lähetetään sähköpostilla valmistettavaksi ja myöhemmin ne tulee postitse valmiina.

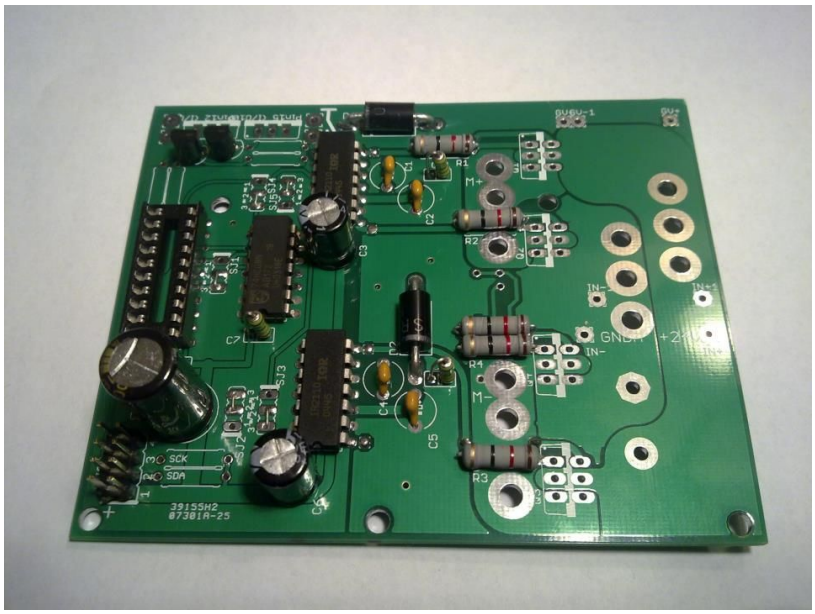
Piirilevyjen saapuessa havaittiin kaksi suunnitteluvirhettä. Pahempi vika oli kytkentävirhe, jonka pystyi korjaamaan hyppyjohdolla. Ennen kuin kytkentävirhe havaittiin, se ehti tuhoamaan yhden fettiohjaimen. Valitettavasti kyseisistä komponenteista oli pulaa jo muutenkin eikä edullisen toimittajan lähemmäs kuukauden toimitusaika asiaa helpottanut. Toinen vika oli myös oma suunnittelussa tapahtunut kuparoinnin vahventamiseksi tarkoitettujen päälle tinattavien kiskojen merkinnän unohtuminen tilatusta levystä. Tästä johtuen vihreä suojapinta oli kokonaan virransyötön päällä ja tulevien kiskojen lisääminen oli hyvin hankalaa.

Ohjaimen N-fetit tarvitsi jäähdytystä. Tähän tarkoitukseen oli varattu massiivinen jäähdytyslevy (kuva 9.), johon kaikki ohjaimet kiinnitettiin. Jäähdytyslevy on paljon suurempi kuin on tarpeen toimien samalla kiinnityspisteenä ja maapotentiaalina. Samalla jäähdytyslevylle mahtuu kaikki moottoriohjaimet kerralla. Kaksi ohjainta tarvitaan ajomoottoreille ja toiset kaksi on nosto ja kallistus.



*Kuva 9. Ohjaimien sijoitus jäähdytyslevylle*

Moottoriohjaimet kun saatiin kasattua (kuva 10.). N-fetit kiinnitettiin lämpöäjohtavan eristelevyn ja kiinnitysruuvien kanssa jäähdytyslevyyn. Jäähdytyslevyyn asennettiin seuraavaksi muoviset ruuvitornit sekä maatasoihin metalliset ruuvitornit, joiden päälle moottoriohjain asennettiin ja N-fetit juotettiin moottoriohjaimeen kiinni.




*Kuva 10. Moottoriohjain kasattuna*

### 4.2.3 Ohjausjärjestelmä

Laitteiston pohjana toimii ARM Intel XScale IXP420. Tämä on otettu Linksys NSLU2 verkkolevypalvelimesta /8/. Käyttöjärjestelmäksi asennettiin Debian Linux muistitikulle /9/. Ratkaisuun päädyttiin siitä syystä että näitä laitteita on itselle useampia jäänyt tämän kaltaista käyttöä varten. Sekä tilalle voi vaihtaa pienin toimenpitein tehokkaampiakin linux käyttöjärjestelmän kehityskortteja.

Oletusasennus ei sisällä GNU C ja C++ kääntäjää eikä USB-laitteiden käsittelyyn tarvittavia kehityskirjastoja, joten ne pitää asentaa käyttöjärjestelmän asennuksen jälkeen. Debianin käyttämällä APT-paketinhallintatyökalulla /10/ asennettavat ohjelmat ovat gcc, cpp, g++, libusb-dev ja make. Kyseiset asentaa ainakin seuraavat ohjelmat ja kirjastot: gcc, gcc-symlinks, libc6-dev, binutils, libusb-dev, make, cpp-

symlinks, g++, g++-symlinks, glibc-extra-nss, libthread-db1, linux-libc-headers-dev, binutils. Hyödyllistä on myös asentaa ntp ja ntpdate ajan synkronoimista varten.

Toista massiivista jäähdytyslevyä oli tarkoitus käyttää ohjausjärjestelmän kiinnittämiseen (kuva 11.). Tästä ajatuksesta kuitenkin luovuttiin laittaen metalliselle reikälevylle johon pääsee suoraan yläpuolelta käsiksi.

*Kuva 11. Ohjausjärjestelmän komponentteja*

Laitteiden ja anturien liittämiseen käytetään pääasiassa USB- ja I2C-väylää. Lisäksi robotissa on LAN-verkko ja sitä varten 8-porttinen LAN-kytkin. Sitä käyttävät IP-kamerat ja varasuunnitelmana voidaan tähän myös internet kytkeä 3G-reitittimellä.

I2C on toteutettu vaihtoehtoisesti käyttäen NSLU2-kortin sisäistä I2C-väylää, joka löytyy piirilevyltä X1205 tai M41T11 -reaaliaikakellopiirin jaloilta 5 ja 6. Tai käyttäen erillistä USB-I2C -liitäntämodulia /11/. Liitäntämoduli näkyy sarjaporttina liittyen ensimmäiseksi vapaaksi USB-sarjaportiksi. I2C-väylään on kytkettynä moottoriohjaimet sekä kompassi-anturi /12/.

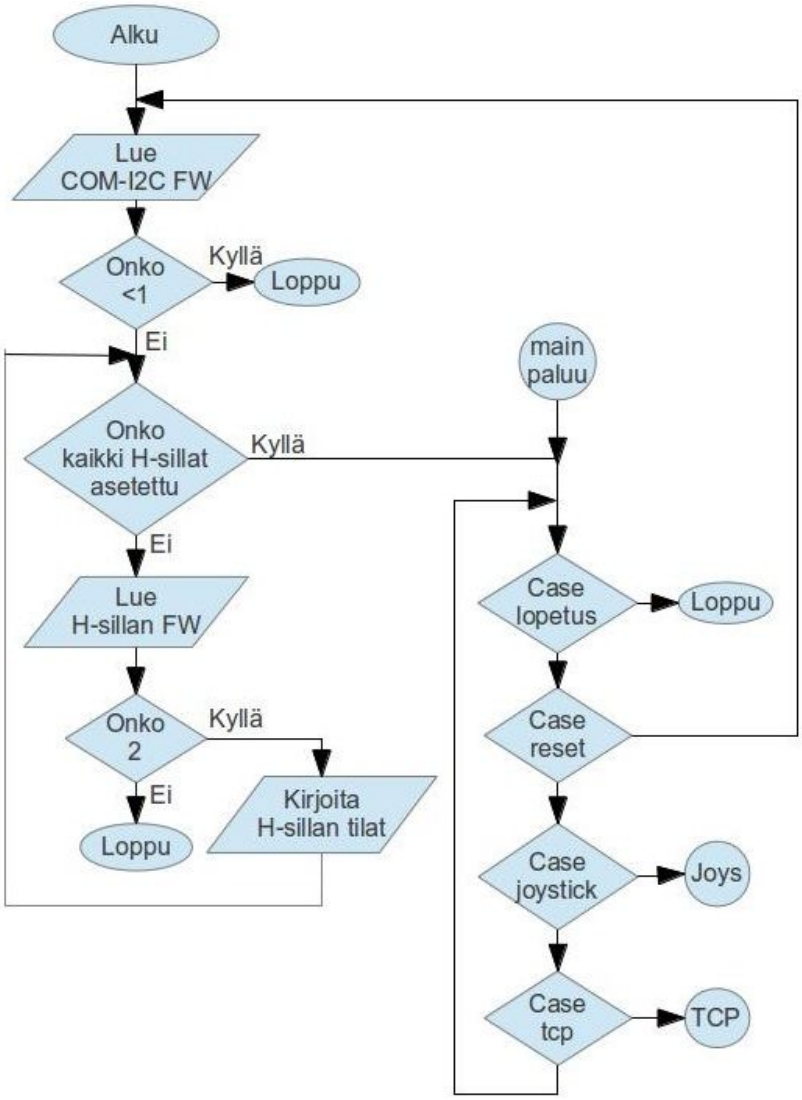
Ohjauskoneen USB-liitäntään on kytketty USB-hub, johon kaikki USB-laitteet on kytketty paitsi massamuistin asemassa oleva muistitikku. 3G internet-yhteyttä varten ohjausjärjestelmässä on Huawei E270 /13/. Laite liittyy kytkettäessä USB-liitäntään

ensimmäiseksi vapaaksi USB-sarjaportiksi. Sijaintietoja varten käytetään ND-100S GPS-tikkua /14/, jossa SiRF-III -piiri. Tikku on APT-paketinhallintatyökalulla löytyvän gpsd-palvelimen kanssa yhteensopiva liittyen myös ensimmäiseksi vapaaksi USB-sarjaportiksi. Useampien USB-sarjaporttina näkyvien laitteiden ja niiden järjestyksen mahdollisen vaihtumisen takia tehtiin laitteille symboliset linkit: /dev/USBi2c, /dev/3G ja /dev/GPS.

Bluetooth on toteutettu Super Mini Bluetooth 2.0 Adabter Dongle /15/ nimisillä sovittimilla. Sovitinta varten asennettiin APT-paketinhallintatyökalulla: bluez-utils ja bluez-hcidump. Päätteen bluetooth on tästä poiketen toteutettu DealExtreme:stä hankittu RS232 TTL-bluetooth -modulilla /16/, joka kiinnitetty ohjaustietokoneen piirikortilla olevaan sarjaliitäntään. Näin päästään päätteeseen silloinkin kun käynnistyksessä on vikatilanne. Äänikorttina toimii DealExtreme:stä ostettu usb-äänikortti /17/ kytkettynä pieneen vahvistimen sisältävään kaiuttimeen. Rakennetta on sen verran muokattu että usb-äänikortti on asennuttu vahvistimen sisältävän kaiuttimen paristokoteloon ja vahvistin ottaa virtansa USB-liitännästä. IP-kamerat ovat myös DealExtreme:stä hankittuja IP-400 kameroita /18/. Kameroissa on webbipalvelin, josta kuvan saa videona tai kuvina url-osoitteella.

# 5 OHJELMISTO

Opinnäytetyössä käsiteltävä ohjelmisto on hyvin alkeellinen versio päämääränä testata laitteiden keskenään toiminta. Ohjelmassa on paljon header-tiedostoihin kirjoitettua koodia mikä kuuluisi olla c-tiedostoissa. Tässä ohjelman versiossa (kuva 12.) valintoina on vain uudelleen moottoriohjaimen alustus, joystick-ohjaus tai logo-komentojen kaltaisilla komennoilla verkkoyhteyden kautta ohjaus.



