Kim Lindholm

Sarjaliikenne protokollan suunnittelu Gerontek-nologia roboteille

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatio tekniikka

Insinöörityö

5.9.2014

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Kim Lindholm Sarjaliikenne Protokollan Suunnittelu Geronteknologia robo-teille 28 sivua + 29 liitettä 5.9.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	Automaation tietotekniikka
Ohjaaja(t)	Lehtori Timo Tuominen Prof. Yeh-Liang Hsu

Gerontechnology Research Center, GRC, suunnittelee ja kasaa geronteknologia tuotteita ja ratkaisuja Yuan Ze yliopisto Taiwanissa. Näillä tuotteilla ja ratkaisuilla on tarkoitus parantaa vanhusten elinoloja sekä luoda turvallisempi ja valvotumpi elinympäristö yksityisiin talouksiin ja vanhainkoteihin. Viime vuosien robotiikkasovellusten noste geronteknologiassa sekä kilpailun kiristymien on luonut painetta tehostaa robotiikan osaamista ja projektien hallintaa tutkimuskeskuksessa. Ajoittain vaihtuvat opiskelijat ja opiskelijoiden ohjelmointitaidon puute on aiheuttanut ristiriitaisuuksia tutkimuskeskuksen projekteissa.

Insinöörityössä kehitettiin yhtenäinen sarjaliikenne protokolla toimimaan Brain & Body verkkotopologiassa sekä ohjelmoitiin kirjasto-ohjelma Arduino-mikrokontrollerille ja Raspberry Pi-tietokoneelle. Tavoitteena oli kehittää protokolla, jota voitaisiin käyttää kaikissa tutkimuskeskuksen projekteissa ja tuotteissa ilman muutoksia alkuperäiseen protokollaan. Työssä haastateltiin eri projektien hoitajia kartoittamaan vaatimukset protokollalle. Protokollaa myös esiteltiin useita kertoja projektien hoitajille, jotta saisivat mahdollisuuden kommentoida sitä.

Protokollan kehitystyö aloitettiin vanhan protokollan päältä, josta se muodostui omaksi protokollakseen. Kirjasto-ohjelma ohjelmoitiin myös toimimaan I^2C kenttäväylällä, lähettäen kehitetyn protokollan viestejä I^2C väylää pitkin.

Työnlopuksi protokolla sekä kirjasto-ohjelma testattiin tutkimuskeskuksen vaativimmassa käyttökohteessa. Tulokset olivat positiiviset ja protokolla otettiin käyttöön tutkimuskeskuksessa välittömästi.

Avainsanat	Sarjaliikenne, Protokolla, I^2C, Arduino, Geronteknologia

Author(s) Title Number of Pages Date	Kim Lindholm Development of a serial communication protocol for gerontechnology robots 28 pages + 29 appendices 5 September 2014
Degree	Bachelor of Engineering
Degree Programme	Automation Engineering
Specialisation option	Automation IT
Instructor(s)	Yeh-Liang Hsu, Professor Timo Tuominen, Senior Lecturer

Gerontechnology Research Center, GRC, designs and builds gerontechnology products and solutions at Yuan Ze University, Taiwan. Their work is meant to increase the life quality of an elder and create a more secure and supervised habitat for private households and retirement homes. In recent years, the increase of robotics in the field of gerontechnology and intensification of competition has caused pressure to enhance the knowledge of robotics and project management in the research center. Intermittently changing students and student's lack of skill in programming has caused contradictions in the research center's projects

In this thesis a uniform serial communication protocol was developed to work in Brain & Body- network topology and also a library program for Arduino microcontroller and Raspberry Pi computer was programmed. The goal was to develop a protocol that could be used in all of the research center's projects and products without making any changes to original protocol. In this thesis, project managers were interviewed to identify the protocol's requirements. The protocol was also presented many times during the development to project managers so they could give comments about the protocol.

The development of the protocol started from the basis of an old protocol from which it was developed into its own protocol. The library program was also programmed to work in I^2C fieldbus sending developed protocols messages along the I^2C bus

At the end of the final project the protocol and library program was tested on the research center's most demanding application. The results were positive and the protocol was introduced immediately at the research center.

Keywords	Serial, Protocol, I^2C, Arduino, Gerontechnology

Sisällys

Lyhenteet

GRC	Gerontechnology Research Center. Yuan Ze yliopiston alla toimiva tutkimuskeskus jolle työ tehtiin.
UART	Universal Synchronous Receiver/Transmitter. Sarjaliikennepiiri joka muuntaa tietoliikenneportista tulleen sarjamuotoisen datan rinnakkaismuotoiseksi.
I^2C	Inter-Integrated Circuit. I^2C on Philipsin kehittelemä yksinkertainen kaksisuuntainen ohjaus- ja tiedonsiirtoväylä.
RS-232	Recommended Standard 232. RS-232 on kahden tietokonelaitteen väliseen tietoliikenteeseen tarkoitettu tietoliikenneportti
TTL	Transistor-transistor logic. Logiikkapiiriperhe joka on levinnyt moniin elektroniikka laitteisiin kuten tietokoneisiin ja kontrollereihin.
CRC	Cyclic redudancy check. CRC on tarkistesumma laskennassa käytetty laskenta tapa.
GPIO	General Purpose Input Output. Yleinen ohjausportti. Pystyy yleensä kirjoittamaan ja lukemaan digitaalisia ja analogisia viestejä.

1 Johdanto

1.1 Työntausta

Yuan Ze yliopiston alla toimivan GRC-tutkimuskeskus suunnittelee ja rakentaa prototyyppi tuotteita vanhuksille ja vanhusten hoitoon. Nämä tuotteet ovat monesti paremmin mekaanisesti toteutettuja kuin ohjelmallisesti taikka sähköisesti. Tutkimuskeskuksessa on menossa muutosvaihe, jossa kaikki eri kehityksen alla olevat tuotteet muutettaisiin toimimaan saman järjestelmän alla. Tästä syntyi myös ajatus yhtenäisestä kenttäväylän tiedonsiirto protokollasta kahden laitteen välillä.

GRC on aikaisemmin suunnitellut ja toteuttanut kenttäväylän tiedonsiirtoprotokollan, mutta protokolla ei ole ollut yhtenäinen, vaan sitä on jouduttu muokkaamaan joka kerta, kun on uusi tuote taikka laite suunniteltu.

1.2 Työn tavoitteet ja toteutus

Insinöörityön tarkoituksena on pohtia vanhan kenttäväylä protokollan puutteita ja suunnitella ja määritellä uusi protokolla, jota tulisi pystyä käyttämään monissa jo valmiissa tuotteissa sekä tulevissa kohteissa. Tehtävänä on myös kirjoittaa Arduino-mikrokontrollerille kirjasto ohjelma, joka lukee ja toimii protokollan käskyjen mukaisesti. Kirjasto ohjelman pitäisi pystyä lukemaan protokollan viestit UART-sarjaliikenne piiristä kuin myös I^2C kenttäväylältä.

Työ toteutetaan yhteistyössä iRW projektin toteutuksen kanssa, sillä iRW projekti antaa tällä hetkellä haastavimman ympäristön kenttäväylä protokollalle, joten se sopii erinomaisesti testi ympäristöksi.

1.3 Rajaukset

Työn painottuu vain ohjelmalliseen työhön. Valitut sarjaportti liitynnät sekä protokollan ominaisuudet tehdään tutkimuskeskuksen ja Arduino mikrokontrollerien ehdoilla.