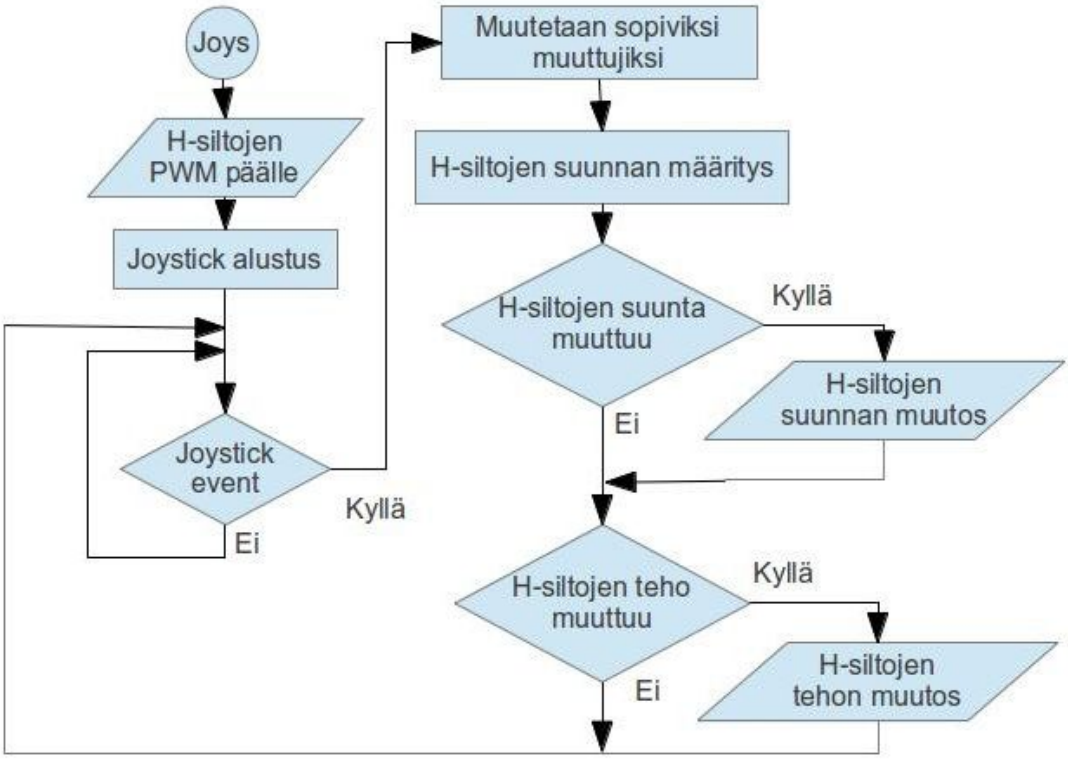

*Kuva 12. Pääohjelman karkea lohkokaavio*

Käynnistyessä pääohjelma (LIITE 2) testaa ja alustaa alustuksen ohjelmakoodissa (LIITE 3) vuorotellen löytyykö moottoriohjaimen ohjauspiiri. Jonka jälkeen alustaa moottoriohjaimen tilat kohdalleen. Ensimmäiseksi testataan I2C-sovittimen sarjaportin olemassa olo ja sen ohjelmiston versio jos käytetään ulkoista sovitinta. Tämän jälkeen kunkin yksittäisen moottoriohjaimen testaaminen aloitetaan lukemalla sen ohjelmistoversio. Oikean ohjelmaversion jälkeen asetetaan moottoriohjaimen tiloja. Aluksi asetetaan shutdown:in I/O-portti output-tilaan ja määritellään tilaksi 0. I/O-portti on määritelty addrreg.h -tiedostossa (LIITE 4) kuten muutkin moottoriohjaimen I/O-portit. Toiminnallinen tarkoitus tällä on sammuttaa fettiohjain. Seuraavaksi asetetaan PWM-signaali ja N-fetit pois päältä samalla tavalla määrittäen I/O-portin suunnan ja antamalla I/O-portin tilaksi 0. Lopuksi määritetään shutdown-tila pois päältä vaihtamalla tilaksi 1.

Suoraan kuitenkaan alustusvaihe ei näitä ohjauksia tee ohjelmakoodissaan vaan siihen tarvitaan sarjaportin ohjelmakoodia (LIITE 5) ja I2C-sovittimen ohjelmakoodia. I2C-sovittimen ohjelmakoodi riippuu siitä käytetäänkö NSLU2-kortin sisäistä jolloin ohjelmakoodi on i2clib.h vaiko ulkoista sovitinta ohjelmakoodin ollen silloin i2crw.h (LIITE 6). Lisäksi tarvitaan vielä moottoriohjaimen käskyjen ohjelmakoodi bv4206.h (LIITE 7). I2C-sovittimen ohjelmakoodi muodostaa komennot, jotka lähetetään sarjaportin välityksellä ohjaimille. Moottoriohjaimen ohjelmakoodin tarkoitus on helpottaa ja selkeyttää ohjelmakoodia siten ettei koodiin tarvitse komentojen arvoja muistaa vaan voi suoraan nimeltä komennon kirjoittaa. Onnistuneen läpi käynnin jälkeen edetään joko valikkoon tai komentoriviltä käynnistyksen yhteydessä määriteltyyn toimintaan. Virhetilanteissa poistutaan heti ohjelmasta antaen virheilmoitus virhekohdasta.

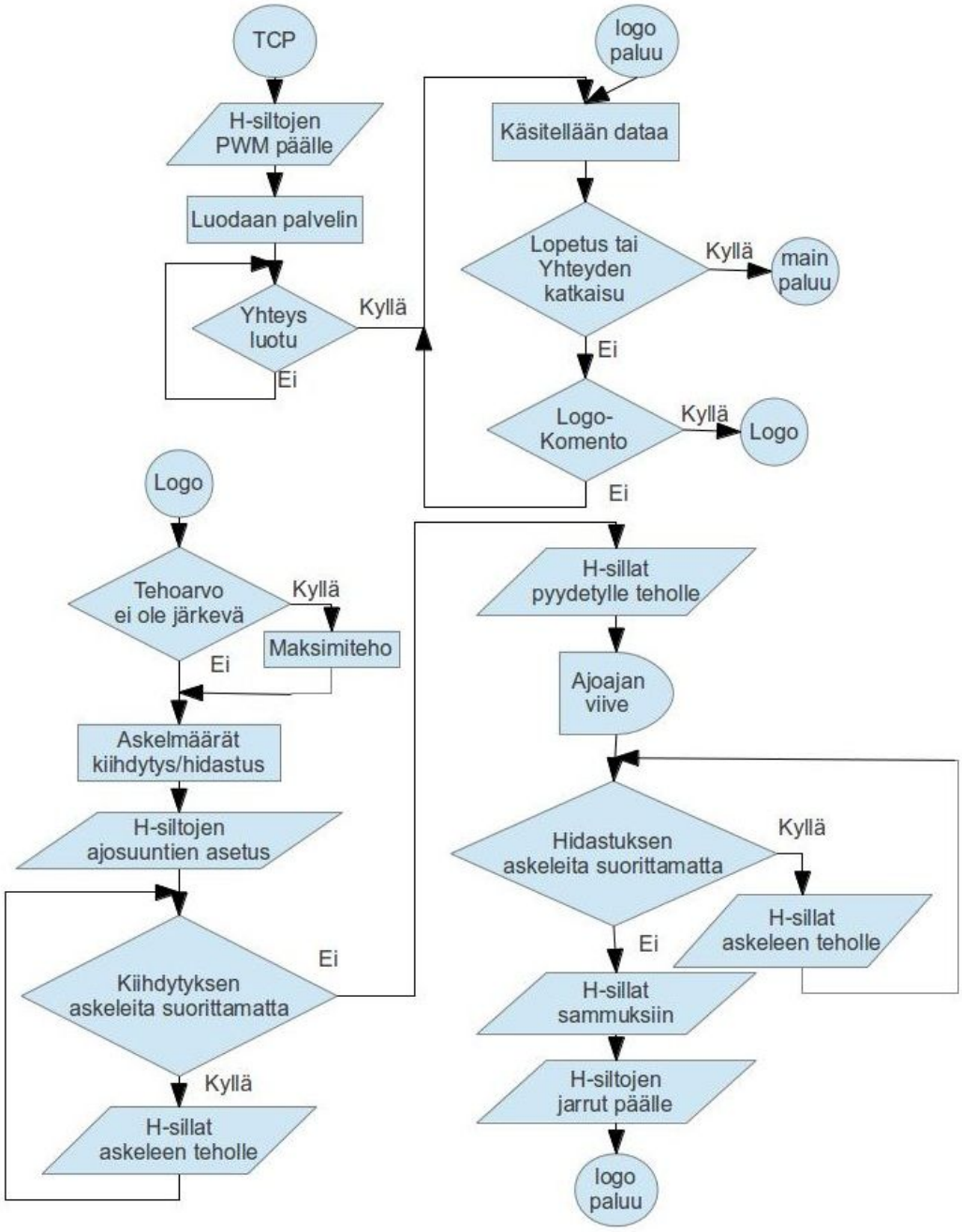
Joystick-ohjaus vaati laitteen /dev/input/js0 löytymisen. Zeemote JS1 -ohjain /19/ toimii linuxissa Maemo:n Zeemoted ajurilla. Ajurin kääntäminen vaati asennettavaksi APT-paketinhallintatyökalulla libbluetooth-dev -paketin. Joystick-ohjelma (kuva 13.) käy läpi joystick tapahtumia silmukkana, josta ohjaimen asennosta riippuen ohjaa H-siltaa. Ohjelmakoodina käytetään joystick.h (LIITE 8) ja valmista muokattua ohjelmakoodia joystickh.h (LIITE 9). H-sillan ohjaamisen

käytetään ohjelmakoodia hbridge.h (LIITE 10) sekä aiemmin mainitut sarjaportin-, I2C- ja moottoriohjaimen ohjelmakoodit.




*Kuva 13. Joystick-ohjelman karkea lohkokaavio*

Logo-komennoilla /20/ ohjattaessa verkkoyhteyden kautta, käynnistyy TCP-palvelin (kuva 14.). Palvelin kuuntelee addreg.h -tiedostossa määritettyä porttia. Oletuksena portti on 3333. Asiakkaan ottaessa yhteyttä porttiin, ohjelma vertaa onko asiakkaalta tuleva data jokin tunnetuista komennoista. Komentojen luoteen takia ohjelma laskee sopivat kiihdytys ja hidastusarvot komennolle. Palvelimen ohjelmakoodi on tcpserver.h (LIITE 11) ja logo-komentojen logo.h (LIITE 12). Robotin ajaminen tapahtuu logo-komentojen koodissa. Ajokomennon tullessa aluksi lasketaan sopivan kiihdytys/hidastus askelien määrä. Tämän jälkeen asetetaan H-sillan tilat sopivaksi ja nollataan ei ajettavat ajomoottorit. Toiminta etenee tästä kiihdytykseen jossa tehoa nostetaan aikaisemmin määritellyin askelin siihen saakka kunnes haluttu teho on saavutettu. Tässä tehossa odotellaan ajoajaksi määritelty aika ja kiihdytyksen kaltaisesti hidastetaan. Lopuksi asetetaan H-silta jarrutilaan, jolloin sillan alemmat fetit ovat johtavia.

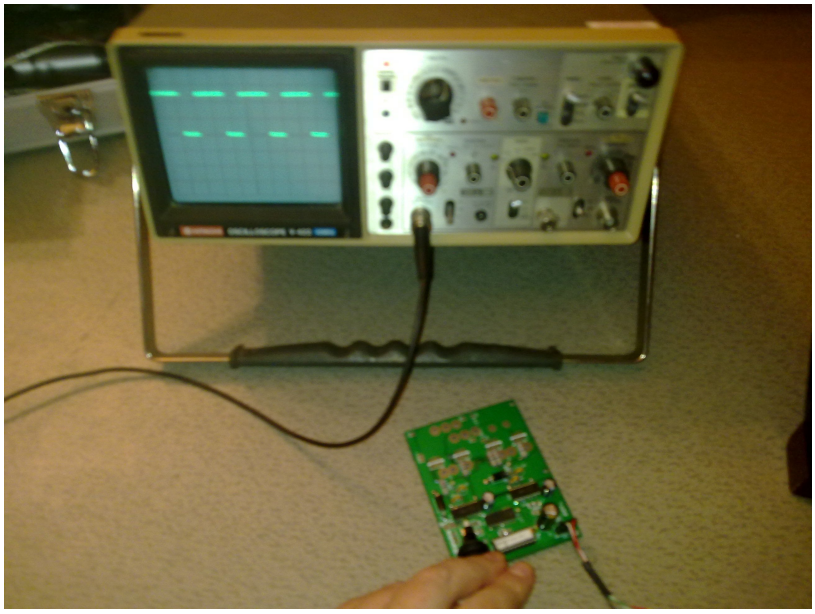



*Kuva 14. TCP-palvelimen ja Logo-ohjelman karkea lohkokaavio*

Puheen tuottamiseen löytyi ohjelma eSpeak /21/. Ohjelmaa on helppo käyttää system() komennolla ja siitä löytyy tuki suomen kielelle. 3G-yhteyksien luominen oli myös yksinkertaista käyttäen wvdial-ohjelmaa. Asetukset on määritelty etc-hakemistossa sijaitsevassa wvdial.conf-tiedostossa.

# 6 TESTAUS

Testaus aloitettiin moottoriohjaimet irrallaan ilman N-fettejä oikosulkujen välttämiseksi. Moottoriohjain alustettiin ja käytiin oskilloskoopilla läpi että signaalit on odotetun mukaisia eri moottorien tehoilla ja suunnilla (kuva 15.).