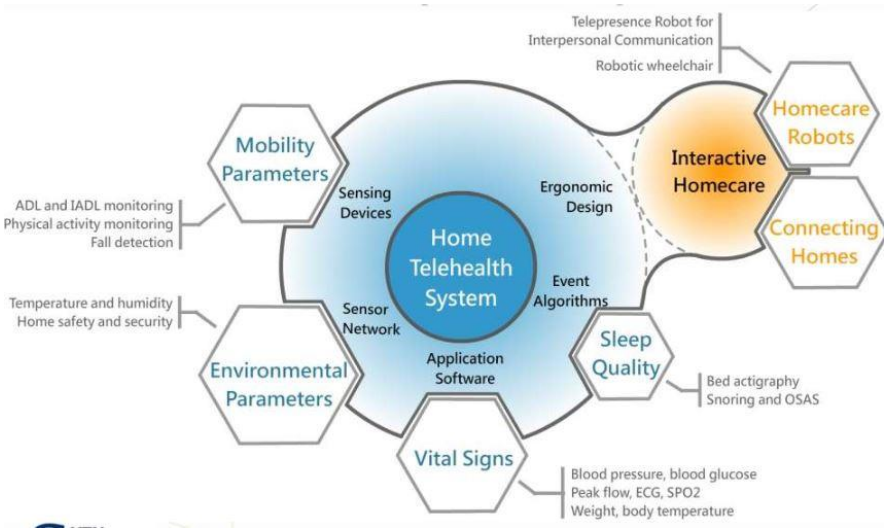
2 Gerontechnology Research Center

Gerontechnology Research Center suunnittelee ja kehittää prototyyppi tuotteita vanhusten hoitoon ja hyvinvoinnin edistämiseen. GRC työllistää 20, henkeä joista suurin osa on Yuan Ze yliopiston opiskelijoita. GRC on perustettu vuonna 2003 ja oli Taiwanin ensimmäinen geronteknologiaan erikoistunut tutkimuslaitos.[4]

GRC:n pääsuunnittelu kohde on älykoti jossa, vanhuksen on turvallinen ja mukava asua. GRC:n ajatuksena on, ettei vanhuksen tarvitsisi muuttaa pois kotoaan, vaan älykkäät laitteet ja anturit tuotaisiin heidän koteihinsa. Tästä on kehitetty ajatus sänky-keskeisestä kodista, jossa vanhemman ihmisen terveyden valvonta ja mukavuuden lisäys rakennetaan sängyn ympärille. Kuvassa 1. on esitetty GRC-tutkimuskeskuksen ajatus tulevaisuuden kotihoidosta ikääntyneille ihmisille.[4]

Etäläsnäolo robotit ovat myös suuressa nosteessa kehitystyössä. Näillä roboteilla voimme kommunikoida vanhuksen kanssa hänen ollessaan kotonaan ja samalla valvoa heidän asuntoaan.[4]

GRC tekee paljon yhteistyötä paikallisten yritysten kanssa tuodakseen kehittämiään tuotteita markkinoille ja kaikille saatavaksi. Osa tuotteista on jo kaupallistettu, kuten WhizPad, älykäs patja, jolla voidaan valvoa ja seurata ikääntyneen ihmisen unenlaatua.[4]



Kuva 1. GRC:n tutkimus alueet

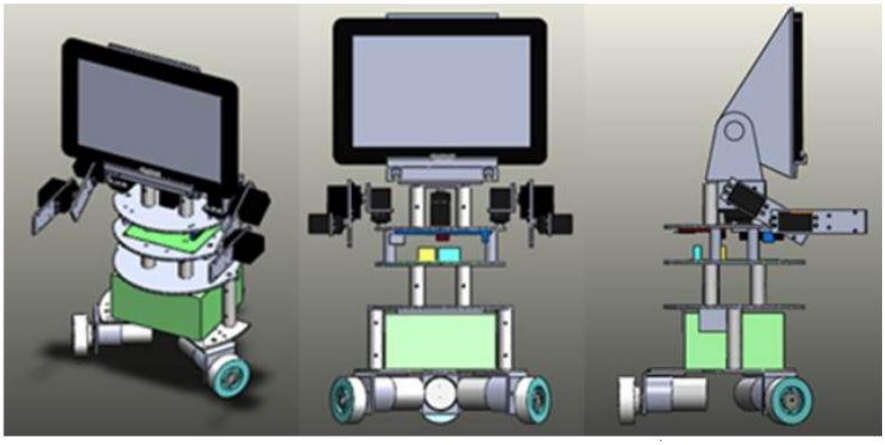
3 Protokollan käyttökohteet

Protokollalle tulee mahdollisesti olemaan paljon eri käyttökohteita GRC:n tutkimuskeskuksessa. GRC:n kehittämät robotit iRW, TricMini+, WoBot ja WoBot Jr on esitelty tarkemmin seuraavissa kappaleissa. Muita mahdollisia käyttökohteita on WhizPad, älypatja, jolla voidaan mitata unen laatua sekä WhizCarpet, älymatto, jolla voidaan tarkkailla vanhuksen päivittäistä aktiivisuutta asunnossaan. Jokainen näistä käyttökohteista on erilainen, mikä asettaa vaatimuksia protokollan suunnittelulle.

3.1 TricMini+, WoBot ja WoBot Jr.

TricMini+ on kolmannen sukupolven TricMini-robotti. TricMini on suunniteltu etäläsnäolo robotiksi ihmisten väliseen kommunikointiin. TricMinillä ikääntynyt henkilö voi jakaa elämän tapahtumiaan perheensä kesken, vaikka he eivät olisi fyysisesti paikalla. TricMinissä on pyritty yhdistämään Skypen kuuluisaksi tuomat videopuhelut, mutta myös ihmismäinen liike ja tunteiden näyttäminen.[2]

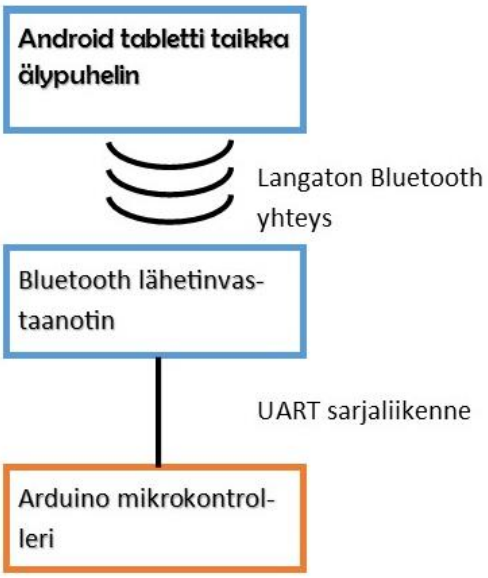
WoBot ja WoBot Jr. ovat TricMinin tavoin etäläsnäolorobotteja mutta ne erottuvat ulkoiselta rakenteeltaan TricMinistä. TricMinissä on kahdeksan servomoottoria liikuttamaan käsiä ja päätä ja kolme moottoria liikuttamaan omni-pyöriä, joten TricMinissä on yhteensä 11 vapausastetta. Kaikkia näitä vapausasteita pitää voida ohjata samanaikaisesti, mikä luo aikarajoitteen protokollan lukemiseen. WoBot sarjan robotit on rakennettu yksinkertaisemmiksi, mutta silti ne kykenevät liikkumaan ja elävöitymään keskusteluun. WoBotissa on kolme vapausastetta ja WoBot Jr. on viisi vapausastetta.



Kuva 2. 3D suunnitelma TricMini+:sta.

TricMini+, WoBot ja WoBot Jr. on rakennettu ”Brain & Body” – rakenteella, jossa kaikki päätöksenteko ja ohjaus tapahtuu ”aivoissa” ja ”keho” vain tottelee aivoista tulevia käskyjä. TricMinin ja WoBotin aivoina toimii Android käyttöjärjestelmällä toimiva tabletti tietokone, joka myös yhdistää puhelut toiseen tablettitietokoneeseen. WobBot Jr käyttää aivoinaan Android-käyttöjärjestelmällä toimivaa puhelinta. Aivot kommunikoi kehon kanssa käyttäen Bluetooth yhteyttä, tekemällä aivojen vaihtamisesta vaivatonta.[3]

Tässä työssä kehitettyä tiedonsiirtoprotokollaa tullaan käyttämään Bluetoothin ja UART sarjaliikenne piirin yli. Android tabletti taikka älypuhelin lähettää protokollan mukaista viestiä. Robotissa on Bluetooth lähetinvastaanotin, joka muuntaa Bluetooth viestin UART sarjaliikenteeksi, jonka Arduino mikrokontrolleri lukee ja reagoi aivojen haluamalla tavalla. Kuvassa 3. on esitetty kyseinen tiedonsiirron topologia.