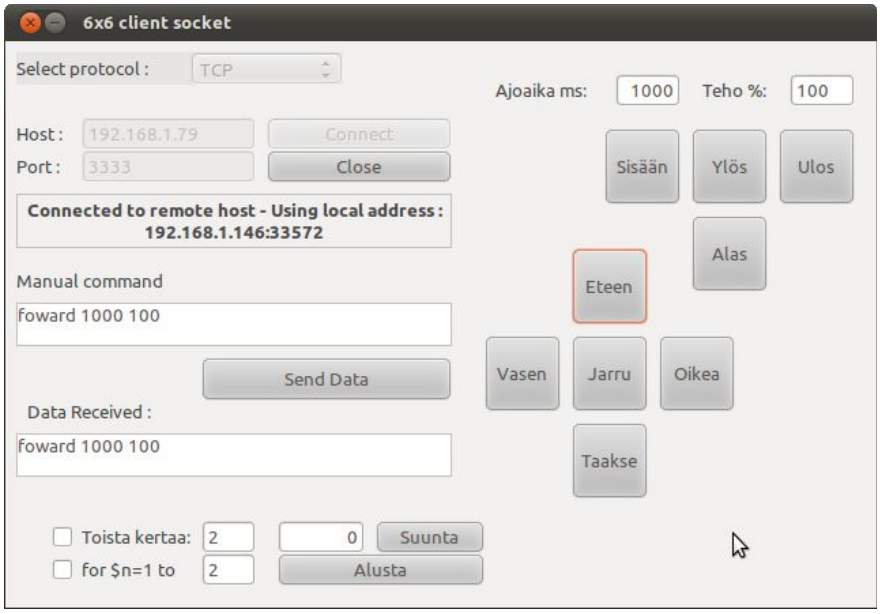


*Kuva 15: Moottoriohjaimen fettien ohjauksen tarkistus*

Kun oli varmistunut että moottoriohjain toimii odotetulla tavalla kytkettiin N-fetit paikalleen ja käyttöjännitteet niille. Yksi suunnittelun virheistä paljastui tässä vaiheessa tuhoten yhden fettiohjaimen. Korjauksen ja uudelleen testaamisen jälkeen moottoriohjain näytti toimivan täysi oikein ja sille kytkettiin isompi kuorma.

Testauksessa kun ei ilmennyt enempää ongelmia kiinnitettiin ohjaimet robottiin. Robotin renkaat nostettiin ilmaan, ja testiä alettiin käymään läpi kytkeytymällä robotin LAN-verkkoon tietokoneella. Tietokonetta varten oli tehty Gambas2 ohjelmalla käyttöliittymä. Gambas /22/ muistuttaa Visual Basic -ohjelmointia ja siinä on valmiina verkkoasiakasohjelman esimerkki johon tarvitsi vain vähän lisäillä tarvittavaa toiminnallisuutta. Tästä johtuen Gambas sopi tarkoitukseen hyvin.

Käyttöliittymässä (kuva 16.) on napit joita painamalla se lähettää tekstikomennot robotin palvelimeen. Lisäksi käyttöliittymässä on mahdollisuus antaa käsin kirjoittamalla komentoja. Robotti kaiuttaa saamansa komennot takaisin virhetilanteiden etsimiseksi ja komentojen perille menon varmistukseksi.

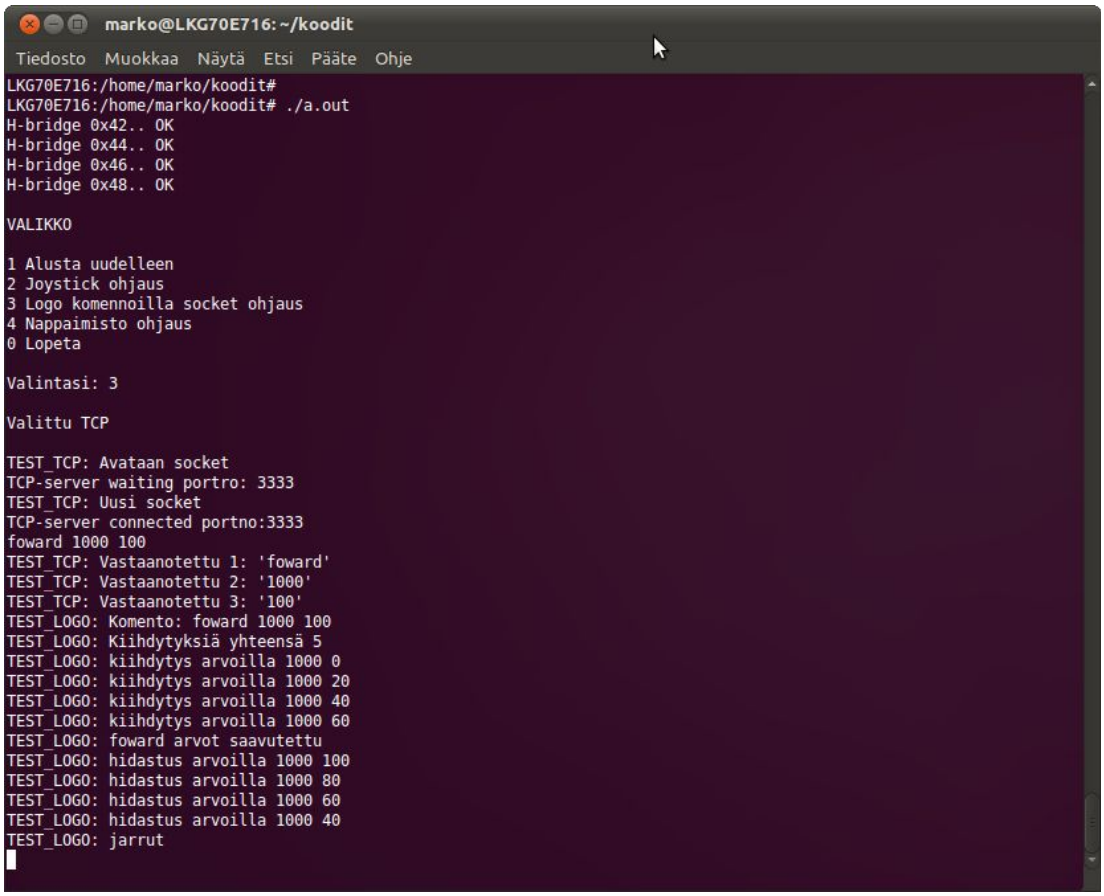



*Kuva 16. Käyttöliittymä testausta varten*

Robotin konsolissa näkyy ohjelmaa suorittaessa mitä toimintoa robotti on tekemässä (kuva 17.). Ilmoitusten määrää pääsi säätämä addrreg.h -tiedostosta tarpeen mukaan ja kääntämällä ohjelma uudestaan.

Kaikki moottoriohjaimet yhdistettynä I2C-väylään ilmeni paljon I/O-virheitä. Välillä toiminnallisuus katosi täysin. Väylän häiriönsietoa koitettiin lisätä lisäämällä ylimääräiset ylösvetovastukset. Häiriöt vähenivät niillä, mutta eivät kadonneet täysin. Vaihtamalla välijohdot paremmin häiriöitä sietäviin, päästiin satunnaisiin häiriöistä. Toinen virhe oli fettiohjaimen sammutuksesta palaamisessa. Fettiohjaimen ohjaukset pitää olla pois ennen kuin fettiohjain palautuu shutdown-tilasta.

Lopulta robotti laskettiin renkaille ja testausta kokeiltiin 3G-yhteydellä. IP-kameroiden kuvaa ei saatu aikaseksi sijainnista johtuvan huonon 3G-verkon takia. Mutta kokeiltaessa WLAN-yhteyksillä etäohjausta ongelmia ei ollut.




*Kuva 17. Ohjelmiston testaus*

Aivan täydellisesti ei I2C-väylän häiriöistä päästy. Välillä ohjelma lähettää uudelleen moottoriohjaimelle komennot. Häiriöiden syntymekanismi ei ole onnistunut selviämään.

# 7 KEHITTÄMISMAHDOLLISUUDET

Prototyypin ollen kyseessä myös kehittämismahdollisuuksia on paljon jäljellä. Osa parannuskohteista oli jo etukäteen tiedossa.

Moottoriohjaimien suhteen on muutettavaa. I2C-väylä todennäköisesti jää pois myöhemmässä toteutuksessa. Sen häiriöiden kanssa on ollut liikaa ongelmia. Tämä muuttaa moottoriohjainta hieman. Muutos on mahdollista ottaa käyttöön kun ohjaimessa siirrytään PIC16F690-kontolleriin nykyisestä BV4206 piiristä. Samalla ohjaimeen saadaan itsensä tarkkailu mikä pysäyttää häiriötilanteessa. Myös virranrajoitus sekä moottorien takaisinkytkentä HALL-antureilla on ehdottomasti oltava seuraavissa malleissa. Ohjaimista tulee älykkäitä kertoen kuinka paljon, millä nopeudella ja millä kiihtyvyydellä.

Ohjausjärjestelmässä ei pahemmin ole ongelmia. USB-hub ei siedä pakkasta yhtään, joten se on vaihdettava. Ohjauskoneena NSLU2 on aivan liian hitaasti käynnistyvä, joten sen tilalle olisi hyvä saada tehokkaampaa korttia. Tehokkaammasta kortista olisi hyötyä myös kamerakuvan käsittelyssä.

Mekaniikassa on pikkuisia ongelmia. Taka-akselin moottorin kiinnikkeet olisi syytä muotoilla hieman eri tavalla. Nyt kun ne ottavat joissain tilanteissa rengasta ennen kiinni. Nostokykyä rajoittaa suhteellisen takana oleva teliakseli ja vastapainon muodostavan akuston suunniteltua pienempi paino. Korjauksena voisi vastapainoa lisätä tai lukitusmahdollisuus telille.

Edellä mainitut asiat kun on hoidettu voi kompassianturin ja gps-sijainnin ottaa mukaan ohjauksiin. Kuten myös pohtia kamerakuvan käyttämistä hyödyksi esimerkiksi OpenCV-kirjastojen kanssa.

# 8 YHTEENVETO

Opinnäytetyönä robotin prototyypin tekeminen tämän kokoisena oli palkitseva kokemus. On hyvin eri asia vain suunnitella koneella asiat toimimaan kuin tehdä ne käytännössä ja saada toimimaan kuten suunnitteli.

Kokonaisuudessaan rakentamisen eteneminen oli suhteellisen ongelmatonta. Komponenttien hidas saapuminen oli hankalaa kun toimitusajat oli useampia viikkoja. Piirilevyllä olleen pienen virheen takia yhden fettiohjaimen hajoaminen aiheutti pientä viivettä etenemisessä, koska kyseisiä fettiohjaimia oli vain tarvittava määrä. Tekemistä kuitenkin helpotti se että komponentteihin oli tullut hyvin tutustuttua aikaisemmin eikä siten niissä voinut kovin pahoja yllätyksiä esiintyä.

Loppuvaihe oli palkitsevinta aikaa kun kaikki alkoi pelaamaan yhteen. Harmittavimmaksi ja työläimmäksi ongelmallisimmaksi tässä työssä tuli I2C-väylän häiriöt joiden satunnaisuus teki vian paikallistamisen vaikeaksi. Robotista tuli mielestäni tavoitellun kaltainen kustannuksien pysyen alhaisina. Joitain asioita tekisin nyt toisin, mutta sitähän prototyyppi juuri on.