Kuva 3. Tiedonsiirron topologiakuva

3.2 iRW

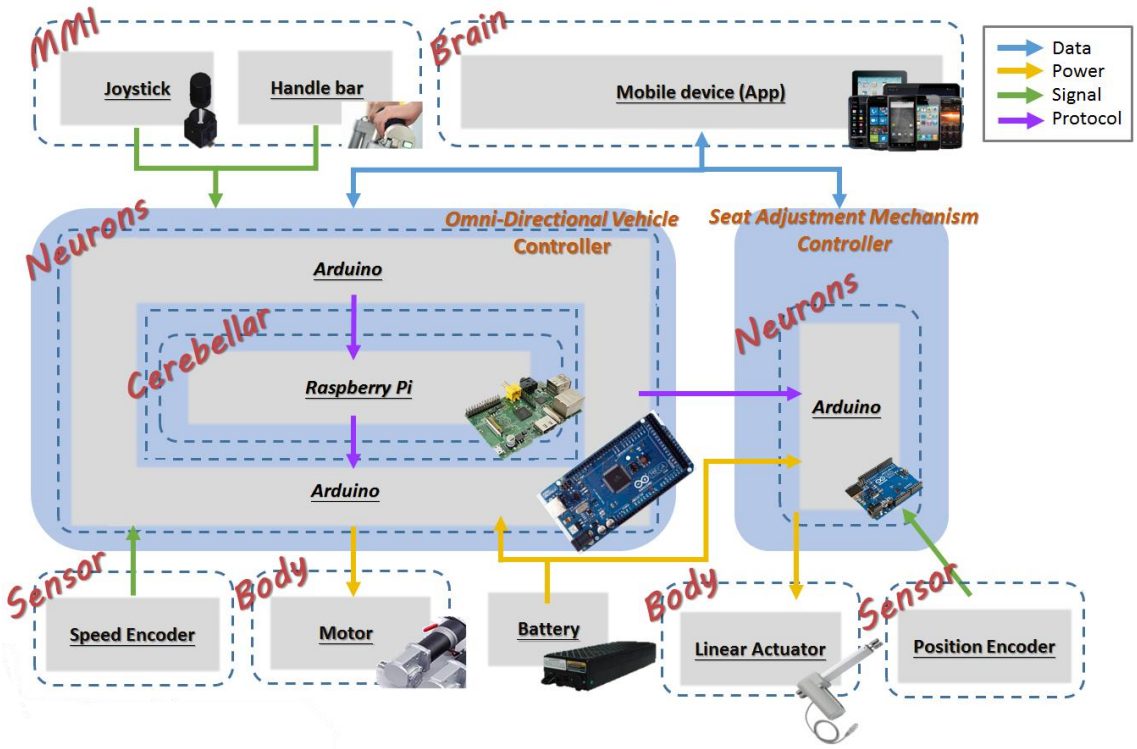
iRW on GRC:n kehittämä robotoitu pyörätuoli. Pyörätuolilla voi liikkua normaalista sähköpyörätuolista poiketen myös sivuttain sekä tehdä 90 asteen käännöksiä paikallaan. Toinen merkittävä ero iRW ja tavallisen sähköpyörätuolin välillä on sen istuin. iRW:n istuin pystyy nousemaan, laskemaan ja kippaamaan. Näin istuimelle istuminen ja nouseminen on mukavampaa ja helpompaa kuin se normaalissa sähköpyörätuolissa olisi.[1]

iRW voidaan ohjata kolmella eritavalla. Ensimmäisenä on perinteinen ohjaussauva jolla voidaan ohjata pyörätuolia ohjaussauvan osoittamaan suuntaan. Toinen ohjaustapa on käyttämällä käsitankoa. Käsitankoon on sijoitettu neljä paine-anturia mittaamaan käsien puristusta ja käsien sijaintia. Mitä kovemmin työntävä henkilö puristaa käsitankoa, sitä enemmän moottorille siirretään energiaa. Kolmas ohjaustapa on käyttää tabletti tietokonetta ohjaamaan. Tabletti tietokoneesta voidaan määrittää mihin huoneeseen halutaan, jolloin tabletti tietokone ohjaa pyörätuolin tähän huoneeseen. Myös manuaalinen nuolinäppäin ohjaustapa on mahdollinen tabletti tietokoneessa. Tabletti tietokoneella myös ohjataan istuimen asentoa haluttuun asentoon.[1]

iRW ei ole rakennettu muista roboteista poiketen ”Brain & Body”-rakenteella, vaan siinä on käytetty ”Brain – Cerebellar – Body”- rakennetta. Tässä rakenteessa aivoina toimii vieläkin tablettitietokone, mutta mikrokontrollerien ja aivojen väliin on laitettu ”pikkuaivot”, jotka käsittelevät suurimman osan datasta ja tekevät suurimman osan päätöksen teosta. Aivojen komennot on kuitenkin priorisoitu muiden komentojen yli. Pikkuaivoina iRW:ssa käytettiin Raspberry Pi Foundationin kehittämää Raspberry Pi tietokonetta.[1]

iRW:ssa protokollaa käytetään pikkuaivojen ja Arduino-mikrokontrollerien välisessä tiedonsiirrossa. Koska iRW käyttää useampaa kuin yhtä mikrokontrolleria, ei voitu enää käyttää UART-sarjaliikennettä, kuten esimerkiksi WoBot ja TricMini käytti. iRW:ssa käytettiin I^2C tiedonsiirto väylää, joka mahdollistaa yhden taikka useamman laitteen liitettävyyden väylään.[1]