# LÄHTEET

*1. Microchip 2012. PIC16F690. [Verkkodokumentti. Viitattu 20.5.2012] Saatavissa: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en023112*

*2. Wikipedia. 2012. H-bridge. [Verkkodokumentti. Viittattu 20.5.2012] Saatavissa: http://en.wikipedia.org/wiki/H_bridge*

*3. International Rectifier. 2012. IRF1405. [Verkkodokumentti. Viitattu 20.5.2012] Saatavissa: http://www.irf.com/product-info/datasheets/data/auirf1405.pdf*

*4. International Rectifier. 2012. IR2110. [Verkkodokumentti. Viitattu 20.5.2012] Saatavissa: http://www.irf.com/product-info/datasheets/data/ir2110.pdf*

*5. ByVac. 2012. BV4206. [Verkkodokumentti. Viitattu 20.5.2012] Saatavissa: http://www.byvac.co.uk/downloads/datasheets/BV4206 DataSheet.pdf*

*6. NXP. 2012. 74HC08. [Verkkodokumentti. Viitattu 20.5.2012] Saatavissa: http://www.nxp.com/documents/data_sheet/74HC_HCT08.pdf*

*7. ITead Studio. 2012. ITead Studio. [Verkkodokumentti, viitattu 20.5.2012] Saavissa:http://iteadstudio.com/*

*8. Wikipedia. 2012. NSLU2. [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://en.wikipedia.org/wiki/NSLU2*

*9. Martin Michlmayr. 2012. Debian On NSLU2.[Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://www.cyrius.com/debian/nslu2/*

*10. Debian.org. 2012.Apt.[Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://wiki.debian.org/Apt*

*11. Robot-Electronics. 2012. USB-I2C Communivations Module. [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://robot-electronics.co.uk/htm/usb_i2c_tech.htm*

*12. Sure elektronics. 2012. DC-SS504.[Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://www.sureelectronics.net/goods.php?id=958*

*13. Linux.fi. 2012. Huawei E220/E270. [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://linux.fi/wiki/Huawei_E220/E270*

*14. DealExtreme. 2012. ND100-S. [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://www.dealextreme.com/p/nd-100s-sirf-iii-gps-usb-receiver-dongle-for-laptop-work-with-street-trips-37137*

*15. DealExtreme. 2012. Super Mini Bluetooth 2.0 Adebter Dongle. [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://www.dealextreme.com/p/super-mini-bluetooth-2-0-adapter-dongle-vista-compatible-11866*

*16. DealExtreme. 2012. Bluetooth RS232 TTL module. [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://www.dealextreme.com/p/wireless-bluetooth-rs232-ttl-transceiver-module-80711*

*17. DealExtreme. 2012. USB to Audio Crystal Blue Sound Card. [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://www.dealextreme.com/p/usb-to-audio-crystal-blue-sound-card-9723*

*18.DealExtreme. 2012. IP-400. [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://www.dealextreme.com/p/ip-400-standalone-security-surveillance-ip-network-camera-with-night-vision-15974*

*19. Zeemote. 2012. Zeemote JS1. [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://www.zeemote.com/*

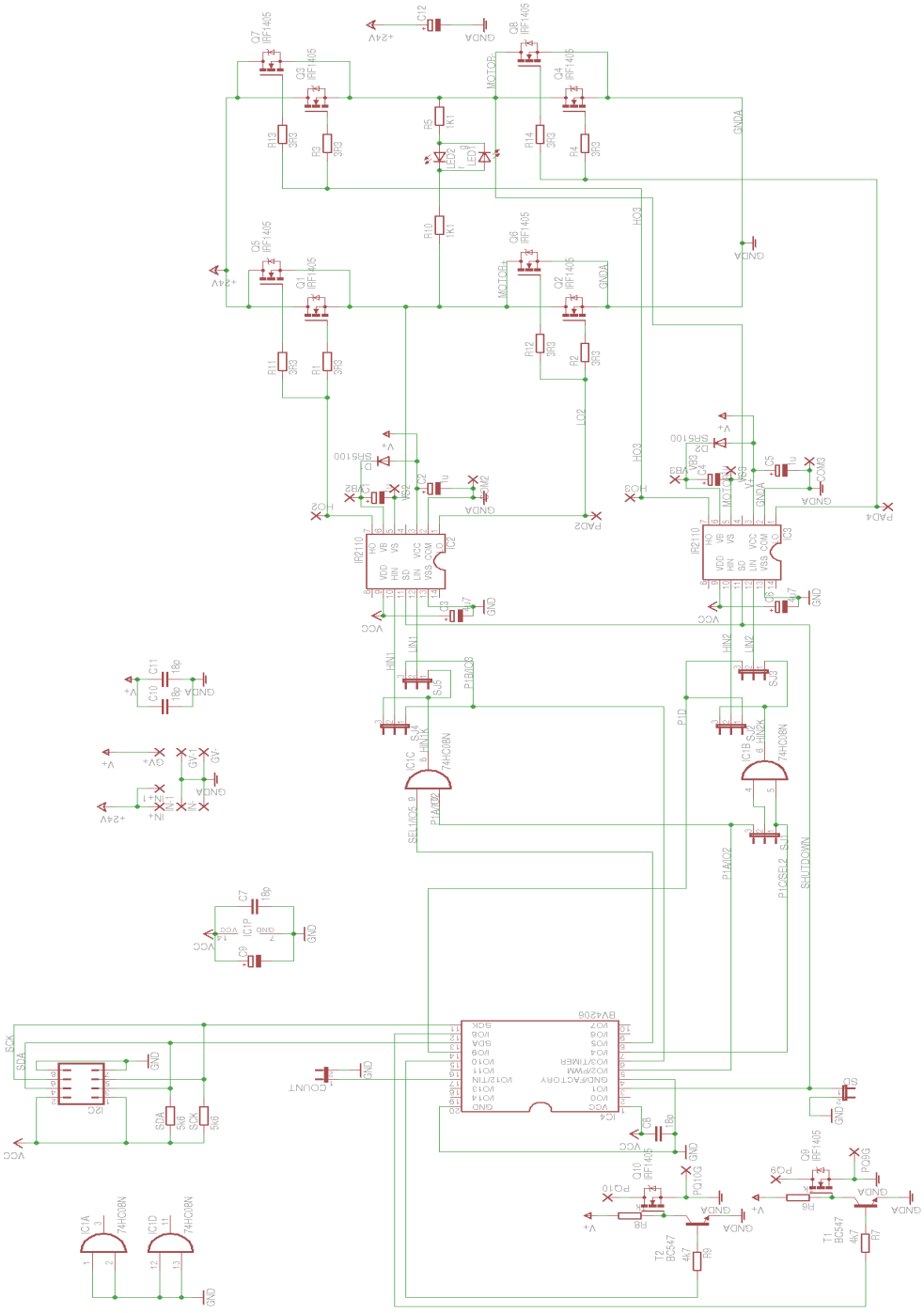
*20. Wikipedia. 2012. Logo (programming language). [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://en.wikipedia.org/wiki/Logo_(programming_language)*

*21. Sourceforge. 2012. eSpeak. [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://espeak.sourceforge.net/*

*22. Wikipedia. 2012. Gambas. [Verkkodokumentti, viitattu 20.5.2012] Saatavissa: http://fi.wikipedia.org/wiki/Gambas*

LIITTEET

LIITE 1



LIITE 2

```
/*
*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <iostream>
using namespace std;

#include "addrreg.h"
//#include "init.h"
//#include "tcpserver.h"
//#include "compass.h"
//#include "joystick.h"
//#include "i2clib.h"



//*********************************************************
//Prototyypit
unsigned char help();
int main(int argc, char *argv[]);
//*********************************************************



unsigned char help() {
	printf("\nKäyttö: 6x6 [VALITSIN]...\n\n");
	printf("      --help\t\tOhje\n");
	printf("  -1\t\t\tAlusta uudelleen\n");
	printf("  -2\t\t\tJoystick ohjaus\n");
	printf("  -3\t\t\tLogo komennoilla socket ohjaus\n");
	exit(0);
}



int main(int argc, char *argv[]) {

// Komentorivi parametrien luku help komennon osalta
	if (argc > 1)		{
		if ( strcmp(argv[1], "--help") == 0)
			help();
	}

	char alustatesti = alusta();
	if (alustatesti < 0)	{ // piirien alustaminen
		return -1;
	}


// Komentorivi parametrien luku
	if (argc > 1) {
		#ifdef _JOYSTICKE
			if ( strcmp(argv[1], "-2") == 0)
				joystickohjaus();
```

```
		#endif
		#ifdef _TCPSERVERE
			else if ( strcmp(argv[1], "-3") == 0)	{
				if (argc > 2)
					tcpserver(atoi(argv[2]));
				else
					tcpserver(TCPport);
			}
		#endif
	}


	int valinta;
	do {
		cout<<"\nVALIKKO";
		cout<<"\n\n1 Alusta uudelleen";
		#ifdef _JOYSTICKE
			cout<<"\n2 Joystick ohjaus";
		#endif
		#ifdef _TCPSERVERE
			cout<<"\n3 Logo komennoilla socket ohjaus";
		#endif
		cout<<"\n4 Nappaimisto ohjaus";
		cout<<"\n0 Lopeta \n\nValintasi: ";
		cin>>ws>>valinta;
		switch(valinta)
		{
			case 0: cout<<"\nValittu lopetus\n\n";
					exit(0);
					break;
			case 1: cout<<"\nValittu uudelleen alustus\n";
					alusta();
					break;
			#ifdef _JOYSTICKE
			case 2: cout<<"\nValittu Joystick\n";
					joystickohjaus();
					break;
			#endif
			#ifdef _TCPSERVERE
				case 3: cout<<"\nValittu TCP\n\n";
						tcpserver(TCPport);
						break;
			#endif
			#ifdef _COMMPASSE
				case 4: cout<<"\nValittu toiminto ei viela valmis\n\n";
						compassread();
						break;
			#endif

		}

	} while (valinta != 0);

  return 0;
}
```

LIITE 3

```
#ifndef _INIT_H // tarkistetaan onko jo määritetty
#define _INIT_H

#include "addrreg.h"

//#include "bv4206.h"
//#include "hbridge.h"
//#include "addrreg.h"
//#include "logo.h"
//#include "i2clib.h"

//*******************************************************
//Prototyypit
char alustasilta(unsigned char Motor);
char alusta();
//*******************************************************



// ************************************************************
char alustasilta(unsigned char Motor) {
        char I2Csend;
//  usleep(500000);
        char test = I2Cread(Motor, BV4206Firmware);
        #ifndef _TEST_NOI2C
        if (test == 2) {
                                                                                                        //
firmware 2,94
                printf("H-bridge 0x%x.. OK \n", Motor);
                        // error
        #endif
                // Motor shutdown
        I2Csend = I2Cwrite2(Motor, BV4206SetIO, Shutdown, 0x00);                                        //
shutdown pin out asentoon
                if ( I2Csend == 0)

                        return -1;
                I2Csend = ShutdownON(Motor);


                //I2Csend = I2Cwrite2(Motor, BV4206SetPin, Shutdown, 0x01);
        // shutdown päälle
                if ( I2Csend < 1)

                        return -1;



                //pwm off
          I2Csend = I2Cwrite(Motor, BV4206PWMAct, 0x00);
                                //PWM pois
                if ( I2Csend < 1)

                        return -1;
```

```
        //motor bridge
        I2Csend = I2Cwrite2(Motor, BV4206SetIO, HbridgeFetHF, 0x00);          // output
        if ( I2Csend < 1)

                return -1;


        I2Csend = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHF, 0x00);         // 0
        if ( I2Csend < 1)

                return -1;


  I2Csend = I2Cwrite2(Motor, BV4206SetIO, HbridgeFetHR, 0x00);          // output
        if ( I2Csend < 1)

                return -1;


 I2Csend = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHR, 0x00);          // 0
        if ( I2Csend < 1)

                return -1;


  I2Csend = I2Cwrite2(Motor, BV4206SetIO, HbridgeFetLF, 0x00);          // output
        if ( I2Csend < 1)

                return -1;


  I2Csend = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLF, 0x00);         // 0
        if ( I2Csend < 1)

                return -1;


  I2Csend = I2Cwrite2(Motor, BV4206SetIO, HbridgeFetLR, 0x00);          // output
        if ( I2Csend < 1)

                return -1;


  I2Csend = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLR, 0x00);                 // 0
        if ( I2Csend < 1)

                return -1;


//extra fet
  I2Csend = I2Cwrite2(Motor, BV4206SetIO, HbridgeFet9, 0x00);                   // output
        if ( I2Csend < 1)

                return -1;


  I2Csend = I2Cwrite2(Motor, BV4206SetPin, HbridgeFet9, 0x00);          // 0
        if ( I2Csend < 1)
```

```
                return -1;


        I2Csend = I2Cwrite2(Motor, BV4206SetIO, HbridgeFet10, 0x00);            // output
                if ( I2Csend < 1)

                        return -1;


        I2Csend = I2Cwrite2(Motor, BV4206SetPin, HbridgeFet10, 0x00);           // 0
                if ( I2Csend < 1)

                        return -1;



        //shotdown bitti

                I2Csend = ShutdownOFF(Motor);


        //I2Csend = I2Cwrite2(Motor, BV4206SetPin, Shutdown, 0x00);                                     //
shutdown pois
                if ( I2Csend < 1)

                        return -1;



        #ifndef _TEST_NOI2C
        }
        else {
                printf("H-bridge 0x%x.. FAIL\n", Motor);
        // error
                return -1;
        }
        #endif
        return 0;
}
// ************************************************************


// ************************************************************
char alusta() {

        unsigned char i2crevi=0;

        // ************************************************************
        // USB-portin tarkistus
        #ifndef _I2C_NSLU // jos ei i2c-NSLU
        printf("\nUSB-COM tarkistus /dev/USBi2c..");

        int fd; // File descriptor for the port
        fd = open_port();
                if ( fd < 0) {
                        printf("FAIL\n\nAnna komento ln -s /dev/ttyUSB0 /dev/USBi2c tai kytke
sovitin\n\n");
                        return -1;
                }


  close(fd);
```

```
	printf("OK\n\n");

	printf("USB-I2C tarkistus ");
	i2crevi = I2Crevision();
	printf("revision %x..", i2crevi);
		if ( i2crevi < 1)	{
			printf("FAIL\n");
			return -1;
		}
	printf("OK\n\n");
	#endif
	// *********************************************************



	char alustasiltatesti = alustasilta(MotorLeft);                                    // Vasemman
moottorin alustus
	if ( alustasiltatesti < 0)
                                                        // yhteyden tarkistus
		return -1;

		// virheestä poikki

	alustasiltatesti = alustasilta(MotorRight);
	// Oikean moottorin alustus
	if ( alustasiltatesti < 0)

		return -1;



	alustasiltatesti = alustasilta(MotorLift);                                         //
Nostimen moottorin alustus
	if ( alustasiltatesti < 0)

		return -1;


	alustasiltatesti = alustasilta(MotorTilt);                                         //
Nostimen moottorin alustus
	if ( alustasiltatesti < 0)

		return -1;


	return 0;
}

#endif
```

LIITE 4

```
//Piirien osoitteet
#ifndef _ADDRREG_H // tarkistetaan onko jo määritetty
	#define _ADDRREG_H

	//testimodet
	#define _TEST_LOGO
	#define _TEST_TCP
	#define _TEST_JOY
	//#define _TEST_NOI2C	// I2C kirjoitukset vain tulostaen

	#include "bv4206.h"
	//#include "bv4206.h"

	#define _I2C_NSLU		// I2C liitäntä
	//#define _HbridgeIC_PIC	// H-sillan piiri

	// Palvelut
	#define _TCPSERVERE	// TCP-palvelin
	//define _COMPASSE		// Compassi
	#define _JOYSTICKE		// Joystick

	// Maksmimi PWM
	#define DutyMax 405 // rajoitetaan maksimia samalla ir2110 takia

	// H-sillat
	#define MotorLeft 0x42
	#define MotorRight 0x44
	#define MotorLift 0x46
	#define MotorTilt 0x48

	// Laskurit
	#define CounterL 0xa0
	#define CounterR 0xa2

	// USB I2C
	#ifndef _I2C_NSLU // jos ei i2c-NSLU
		#include "serialport.h"
		#include "i2crw.h"
		#define I2C_SGL 0x53
		#define I2C_MUL 0x54
		#define I2C_AD1 0x55
		#define I2C_AD2 0x56
		#define I2C_USB 0x5a
	#endif

	// NSLU I2C
	#ifdef _I2C_NSLU // jos i2c-NSLU
		#define I2CDEVICE "/dev/i2c-0"
		#define i2cdelay 5000
		#include "i2clib.h"
	#endif

	// Kompassi
	#ifdef _COMPASSE //
		#define MMC2120MG 0x60
		#include "compass.h"
	#endif
```

```
//* * * * * * * * * * * * * * * *
// H-bridge
// Shutdown
#define Shutdown 1
//          Ajo fetit
//PIC - Jumpers 1-2
#ifdef _HbridgeIC_PIC                   // jos PIC piiri
        #define HbridgeFetHF 2          // (P1A) PWM Foward High
        #define HbridgeFetLR 3          // (P1B) Reverse Low
        #define HbridgeFetHR 4          // (P1C) PWM Reverse High
        #define HbridgeFetLF 9          // (P1D) Foward Low
#endif

// Jumper 2-3
// PWM signal both highfet is 2
#ifndef _HbridgeIC_PIC                          // jos ei ole PIC piiri
        #define HbridgeFetLR 3                  // Reverse Low Q2/IC2LIN
        #define HbridgeFetHR 4                  // Select PWM Reverse Q3SEL/IC3HINSEL
        #define HbridgeFetHF 5                  // Select PWM Foward Q1SEL/IC2HINSEL
        #define HbridgeFetLF 9                  // Foward Low Q4/IC3LIN
#endif

// Kytkin fetit
#define HbridgeFet9 8
#define HbridgeFet10 10

#define GMAX 20             // Kiihtyvyys maksimi
#define TIRESIZE 200        // Rengaskoko 200mm
#define MOTORNM 10                  // Moottorin vääntö 1Nm
#define MOTORRPM 3000  // Moottorin kierrosluku 3000rpm
#define GEARRATION 258  // Vaihteiston välitys 25,8:1

#include "hbridge.h"
#include "init.h"
#include "logo.h"


// Joystick
#ifdef _JOYSTICKE
        #define JOYSTICK_DEVNAME "/dev/input/js0"
        #include "joystick.h"
#endif

// TCP
#ifdef _TCPSERVERE
        #include "tcpserver.h"
        #define TCPport 3333
#endif


#endif /* _ADDRREG_H */
```

LIITE 5

```
#ifndef _SERIALPOR_H // tarkistetaan onko jo määritetty
#define _SERIALPORT_H

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>

struct termios options;

int open_port(void) {
  // this function taken from http://www.robot-electronics.co.uk/forum/viewtopic.php?f=5&t=165
  int fd; // File descriptor for the port
  fd = open("/dev/USBi2c", O_RDWR | O_NOCTTY);
  if (fd == -1) {
    perror("Unable to open /dev/USBi2c"); // Could not open the port.
                        return -1;
  }
  else {
    fcntl(fd, F_SETFL, 0);

    // Get the current options for the port...
    tcgetattr(fd, &options);

    // Set the baud rates to 19200...
    cfsetispeed(&options, B19200);

    // Enable the receiver and set local mode...
    options.c_cflag |= (CLOCAL | CREAD);

    // Set no parity bit
    options.c_cflag &= ~PARENB;

    // Set 2 stop bits
    options.c_cflag &= ~CSTOPB;
    // Set 1 stop bits
//    options.c_cflag |= CSTOPB;

    // Set the character size
    options.c_cflag &= ~CSIZE;
    options.c_cflag |= CS8;

    // Set the new options for the port...
    tcsetattr(fd, TCSANOW, &options);

                        //Ei jumita
//                      fcntl(fd, F_SETFL, FNDELAY);

  }
  return (fd);
}

#endif
```

LIITE 6

```
#ifndef _I2CRW_H // tarkistetaan onko jo määritetty
#define _I2CRW_H


//*******************************************************
//Prototyypit
unsigned char I2Cread(unsigned char addr, unsigned char reg);
unsigned char I2Cread2(unsigned char addr, unsigned char reg);

unsigned char I2Cwrite(unsigned char addr, unsigned char reg, unsigned char byte);
unsigned char I2Cwrite2(unsigned char addr, unsigned char reg, unsigned char hbyte, unsigned char lbyte);

unsigned char I2CwriteChangeAddr(unsigned char addr, unsigned char newaddr);
unsigned char I2Crevision();
//*******************************************************




unsigned char I2Cread(unsigned char addr, unsigned char reg) {
  unsigned char bufs[4], bufr[1];
  int fd;

  bufs[0] = I2C_AD1;                        // ftdi mode (multibyte device)
  bufs[1] = addr + 1;                       // S-addr +1 address of the i2c device
  bufs[2] = reg;                    // registers
  bufs[3] = 0x01;                   // bytes to read

        #ifndef _TEST_NOI2C
  fd = open_port();
    write(fd, bufs, 4);
    usleep(1000);
    read(fd, bufr, 1);
  close(fd);

        usleep(500000);

        #ifdef _TEST
                printf("bufr[0]: %u\n", bufr[0]);       // Display the data to the screen
        #endif
        #endif
        #ifdef _TEST_NOI2C
        printf("TEST_NOI2C: I2Cread(addr=%x, reg=%x\n",addr,reg);
        bufr[0] = 1;
        #endif

        return bufr[0];
}


unsigned char I2Cread2(unsigned char addr, unsigned char reg) {
  unsigned char bufs[4], bufr[2];
  int fd;

  bufs[0] = I2C_AD1;                        // ftdi mode (multibyte device)
  bufs[1] = addr + 1;                       // S-addr +1 address of the i2c device
```

```
 bufs[2] = reg;                         // registers
 bufs[3] = 2;                           // bytes to read

        #ifndef _TEST_NOI2C
 fd = open_port();
  write(fd, bufs, 4);
  usleep(1000);
  read(fd, bufr, 2);
 close(fd);

        usleep(500000);

        #ifdef _TEST
                printf("bufr[0]: %x, bufr[1]: %x\n", bufr[0], bufr[0]);
        #endif
        #endif
        #ifdef _TEST_NOI2C
        printf("TEST_NOI2C: I2Cread2(addr=%x, reg=%x\n",addr,reg);
        bufr[0] = 1;
        #endif

        return bufr[0];
}


// * * * * *
// Write

unsigned char I2Cwrite(unsigned char addr, unsigned char reg, unsigned char byte) {
 unsigned char bufs[5], bufr[1];
 int fd;

 bufs[0] = I2C_AD1;                             // ftdi mode (multibyte device)
 bufs[1] = addr;                        // S-addr address of the i2c device
 bufs[2] = reg;                         // registers
 bufs[3] = 0x01;                        // bytes to write
 bufs[4] = byte;

        #ifndef _TEST_NOI2C
 fd = open_port();
  write(fd, bufs, 5);
  read(fd, bufr, 1);
 close(fd);

        if (bufr[0] == 0) {                                                     //yritetään uudelleen
                printf("I2Cwrite addr=%xh reg=%xh byte=%xh failed!\n", addr, reg, byte);
                usleep(500000);
          fd = open_port();
            write(fd, bufs, 5);
            read(fd, bufr, 1);
          close(fd);
                if (bufr[0] == 0)
                        printf("I2Cwrite addr=%xh reg=%xh byte=%xh failed two times!\n", addr, reg,
byte);
        }
        #endif
        #ifdef _TEST_NOI2C
        printf("TEST_NOI2C: I2Cwrite(addr=%x, reg=%x, data=%x\n",addr,reg,byte);
        #endif
        return bufr[0];                         // palautetaan kirjoituksen onnistuminen
}

unsigned char I2Cwrite2(unsigned char addr, unsigned char reg, unsigned char hbyte, unsigned char lbyte) {
```

```
  unsigned char bufs[6], bufr[1];
  int fd;

  bufs[0] = I2C_AD1;                          // ftdi mode (multibyte device)
  bufs[1] = addr;                     // S-addr address of the i2c device
  bufs[2] = reg;                      // registers
  bufs[3] = 2;                        // bytes to write
  bufs[4] = hbyte;
  bufs[5] = lbyte;

        #ifndef _TEST_NOI2C
  fd = open_port();
    write(fd, bufs, 6);
    read(fd, bufr, 1);
  close(fd);

        if (bufr[0] == 0)     {
                printf("I2Cwrite2 addr 0x%x reg 0x%x failed!\n", addr, reg);
                usleep(500000);
    fd = open_port();
      write(fd, bufs, 6);
      read(fd, bufr, 1);
    close(fd);
                if (bufr[0] == 0)
                        printf("I2Cwrite2 addr 0x%x reg 0x%x failed two times!\n", addr, reg);
        }
        #endif
        #ifdef _TEST_NOI2C
        printf("TEST_NOI2C: I2Cwrite2(addr=%x, reg=%x, hbyte=%x, lbyte=%x\n",addr,reg,hbyte,lbyte);
        #endif

        return bufr[0];
}


/*
//<S-addr><0x99><New-Addr><0x55><0xaa><Current-Addr><Stop>
unsigned char I2CwriteChangeAddr(unsigned char addr, unsigned char newaddr) {
  unsigned char bufs[8], bufr[1];
  int fd;

  bufs[0] = I2C_AD1;                          // ftdi mode (multibyte device)
  bufs[1] = addr;                     // S-addr address of the i2c device
  bufs[2] = 0x99;                     // registers
  bufs[3] = 4;                        // bytes to write
  bufs[4] = newaddr;
  bufs[5] = 0x55;
  bufs[6] = 0xaa;
  bufs[7] = addr;

  fd = open_port();
    write(fd, bufs, 8);
    read(fd, bufr, 1);
  close(fd);

        if (bufr[0] == 0)
                printf("I2CwriteChangeAddr failed %x\n",bufr[0]);

        return bufr[0];
}
*/

// USB-I2C revision
```

```
unsigned char I2Crevision() {
  unsigned char bufs[4], bufr[1];
  int fd;

  bufs[0] = I2C_USB;                              // A range of commands to the USB-I2C module
  bufs[1] = 0x01;                         // Revision
  bufs[2] = 0xF0;                         // trash data
  bufs[3] = 0x0F;                         // trash data

            #ifndef _TEST_NOI2C

  fd = open_port();
              write(fd, bufs, 4);
                      usleep(1000000);
    read(fd, bufr, 1);
  close(fd);
            #endif
            #ifdef _TEST_NOI2C
            printf("TEST_NOI2C: I2Cread(addr=%x, reg=%x\n",addr,reg);
            bufr[0] = 2;
            #endif

return bufr[0];
}



#endif
```