iRW käytettiin protokollan testaamiseen, koska se antoi kaikkein haastavimman ja monipuoleisimman käyttöalustan protokollalle. iRW on sijoitettu yhteensä seitsemän Arduino-mikrokontrolleria toimimaan mittauksissa ja moottorien ohjauksissa sekä Raspberry Pi ohjaamaan näitä. Arduinot keskustelee keskenään I^2C väylää pitkin rajapinta Arduinon toimiessa isäntänä muille Arduinoille ja orjana Raspberry Piille. Rajapinta Arduino käyttää UART sarjapiiriä keskustellessaan Raspberry Piin kanssa koska Rasberry Piin SMBus ei keskustellut luetettavasti I^2C:n kanssa. Kummassakin tiedonsiirrossa, yli I^2C:n ja UART:n, käytettiin protokollaa datan siirtämiseen. Kuvaan 4 on piirretty iRW datan ja signaalien siirto topologia.[1]



Kuva 4. iRW signaalien ja tiedonsiirron topologiakuva. Protokollan tiedonsiirtoliikenne on merkattu tumman sinisellä

4 Sarjaliikennepiirit ja väylät

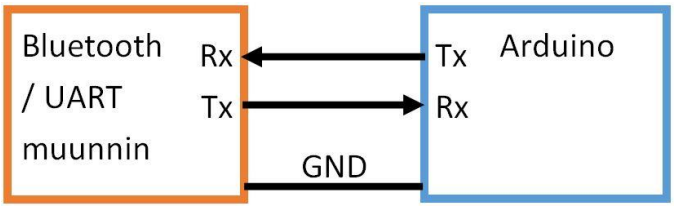
Tässä työssä suunnitellun protokollan piti olla mahdollista käyttää UART sarjaliikennepiirin yli, sekä luoda ohjelma jolla voitaisiin protokollassa luokiteltuja viestejä vastaanottamaan myös I^2C:n tiedonsiirtoväylältä. I2C:ssä on jo itsessään oma tiedonsiirto protokolla. Tässä luvussa esitellään nämä kaksi toisistaan eroavaa tiedonsiirron muotoa.

4.1 UART

UART:lla tarkoitetaan universaalia, ei reaaliaikaista, lähetin vastaanotinta. UART:n mikropiiri purkaa tavussa olevat rinnakkaismuotoiset bitit sarjamuotoiseksi ja lähettää ne sarjaliikennettä pitkin vähemmän merkitsevästä bitistä enemmän merkitsevään vastaanottavalle laitteelle. Lukiessaan viestiä sarjaliikenne piiriltä UART toimii päinvastoin. Ennen varsinaisten bittien lähettämistä, UART lähettää aloitus bitin ilmoittamaan tulevasta viestistä. Viesti bittien jälkeen tulee lopetus bitti, joka lopettaa kommunikaation. Arduino-mikrokontrollereissa sarjaliikenne kommunikointi on toteutettu ainoastaan käyttämällä UART-kommunikointia. UART:ssa ei ole isäntää eikä orjaa, vaan molemmat

osapuolet ovat tasavertaisia kommunikoinnissa. Tällöin kellotaajuus ei tule sarjaliikenne piiriä pitkin, vaan se asetetaan ohjelmassa, joka käynnistää UART kommunikoinnin. Perinteisiä sarjaliikenteen nopeuksia ovat 300 bit/s, 600 bit/s, 1 200 bit/s, 2 400 bit/s, 4 800 bit/s, 9 600 bit/s, 19 200 bit/s ja 38 400 bit/s.[5,6]

UART:ssa ja sarjaliikenne piireissä yleensäkin lähetetyt ja vastaanotetut bitit verrataan maatasoa vasten. Yleisimpiä kytkentä tapoja ovat RS-232, RS-485 ja TTL. Arduinossa käytetään +5V TTL kytkentää bitin lähetyksessä ja lukemisessa. +5V TTL kytkennässä vastaanotettu +5V signaali luetaan ykköseksi ja 0V signaali nollaksi. RS-232:sessa tilanne on hiukan erilainen. Siinä kytkentä on toteutettu +-5V vertailulla, jossa +5V on nolla ja -5V on ykkönen. TTL kytkennässä laitteet kytketään kolmella johtimella toisiinsa. Tarvittavat liitännät ovat esitetty kuvassa 5. Tx tarkoittaa lähettävää porttia ja Rx vastaan ottavaa. GND on yhteinen maajohdin, johon bittejä verrataan. [5,6]



Kuva 5. Esimerkki TTL kytkennästä.