LIITE 7

```
/* I2C-GP I/O

ByVac BV4206

The BV4206 is an I2C two wire compatible integrated circuit with 15 general purpose,
user configurable input or output lines. In addition it has a Pulse Width Modulated output
and Timer features.

Features    I2C up to 400kHz
            Simple command set
            15 inputs or outputs or mixed
            High current source/sink for direct LED drive (25mA)
            PWM running at 19.6 KHz
            Timer with internal or external clock source
            EEPROM for general use
            Sleep mode to save power
            Operating voltage 2.0 to 5.5V
            Current 1.8mA @ 5V
            Sleep current 200uA
*/

#define BV4206AddDefault 0x42
//const unsigned char BV4206AddDefault = 0x42;

#define BV4206SetIO 0x01
//const unsigned char BV4206SetIO = 0x01;
/*          Name: Set I/O
            Format: <S-addr><1><channel><1 or 0><Stop>
            The channel is a value from 0 to 14 and refers to the I/O number for a given pin. If this is
            followed by a 0 then that pin will be an output, if followed by a 1 then it will be an input.
            At reset all of the pins are configured for input.          */

#define BV4206SetPin 0x02
//const unsigned char BV4206SetPin = 0x02;
/*          Name: Set Pin (channel)
            Format: <S-addr><2><channel><1 or 0><Stop>
            This will only work when the selected pin is set to an output using command 1. The pin will
            reflect the second byte thus if the second byte were a 0 the pin would be low-*/

#define BV4206ReadPin 0x03
//const unsigned char BV4206ReadPin = 0x03;
/*          Name: Read Pin
            Format:<S-addr><3><channel><RAddr><bit><Stop>
            This returns the value on the selected pin (channel) which will be either 0 or 1.          */

#define BV4206PullUp 0x04
//const unsigned char BV4206PullUp = 0x04;
/*          Name: Weak pull ups
            Format: < S-addr><4><0 or 1><Stop>
            This enables (1) or disables (0) weak pull ups on the following channels.
            0,1,7,8,12,13 and 14 At reset the week pull ups are disabled. The
            effect of this can be seen when reading the   pins using command 3.          */

#define BV4206TimerOP 0x10
//const unsigned char BV4206TimerOP = 0x10;
/*          Name: Timer o/p Control
            Format: <S-addr><0x10><0 or 1><Stop>
            By default pin 6 is an input, using this command with '1' will set this pin to an output
            and each time the timer overflows it will toggle this pin.
            Once this command is used to set pin 6 to be a timer output it will remain an output until
            command 1 is used to set it back to an input. If '0' is used the pin will remain an output but
            will not be toggled by the timer. 0 sets pin 6 as normal i/o pin (default)
            1 sets pin 6 to be o/p from timer  */

#define BV4206Precaler 0x11
```

```
//const unsigned char BV4206Precaler = 0x11;
/*          Name: Timer prescaler
            Format: <S-addr><0x11><0 to 8><Stop>
            This sets the prescaler value.
            0 = 1:2, 1 = 1:4, 2 = 1:8, 3 = 1:18, 5 = 1:64,6 = 1:128, 7 = 1:256, 8 = 1:1      */

#define BV4206TimerValue 0x12
//const unsigned char BV4206TimerValue = 0x12;
/*          Name: Timer value
            Format: <S-addr><0x12><0 to 230><Stop>
            This value along with the prescaler value will set the time frequency or period. Table 3 gives
            some examples of what can be expected for the various combinations of values.
            Note that the maximum value for this is 230, a value larger than this will be ignored.     */

#define BV4206TimerRead 0x13
//const unsigned char BV4206TimerRead = 0x13;
/*          Name: Read Timer
            Format: <S-addr><0x13><RAddr><value><Stop>
            This will read the value of the timer register. The timer register is loaded with the value set
            by command 0x12, it then increments until it reached 255 and then is loaded again with the
            value set by command 0x12. The minimum value that can be read is
            therefore set by command 0x12.  */

#define BV4206TimerSource 0x14
//const unsigned char BV4206TimerSource = 0x14;
/*          Name: Timer Source
            Format: <S-addr><0x14><0 or 1><Stop>
            The timer can be fed from the internal oscillator or an external pin (pin 17). This
            command select the source of the timer. At reset the source is from pin 17 that
            corresponds to '0' for the second byte. A 1 will set the source to the internal oscillator.
            0 sets input from pin 17 (default) 1 sets input from internal oscillator          */

#define BV4206TimerExtIP 0x15
//const unsigned char BV4206TimerExtIP = 0x15;
/*          Name: Timer External i/p control
            Format: <S-addr><0x15><0 or 1><Stop>
            When the timer source is provided externally through pin 17, this controls if the timer is
            incremented from a low to high transition or a high to low transition.
            0 = increment on low-high 1 = increment on high-low.  */

#define BV4206PWMAct 0x20
//const unsigned char BV4206PWMAct = 0x20;
/*          Name: PWM Activate
            Format: <S-addr><0x20><0 or 1><Stop>
            This command will set pin 5 to be an output and start the PWM working.
            Using a 1 as the second byte will make the pin high when the PWM is fully on. Using a 0 as
            the second byte will make the PWM low when fully on. The PWM period is set to 19.61kHz.      */

#define BV4206PWMWidth 0x21
//const unsigned char BV4206PWMWidth = 0x21;
/*          Name: PWM Pulse Width
            Format: <S-addr><0x21><high><low><Stop>
            The width of the pulse within the PWM period is determined by this value made up of a high
            byte and a low byte. The actual value can be between 0 and 408. 0 is fully off and 408 is
            fully on, larger numbers than 408 will not make any difference.
            The value must be sent as two bytes.          */

#define BV4206Test 0x55
//const unsigned char BV4206Test = 0x55;
/*          Name: Test
            Format: < S-addr><0x55><start><RAddr><Value..><NACK><Stop>
            It can be used for testing the presence or other wise of a device at a particular address.
            It is also useful during the development stage to ensure that the I2C is working for that
            device.     */

#define BV4206EERead 0x90
//const unsigned char BV4206EERead = 0x90;
/*          Name: Read EEPROM
            Format: <S-addr><0x90><EEAddress><R-Addr><data…><Stop>
            This command will allow a single or several bytes to be read from a specified EEPROM
            address.    */
```

```
#define BV4206EEWrite 0x91
//const unsigned char BV4206EEWrite = 0x91;
/*      Name: Write EEPROM
        Format: <S-addr><0x91><EEAddress><data…><Stop>
        This command will write one or more, up to a maximum of 30 bytes at any one time, to be
        written to the EEPROM. Address 0 of the EEPROM is the device address and this cannot
        be written to by this command. A special command 0x99 is used for this purpose.
        The first 16 bytes 0 to 15 are reserved for system use.   */

#define BV4206EEEnd 0x93
//const unsigned char BV4206EEEnd = 0x93;
/*      Name: End of EEPROM
        Format: <S-addr><0x93><RAddr><data><Stop>
        The system only uses a small portion of the first part of the EEPROM, the rest of the
        EEPROM can be used for user data or other purposes depending on the device. This
        command returns a single byte that will determine the last writeable address of
        EEPROM, normally 0xFF.          */

#define BV4206Sleep 0x94
//const unsigned char BV4206Sleep = 0x94;
/*      Name: Sleep
        Format: <S-addr><0x94><Stop>
        This will put the IC into sleep mode. Any other command will wake the IC. Depending on the
        device this can be a considerable power saving.          */

#define BV4206Reset 0x95
//const unsigned char BV4206Reset = 0x95;
/*      Name: Reset
        Format: <S-addr><0x95><Stop>
        Resets the device, this is equivalent to disconnecting and then connecting the power again.          */

#define BV4206ChangeAddTemp 0x98
//const unsigned char BV4206ChangeAddTemp = 0x98;
/*      Name: Change Device Address Temporary
        Format: <S-addr><0x98><New-Addr><Stop>
        This will change the device address with immediate effect and so the next command
        must use the new address. The address must be a write address (even number) Odd
        numbers will simply be ignored. The effect will last as long as the device is switched on.
        Resetting the device will restore the address to its original value. The address is stored in
        EEPROM location 0. */

#define BV4206ChangeAdd 0x99
//const unsigned char BV4206ChangeAdd = 0x99;
/*      Name: Change Device Address Permanent
        Format: <S-addr><0x99><New-Addr><0x55><0xaa><Current-Addr><Stop>
        This command changes the address immediately (the next command will need to
        use the new address) and permanently (see hardware reset). The address must be a write
        address (even number) and follow the sequence exactly.
        Permanent in this case means that the device will retain this address after power down, i.e.
        it is stored in EEPROM. Should anything go wrong the default address can be restored by
        using a hardware reset.          */

#define BV4206Firmware 0xa0
//const unsigned char BV4206Firmware = 0xa0;
/*      Name: Firmware version
        Format: <S-addr><a0><RAddr><byte><byte><Stop>
        This simply returns two bytes that represents the firmware version.          */
```

LIITE 8

```
#ifndef _JOYSTICK_H // tarkistetaan onko jo määritetty
	#define _JOYSTICK_H


#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

#include "joystickh.h"

//*********************************************************
//Prototyypit
int open_joystick();
int read_joystick_event(struct js_event *jse);
void close_joystick();
int get_joystick_status(struct wwvi_js_event *wjse);
int moottoriteho(int MoPower, int Motor);
int joystickohjaus();
//*********************************************************



static int joystick_fd = -1;

int open_joystick() {
	joystick_fd = open(JOYSTICK_DEVNAME, O_RDONLY | O_NONBLOCK); /* silsuun[1]d write for
force fesilsuun[4]ack? */
	if (joystick_fd < 0)
		return joystick_fd;

	/* maybe ioctls to interrogate features here? */

	return joystick_fd;
}

int read_joystick_event(struct js_event *jse)	{
	int bytes;

	bytes = read(joystick_fd, jse, sizeof(*jse));
	if (bytes == -1)
		return 0;
	if (bytes == sizeof(*jse))
		return 1;

	printf("Unexpected bytes from joystick:%d\n", bytes);

	return -1;
}

void close_joystick()		{
	close(joystick_fd);
}

int get_joystick_status(struct wwvi_js_event *wjse)
```

```
{
	int rc;
	struct js_event jse;
	if (joystick_fd < 0)
		return -1;

	// memset(wjse, 0, sizeof(*wjse));
	while ((rc = read_joystick_event(&jse) == 1)) {
		jse.type &= ~JS_EVENT_INIT; /* ignore synthetic events */
		if (jse.type == JS_EVENT_AXIS) {
			switch (jse.number) {
			case 0: wjse->stick_x = jse.value;
				break;
			case 1: wjse->stick_y = jse.value;
				break;
			default:
				break;
			}
		} else if (jse.type == JS_EVENT_BUTTON) {
			if (jse.number < 10) {
				switch (jse.value) {
				case 0:
				case 1: wjse->button[jse.number] = jse.value;
					break;
				default:
					break;
				}
			}
		}
	}
	// printf("%d\n", wjse->stick1_y);
	return 0;
}




int moottoriteho(int MoPower, int Motor)	{
	if (MoPower > 1)   {			// ajetaan eteen jos joystick yli 1%
		if (MoPower > 100)		// estetään ylittämästä maksimia jotta yläfetit pysyy auki
			MoPower = 100;
		MoPower = (MoPower * DutyMax)/100;
		if (MoPower > DutyMax)	// estetään ylittämästä maksimia jotta yläfetit pysyy auki
			MoPower = DutyMax;
		#ifdef _TEST_JOY
		printf("TEST_JOY: MoPower=%d\n", MoPower);
		#endif
		unsigned char hbyte= (unsigned char) (MoPower >> 8);
		unsigned char lbyte= (unsigned char) (MoPower);


		I2Cwrite2(Motor, BV4206PWMWidth, hbyte, lbyte);
	}
	else

			// nollataan pwm
		I2Cwrite2(Motor, BV4206PWMWidth, 0x00, 0x00);
	return 0;
}
```

```
// *********************************************************
// Joystickohjaus
//int main(int argc, char *argv[]) {

int joystickohjaus() {

	int fd, rc, done = 0;
 unsigned char lbyte, hbyte;
	unsigned char but[4];			// 0=muutosta tiloissa, 1=A, 2=B, 3=C, 4=D
	unsigned char silsuun[4];	// 0=muutosta tiloissa, 1=vasen, 2=oikea, 3=vanhavasen, 4=vanhaoikea
	int Joys[4];
	Joys[0] = 0;
	Joys[1] = 0;
	Joys[2] = 0;
	char alustatesti;

	struct js_event jse;

	PWMactive(MotorLeft);
	PWMactive(MotorRight);

	fd = open_joystick();
		if (fd < 0) {
			printf("Joystick open failed.\n");
			printf("Chip reset.\n");
			alustatesti = alusta();
			  if (alustatesti < 0) { // piirien alustaminen
					printf("Chip reset.\n");
					return -1;
				}
			exit(1);
		}

		while (!done) {
			rc = read_joystick_event(&jse);
			//usleep(1000);
			if (rc == 1) {

	// jos eventti
					#ifdef _TEST_JOY
						printf("TEST_JOY: Value %8hd, type: %3u, axis/button: %u\n",
jse.value, jse.type, jse.number);
					#endif

					// Siirretään eventti muuttujaksi
					if (jse.type == 2){
						if (jse.number == 0) {
							Joys[1] = jse.value / 330;
											//korjaa jakaja
							Joys[0] |= 1;
							Joys[3] = (Joys[2] + Joys[1])/2;
							Joys[4] = (Joys[2] - Joys[1])/2;
						}
						else	{
							Joys[2] = (jse.value / 330) *-1;
									//korjaa jakaja
							Joys[0] |= 2;
							Joys[3] = (Joys[2] + Joys[1])/2;
							Joys[4] = (Joys[2] - Joys[1])/2;
						}
					}
```

```
		if (jse.type == 1){
			if (jse.number == 0)		{		// A -NAPPI
				but[0] |= 1;
				but[1] = jse.value;
			}
			else if (jse.number == 1)	{		// B -NAPPI
				but[0] |= 2;
				but[2] = jse.value;
			}
			else if (jse.number == 2)	{		// C -NAPPI
				but[0] |= 4;
				but[3] = jse.value;
			}
			else		{
				but[0] |= 8;			//D -NAPPI
				but[4] = jse.value;
			}
		}

	// Sillan suunnan määritys
	if (Joys[0] > 0)	{		// jos muutosta joystickeissä
		if (Joys[3] > 1)	{		/ eteen jos joystick yli raja-arvon
			silsuun[0] |=		1;		// muutosta sillassa
			silsuun[1] = 0;			// vasen eteen
		}
		else if (Joys[3] < -1)		{		// taakse jos joystick yli raja-arvon
			silsuun[0] |= 2;		// muutosta sillassa
			silsuun[1] = 1;			// vasen taakse
			Joys[3] = Joys[3] * (-1);	// invertoidaan
		}
		else if (silsuun[1] != 3)	{		// jollei jarru päällä
			silsuun[0] |= 4;		// muutosta sillassa
			silsuun[1] = 3;			// keskellä jarru
		}

		if (Joys[4] > 1)	{			// eteen jos joystick yli raja-arvon
			silsuun[0] |= 8;	// muutosta sillassa
			silsuun[2] = 0;			// oikea eteen
		}
		else if (Joys[4] < -1)		{		// taakse jos joystick yli raja-arvon
			silsuun[0] |= 16;	// muutosta sillassa
			silsuun[2] = 1;			// oikea taakse
			Joys[4] = Joys[4] * (-1);	// invertoidaan
		}
		else if (silsuun[2] != 3){			// jollei jarru päällä
			silsuun[0] |= 32;	//muutosta sillassa
			silsuun[2] = 3;			// keskellä jarru
		}
	}


	// Sillan ohjaus
	if (silsuun[0] > 0)	{				// jos muutosta suunnassa
		if (silsuun[3] != silsuun[1]) {		// jos tilamuutos vasemmassa
			silsuun[3] = bridgedirect(silsuun[1], MotorLeft);
		}

		if (silsuun[4] != silsuun[2]) {		// jos tilamuutos oikeassa
			silsuun[4] = bridgedirect(silsuun[2], MotorRight);
		}
		silsuun[0] = 0;
	}
```

```
                        // Moottorin tehonsäätö
                        if (Joys[0] > 0){
                                // jos muutos
                                moottoriteho(Joys[3], MotorLeft);

                                moottoriteho(Joys[4], MotorRight);
                                Joys[0] = 0;                    // kuitataan muutos
                        }

                        // Nostolaite
                        if ( but[0] > 0) {                      // jos tilamuutoksia
                                but[0] = 0;
                                if ( but[1] == 1)       {
                                        bridgedirect(0, MotorLift);     // ylös (eteen)
                                hbyte= (unsigned char) (DutyMax >> 8);
                                lbyte= (unsigned char) (DutyMax);
                                        I2Cwrite2(MotorLift, BV4206PWMWidth, hbyte, lbyte);
                                }
                                else    if ( but[2] == 0) {             // jollei alas painettu
                                        bridgedirect(3, MotorLift);
                                                                        // jarru
                                        I2Cwrite2(MotorLift, BV4206PWMWidth, 0x00, 0x00);
                                }
                                else    {
                                        bridgedirect(2, MotorLift);
                                                                        // vapaalle
                                }
                                if ( but[2] == 1)       {
                                        bridgedirect(1, MotorLift);
                                                                        // alas (taakse)
                                  hbyte= (unsigned char) (DutyMax >> 8);
                                  lbyte= (unsigned char) (DutyMax);
                                        I2Cwrite2(MotorLift, BV4206PWMWidth, hbyte, lbyte);
                                }
                                else    if ( but[1] == 0) {             // jollei ylös painettu
                                        bridgedirect(3, MotorLift);
                                                                        // jarru
                                        I2Cwrite2(MotorLift, BV4206PWMWidth, 0x00, 0x00);
                                }
                                else    {
                                        bridgedirect(2, MotorLift);
                                                                        // vapaalle
                                }
                        }
                        // etialotsoN

                }
        }
        return 0;
}

#endif
```

LIITE 9

```
/*
   (C) Copyright 2007,2008, Stephen M. Cameron.

   This file is part of wordwarvi.

   wordwarvi is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2 of the License, or
   (at your option) any later version.

   wordwarvi is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with wordwarvi; if not, write to the Free Software
   Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA

 */
#include <stdio.h>
#ifndef __JOYSTICK_H__
#define __JOYSTICK_H__

//#define JOYSTICK_DEVNAME "/dev/input/js0"

#define JS_EVENT_BUTTON         0x01    /* button pressed/released */
#define JS_EVENT_AXIS           0x02    /* joystick moved */
#define JS_EVENT_INIT           0x80    /* initial state of device */



struct js_event {
        unsigned int time;   /* event timestamp in milliseconds */
        short value;   /* value */
        unsigned char type;     /* event type */
        unsigned char number;   /* axis/button number */
};

struct wwvi_js_event {
        int button[11];
        int stick_x;
        int stick_y;
};

extern int open_joystick(char *joystick_device);
extern int read_joystick_event(struct js_event *jse);
extern void set_joystick_y_axis(int axis);
extern void set_joystick_x_axis(int axis);
extern void close_joystick();
extern int get_joystick_status(struct wwvi_js_event *wjse);

#endif
```

LIITE 10

```
#ifndef _HBRIDGE_H // tarkistetaan onko jo määritetty
#define _HBRIDGE_H


/*
        Sillan tilan asetus

        dir tilat
        0 = eteen
        1 = taakse
        2 = jarru
        3 = vapaa

        Tilanvaihdon varmistus kirjoituksen kautta palautuvasta
        tilasta joka vain 0 jos kirjoitus epaonnistunut.
        Epaonnistuessa palauttaa keskeyttaen virheilmoituksella.
*/



//*******************************************************
//Prototyypit
unsigned char bridgedirect(unsigned char dir, unsigned char Motor);
unsigned char PWMactive(unsigned char Motor);
unsigned char PWMdeactive(unsigned char Motor);
unsigned char ShutdownON(unsigned char Motor);
unsigned char ShutdownOFF(unsigned char Motor);
//*******************************************************




unsigned char bridgedirect(unsigned char dir, unsigned char Motor) {  // ebA/ebB, MotorLeft/MotorRight
        unsigned char I2Ctest;

        if (dir == 0) {

        // eteen tilanvaihdon varmistuksella
//              printf("Silta eteen\n");

         I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHR, 0x00);
                if (I2Ctest == 0) {
                                                                                                        // error
                        printf("Bridge fail addr 0x%x\n", Motor);
                        return -1;

                        // keskeyttää virheilmoituksella
         }

                I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLR, 0x00);
                if (I2Ctest == 0) {
                                                                                                        // error
                        printf("Bridge fail addr 0x%x\n", Motor);
                        return -1;
         }

         I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHF, 0x01);
                if (I2Ctest == 0) {
                                                                                                        // error
                        printf("Bridge fail addr 0x%x\n", Motor);
                I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHF, 0x00);                   // turvalliseen
tilan
                        return -1;
         }
```

```
 I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLF, 0x01);
                if (I2Ctest == 0) {
                                                                                                // error
                        printf("Bridge fail addr 0x%x\n", Motor);
                I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLF, 0x00);// turvalliseen tilaan
                        return -1;
 }

                dir = 0;
}

else if (dir == 1) {
                                                                                        // taakse
//      printf("Silta taakse\n");

I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHF, 0x00);
                if (I2Ctest == 0) {
                                                                                                // error
                        printf("Bridge fail addr 0x%x\n", Motor);
                        return -1;
                                        // keskeyttää virheilmoituksella
 }

I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLF, 0x00);
                if (I2Ctest == 0) {
                                                                                                // error
                        printf("Bridge fail addr 0x%x\n", Motor);
                        return -1;              // keskeyttää virheilmoituksella
 }

I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHR, 0x01);
                if (I2Ctest == 0) {
                                                                                                // error
                        printf("Bridge fail addr 0x%x\n", Motor);
                I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHR, 0x00);
                        return -1;              // keskeyttää virheilmoituksella
 }

 I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLR, 0x01);
                if (I2Ctest == 0) {
                                                                                                // error
                        printf("Bridge fail addr 0x%x\n", Motor);
                  I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLR, 0x00);
                        return -1;
                                        // keskeyttää virheilmoituksella
 }

                dir = 1;
}

else if (dir == 3) {

                // jarru
//      printf("Silta jarru\n");
  I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHF, 0x00);
                if (I2Ctest == 0)

                        // error
                        return -1;              // keskeyttää virheilmoituksella

  I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHR, 0x00);
                if (I2Ctest == 0)

                        // error
                        return -1;              // keskeyttää virheilmoituksella

  I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLF, 0x01);
                if (I2Ctest == 0) {

                // error
```

```
                I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLF, 0x00);
                        return -1;              // keskeyttää virheilmoituksella
        }

        I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLR, 0x01);
                if (I2Ctest == 0) {

                // error
                I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLR, 0x00);
                        return -1;              // keskeyttää virheilmoituksella
        }
                dir = 3;
        }

        else {

                                                                        // pysäksissä
//      printf("Silta nollaus\n");
        I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHF, 0x00);
                if (I2Ctest == 0)

                        // error
                        return -1;              // keskeyttää virheilmoituksella

        I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLF, 0x00);
                if (I2Ctest == 0)
                                                                                // error
                        return -1;              // keskeyttää virheilmoituksella

        I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetHR, 0x00);
                if (I2Ctest == 0)
                                                                                // error
                        return -1;              // keskeyttää virheilmoituksella

        I2Ctest = I2Cwrite2(Motor, BV4206SetPin, HbridgeFetLR, 0x00);
                if (I2Ctest == 0)
                                                                                // error
                        return -1;              // keskeyttää virheilmoituksella

                dir = 2;
        }
        return dir;
}

unsigned char PWMactive(unsigned char Motor) {

  I2Cwrite2(Motor, BV4206PWMWidth, 0x00, 0x00);
  I2Cwrite(Motor, BV4206PWMAct, 0x01);                          //PWM käyttöön
}

unsigned char PWMdeactive(unsigned char Motor) {

  I2Cwrite2(Motor, BV4206PWMWidth, 0x00, 0x00);
  I2Cwrite(Motor, BV4206PWMAct, 0x00);                          //PWM käyttöön
}

unsigned char ShutdownON(unsigned char Motor)   {
        char I2Csend = I2Cwrite2(Motor, BV4206SetPin, Shutdown, 0x01);                          //
shutdown päälle
                if ( I2Csend < 1)
                        return -1;

}
unsigned char ShutdownOFF(unsigned char Motor)  {
        char I2Csend = I2Cwrite2(Motor, BV4206SetPin, Shutdown, 0x00);                          //
shutdown pois
                if ( I2Csend < 1)

                        return -1;                                                              }

#endif
```