4.2 I²C

I²C on Philipsin kehittämä ja hallinnoima kaksisuuntainen ohjaus- ja tiedonsiirtoväylä kuluttaja elektroniikan tarkoituksiin. I²C:n ensimmäinen versio julkaistiin 1982 ja on jatkuvasti kehittynyt tähän päivään saakka. Se onkin yleistynyt monitorien kytkentä kaapeleissa kuten VGA ja HDMI. I²C:stä on kehitetty monia muita väylätekniikoita kuten System Management Bus (SMBus) ja PowerManagement Bus (PMBus) käyttämällä samaa arkkitehtuuria kuin I²C:ssä, mutta lisäämällä väylän vaatimuksia.[7,8]

I²C on aidolla moni-isäntä-orja periaatteella toimiva väylä, mikä tarkoittaa sitä, että väylällä voi olla samaan aikaan toiminnassa monta eri isäntää ja orja laitetta. Isännät myös pystyvät pitämään toisia isäntiä orjinaan. Välttääkseen samaan aikaan tiedon lähettämisen, isäntä varaa koko väylän näin estäen muita kommunikoimasta. Viesti alkaa START ilmoituksella, joka varaa väylän isännän käyttöön. Isäntä keskustelee suoraa orjalle kutsumalla orjan osoitenumeroa. Viestin loppuessa tulee STOP ilmoitus, jonka

jälkeen väylä on vapaa. I^2C väylässä käytetään kahdenlaisia viestejä, kirjoitusta ja lukua. Kirjoitus viestillä isäntä vain kirjoittaa tiedon orjalle, johon orja vastaa ymmärtävänsä. Luku viesteillä isäntä pyytää orjaa lähettämään tieyty määrä tavuja vastauksesi. [7,8]

Fyysiseltä rakenteeltaan I^2C koostuu kolmesta johdosta. SDA, SCL ja GND. GND on tietenkin maajohto, jota käytetään tiedon ja kellon tiedon vertailussa. SDAta pitkin tieto kulkee molempiin suuntiin ja SCL toimii väylän kellona. Isäntä aloittaa kellon kirjoittamisen keskustelu yhteyden avautuessa. Jännite I2C väylässä on yleensä joko 3.3V taikka 5V. Jännitteen suuruus määräytyy isännän mukaan, joten 3.3V väylän voi kytkeä 5V väylään, jos isäntä käyttää 3.3V. [7,8]

5 Vanha Protokolla

5.1 Rakenne

GRC:n tutkimuskeskus on kehittänyt aikaisemmin protokollan käytettäväksi roboteissaan, jotta ne saataisiin toimimaan ’Brain & Body’-rakenteella. Protokolla on jaettavissa viiteen osaan:

- Aloitus
- Komentojen lukumäärä
- Komennot
- Viive
- Lopetus

Protokollan rakenne on esiteltynä kuvassa 6. Jokainen merkki protokollassa lähetettiin ASCII-merkistön mukaisina tavuina, joten jokainen numero ja kirjain vei yhden tavun verran tilaa. Iso S-kirjain ilmoitti orjalaitteelle kommunikoinnin alkamisen. Sitä seuraa luku, joka ilmoittaa komentojen määrän. Kirjaimet erottavat komennot toisistaan. Isolla

A-kirjaimella halutaan vaikuttaa mikrokontrollerissa vastaavaksi nimettyyn kenttälaitteeseen. Perässä tulevat kahdeksan numeroa ovat tiedot, jota halutaan laitteelle kirjoittaa. Numerot jaetaan kahteen osaan, jolloin molempiin ryhmiin jää neljä numeroa. Näin ollen, jos A-laite olisi ollut servomoottori, olisi nopeudeksi kirjoitettu 100 ja positioksi 160. B-laitteen komento luetaan vastaavalla tavalla. Viiveellä tarkoitetaan aikaa, jonka orja odottaa viestin loppumisen jälkeen ennen kuin lähettää OK signaalin takaisin isännälle. Viiveen maksimi aika on 9 999 ms. Moottorin ajoaika on aika, jonka moottorit ajaa kunnes sammuvat. Viiveen aika lasketaan vasta moottorin ajoajan jälkeen.

Kuva 6. Vanhan protokollan rakenne

5.2 Puutteet

Edellä esitellyssä protokollassa todettiin puutteita nopeasti sen käyttöönoton jälkeen, mutta niihin ei riittänyt aikaa puuttua, joten protokolla vakiinnutti paikkansa tällaisenaan. Kuitenkin kun ongelmat alkoivat ilmentyä tuli selväksi, että uusi protokolla piti suunnitella.