LIITE 11

```
/* A simple server in the internet domain using TCP
   The port number is passed as an argument */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

void error(const char *msg);
int tcpserver(int portno);

void error(const char *msg)   {
        perror(msg);
        exit(1);
}

int tcpserver(int portno)       {
//      int sockfd, newsockfd, portno;
        int sockfd, newsockfd;
        socklen_t clilen;
        char buffer[256];
        struct sockaddr_in serv_addr, cli_addr;
        int n;
        char *Osa, Komento[10], Matka[6], Teho[6];
  int Index = 0;

        PWMactive(MotorLeft);
        PWMactive(MotorRight);

        #ifdef _TEST_TCP
                printf ( "TEST_TCP: Avataan socket");
        #endif
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd < 0)
                error("ERROR opening socket");
        bzero((char *) &serv_addr, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = INADDR_ANY;
        serv_addr.sin_port = htons(portno);
        if (bind(sockfd, (struct sockaddr *) &serv_addr,
       sizeof(serv_addr)) < 0)
       error("ERROR on binding");
                listen(sockfd,5);
        clilen = sizeof(cli_addr);
        printf("\nTCP-server waiting portro: %d\n",portno);
        #ifdef _TEST_TCP
                printf ( "TEST_TCP: Uusi socket");
        #endif
        newsockfd = accept(sockfd,
         (struct sockaddr *) &cli_addr,
         &clilen);
        if (newsockfd < 0)
                error("ERROR on accept");

        int lopetus;
        printf("\nTCP-server connected portno:%d\n",portno);
```

```
		while (1) {
			bzero(buffer,256);
			n = read(newsockfd,buffer,255);
			lopetus = atoi(buffer);
			if (n < 0) error("ERROR reading from socket");
				printf("%s\n",buffer);
			n = write(newsockfd,buffer,255);
			if (n < 0) error("ERROR writing to socket");
			if (lopetus == 1)
				break;
			/* aloita jakaminen */
			Index = 0;
			strcpy(Komento,"NULL");
			strcpy(Matka,"0");
			strcpy(Teho,"80");
			Osa = strtok ( buffer, " " );
			/* iteroi niin kauan kun osia riittää */
			while ( Osa != NULL )		{
				/* kirjoita osa */
				Index++;
				#ifdef _TEST_TCP
					printf ( "TEST_TCP: Vastaanotettu %d: '%s'\n", Index, Osa );
				#endif
				if (Index == 1)
					strcpy(Komento,Osa);
				if (Index == 2)
					strcpy(Matka,Osa);
				if (Index == 3)
					strcpy(Teho,Osa);
				/* hae seuraava osa */
				Osa = strtok ( NULL, " " );
			}
			/* logo komennot */
			if ( strcmp(Komento, "foward") == 0)
				foward(atoi(Matka),atoi(Teho));
			if ( strcmp(Komento, "back") == 0)
				back(atoi(Matka),atoi(Teho));
			if ( strcmp(Komento, "right") == 0)
				right(atoi(Matka),atoi(Teho));
			if ( strcmp(Komento, "left") == 0)
				left(atoi(Matka),atoi(Teho));
			if ( strcmp(Komento, "brake") == 0)
				brake(atoi(Matka),atoi(Teho));
			if ( strcmp(Komento, "pendown") == 0)
				pendown(atoi(Matka),atoi(Teho));
			if ( strcmp(Komento, "penup") == 0)
				penup(atoi(Matka),atoi(Teho));
			if ( strcmp(Komento, "penout") == 0)
				penout(atoi(Matka),atoi(Teho));
			if ( strcmp(Komento, "penin") == 0)
				penin(atoi(Matka),atoi(Teho));
		}
		close(newsockfd);
	close(sockfd);
	return 0;
}
```

LIITE 12

```
int moottorilogo(int MoPower, int Motor);
int steps (int power);
int foward (int x, int power);
int back (int x, int power);
int right(int x, int power);
int left(int x, int power);
int brake(int x, int power);
int pendown(int x, int power);
int penup(int x, int power);
int penout(int x, int power);
int penin(int x, int power);
int reset(int x, int power);
int direction(int x, int power);


int moottorilogo(int MoPower, int Motor)    {
        if (MoPower > 0)       {
                                                                                                //
ajetaan eteen jos joystick yli 10
                if (MoPower > 100)
                                                                        // estetään ylittämästä maksimia jotta
yläfetit pysyy auki
                        MoPower = DutyMax;
                MoPower= (MoPower * DutyMax) /100;
                if (MoPower > DutyMax)
                                                                                // estetään ylittämästä
maksimia jotta yläfetit pysyy auki
                        MoPower = DutyMax;
                unsigned char hbyte= (unsigned char) (MoPower >> 8);
                unsigned char lbyte= (unsigned char) (MoPower);
                I2Cwrite2(Motor, BV4206PWMWidth, hbyte, lbyte);
        }
        else

                        // nollataan pwm
                I2Cwrite2(Motor, BV4206PWMWidth, 0x00, 0x00);

        return 0;
}

int steps (int power)   {
        int steps;
        if (power >=1 )         {                               // ei jaeta 0
                steps = power / GMAX;
        }
        else        {
                power = 0;
        }
        return steps;
}

int insan (int power)   {
        if (power > 100)
                power = 100;
        else if (power <= 0)
                power = 0;
        return power;

}


// ***************************************************
// ETEEN
int foward (int x, int power)       {                                   //eteen
        power = insan(power);
```

```
	int kiihdytys = steps(power);
	int hidastus = kiihdytys;

	#ifdef _TEST_LOGO
		printf("TEST_LOGO: Komento: foward %d %d\n", x, power);
		printf("TEST_LOGO: Kiihdytyksiä yhteensä %d\n", kiihdytys);
	#endif

	bridgedirect(0, MotorLeft);
	bridgedirect(0, MotorRight);

	while ( kiihdytys > 1) {
		#ifdef _TEST_LOGO
			printf("TEST_LOGO: kiihdytys arvoilla %d %d\n", x, ( power-(kiihdytys*GMAX)) );
		#endif
		moottorilogo( (power-(kiihdytys*GMAX)) , MotorLeft);

		moottorilogo( (power-(kiihdytys*GMAX)) , MotorRight);
		kiihdytys--;
	}

	#ifdef _TEST_LOGO
		printf("TEST_LOGO: foward arvot saavutettu\n");
	#endif
	moottorilogo(power, MotorLeft);

	moottorilogo(power, MotorRight);

	usleep(x*1000);

	while ( hidastus > 1)   {
		#ifdef _TEST_LOGO
		printf("TEST_LOGO: hidastus arvoilla %d %d\n", x, (hidastus*GMAX));
		#endif
		moottorilogo( (hidastus*GMAX) , MotorLeft);

		moottorilogo( (hidastus*GMAX) , MotorRight);
		hidastus--;
	}

	brake(x,power);
}

// ***************************************************
// TAAKSE
int back (int x, int power)		{					//taakse
	power = insan(power);
	int kiihdytys = steps(power);
	int hidastus = kiihdytys;

	#ifdef _TEST_LOGO
		printf("TEST_LOGO: Komento: back %d %d\n", x, power);
		printf("TEST_LOGO: Kiihdytyksiä yhteensä %d\n", kiihdytys);
	#endif

	bridgedirect(1, MotorLeft);
	bridgedirect(1, MotorRight);

	while ( kiihdytys > 1) {
		#ifdef _TEST_LOGO
			printf("TEST_LOGO: kiihdytys arvoilla %d %d\n", x, ( power-(kiihdytys*GMAX)) );
		#endif
		moottorilogo( (power-(kiihdytys*GMAX)) , MotorLeft);

		moottorilogo( (power-(kiihdytys*GMAX)) , MotorRight);
		kiihdytys--;
	}

	#ifdef _TEST_LOGO
		printf("TEST_LOGO: back arvot saavutettu\n");
	#endif
```

```
	usleep(x*1000);

	while ( hidastus > 1)  {
		#ifdef _TEST_LOGO
		printf("TEST_LOGO: hidastus arvoilla %d %d\n", x, (hidastus*GMAX));
		#endif
		moottorilogo( (hidastus*GMAX) , MotorLeft);

		moottorilogo( (hidastus*GMAX) , MotorRight);
		hidastus--;
	}

	brake(x,power);

}

// ***************************************************
// OIKEALLE
int right(int x, int power)		{					//oikealle
	power = insan(power);
	int kiihdytys = steps(power);
	int hidastus = kiihdytys;

	#ifdef _TEST_LOGO
		printf("TEST_LOGO: Komento: right %d %d\n", x, power);
		printf("TEST_LOGO: Kiihdytyksiä yhteensä %d\n", kiihdytys);
	#endif

	bridgedirect(0, MotorLeft);
	bridgedirect(3, MotorRight);
	moottorilogo(0, MotorRight);

	while ( kiihdytys > 1) {
		#ifdef _TEST_LOGO
			printf("TEST_LOGO: kiihdytys arvoilla %d %d\n", x, ( power-(kiihdytys*GMAX)) );
		#endif
		moottorilogo( (power-(kiihdytys*GMAX)) , MotorLeft);

		kiihdytys--;
	}

	#ifdef _TEST_LOGO
		printf("TEST_LOGO: right arvot saavutettu\n");
	#endif


	usleep(x*1000);

	while ( hidastus > 1)  {
		#ifdef _TEST_LOGO
		printf("TEST_LOGO: hidastus arvoilla %d %d\n", x, (hidastus*GMAX));
		#endif
		moottorilogo( (hidastus*GMAX) , MotorLeft);

		hidastus--;
	}

	brake(x,power);


}

// ***************************************************
// VASEMMALLE
int left(int x, int power)		{					//vasemmalle
	power = insan(power);
	int kiihdytys = steps(power);
	int hidastus = kiihdytys;

	#ifdef _TEST_LOGO
		printf("TEST_LOGO: Komento: left %d %d\n", x, power);
```

```
                printf("TEST_LOGO: Kiihdytyksiä yhteensä %d\n", kiihdytys);
        #endif


        bridgedirect(3, MotorLeft);
        bridgedirect(0, MotorRight);
        moottorilogo(0, MotorLeft);


        while ( kiihdytys > 1) {
                #ifdef _TEST_LOGO
                        printf("TEST_LOGO: kiihdytys arvoilla %d %d\n", x, ( power-(kiihdytys*GMAX)) );
                #endif
                moottorilogo( (power-(kiihdytys*GMAX)) , MotorRight);
                kiihdytys--;
        }

        #ifdef _TEST_LOGO
                printf("TEST_LOGO: right arvot saavutettu\n");
        #endif
        usleep(x*1000);

        while ( hidastus > 1)   {
                #ifdef _TEST_LOGO
                printf("TEST_LOGO: hidastus arvoilla %d %d\n", x, (hidastus*GMAX));
                #endif
                moottorilogo( (hidastus*GMAX) , MotorRight);
                hidastus--;
        }

        brake(x,power);
}


int brake(int x, int power)          {                                          //jarru
        #ifdef _TEST_LOGO
                printf("TEST_LOGO: jarrut\n");
        #endif
        bridgedirect(3, MotorLeft);
        bridgedirect(3, MotorRight);
        moottorilogo(0, MotorLeft);

        moottorilogo(0, MotorRight);
}


int pendown(int x, int power)      {                                  //nostin alas
        power = insan(power);

        #ifdef _TEST_LOGO
                printf("TEST_LOGO: Komento: pendown %d %d\n", x, power);
        #endif

        bridgedirect(0, MotorLift);
        moottorilogo(power, MotorLift);


        usleep(x*1000);

        bridgedirect(3, MotorLift);
        moottorilogo(0, MotorLift);

}


int penup(int x, int power)          {                                          //nostin ylös
        power = insan(power);

        #ifdef _TEST_LOGO
                printf("TEST_LOGO: Komento: penup %d %d\n", x, power);
        #endif
```

```
		bridgedirect(1, MotorLift);
		moottorilogo(power, MotorLift);


		usleep(x*1000);

		bridgedirect(3, MotorLift);
		moottorilogo(0, MotorLift);

}

int penout(int x, int power)	{					//kallistus eteen
		power = insan(power);

		#ifdef _TEST_LOGO
			printf("TEST_LOGO: Komento: penout %d %d\n", x, power);
		#endif

		bridgedirect(0, MotorTilt);
		moottorilogo(power, MotorTilt);

		usleep(x*1000);

		bridgedirect(3, MotorTilt);
		moottorilogo(0, MotorTilt);

}

int penin(int x, int power)	{					//kallistus taakse
		power = insan(power);

		#ifdef _TEST_LOGO
			printf("TEST_LOGO: Komento: penin %d %d\n", x, power);
		#endif

		bridgedirect(1, MotorTilt);
		moottorilogo(power, MotorTilt);

		usleep(x*1000);

		bridgedirect(3, MotorTilt);
		moottorilogo(0, MotorTilt);

}

int reset(int x, int power)	{
}


int direction(int x, int power)	{
}
```