Merkittävimpiä puutteita vanhalle protokollalle on häiriön havaitseminen. GRC:n robottien luettaessa vahvasti Bluetooth- ja UART-tiedonsiirtoon tulee häiriöherkkyys erityisesti esille. Esimerkiksi TricMini+ sisältää kahdeksan servomoottoria ja kolme tasajännite moottoria, joita kaikkia pitää voida ohjata samaan aikaan, on taustakohinasta syntyneen häiriön riski kohtalainen.

Myös protokollan koko on ongelmallinen. Ilman ensimmäistäkään komentoa protokolla vie jo 11 tavun verran tilaa. Arduino-mikrokontrollerin UART-puskurin enimmäiskoko on

64 tavua, joten komennoille jää 53 tavua tilaa. Yksi komento vie yhdeksän tavun tilan joten vain viisi komentoa voidaan lähettää yhdessä paketissa.

Ongelmana oli myös protokollan monet muodot. Kuvan 6 esimerkki protokollasta on vain yksi protokollan muodoista. Joissain käyttökohteissa komennoissa ei ollut kahdeksaa numeroa vaan kuusi, kun robotissa olevat kenttälaitteet eivät vaatineet nelinumeroista käskyä. Moottoriviive aika tippui pois jossain protokollan versioissa kun, käyttökohteella ei ollut moottoreita.

6 Vaatimukset

Ennen kuin protokollaa alettiin suunnittelemaan, asetettiin sille tietyt vaatimukset ja tavoitteet, jotka sen pitää olla mahdollista toteuttaa. Kaikki näistä vaatimuksista oli kriittisiä saavuttaa paitsi Porttiasetusten tallentaminen mikrokontrolleriin. Myös tarvittavasta tarkkuudesta käytiin keskustelua.

6.1 ’Brain & Body’– ja ‘Brain– Cerebellar– Body- rakenne

Tärkein vaatimus ja perusta koko projektille on protokollan toteuttaminen ’Brain & Body’– ja ‘Brain– Cerebellar– Body’- rakenteen mukaisesti. Brain & Body– ja ‘Brain– Cerebellar– Body’-rakenne ei suoranaisesti tuo protokollalle erityisiä lisävaatimuksia, vaan vaikuttaa käyttökohteen fyysiseen sekä sähköiseen suunnitteluun, jotka vaikuttavat protokollan suunnitteluun.

6.2 Aikakriittisyys

Ohjatessa laitetta, jonka liikkeiden ja liikeratojen pitäisi näyttää mahdollisimman ihmismäiseltä ja aidolta, nousee tärkeäksi vaatimukseksi aika. Protokolla pitäisi olla mahdollisimman nopea ohjelman lukea, jotta komennot saataisiin näyttämään kuin ne olisi käynnistynyt samaan aikaan.

6.3 Taustakohinasta syntyneiden virheiden havainnointi

UART:ssa ja Bluetoothin langattomassa tiedonsiirrossa on kohtalainen riski syntyä häiriöitä viestiin. Yleisimpinä aiheuttajille näille häiriöille voidaan pitää servo- ja tasajännitemoottorien aiheuttamaa taustakohinaa. Mahdollinen virhe voi aiheuttaa suuria virheitä robotin liikkumisessa, kun UART-linjaa kulkeva nolla arvo muuttuu ykköseksi. Täten on tärkeätä että mahdolliset virheet saadaan karsittua pois.

6.4 Porttiasetusten tallentaminen mikrokontrolleriin

Porttiasetusten tallentamisesta mikrokontrolleriin esitettiin toive aloituskokouksessa. Jos tämä olisi mahdollista, voisi mikrokontrolleria ohjelmoida uudestaan tabletin taikka älypuhelimen avulla, eikä ohjelmaa tarvittaisi aina muuttaa uudestaan. Ajatus tähän ideaa tulee kuluttaja elektroniikan puolelta. Tutkimuskeskuksen yhtenä ideana on kehittää alusta, josta käyttäjä voisi rakentaa oman etäläsnäolo robotin näin saavuttaen paremman vastaanoton roboteille.

6.5 Kirjoitettavan arvon koko

Kirjoitettavan arvon koolla tarkoitetaan sitä kuinka tarkka arvo voidaan kirjoittaa mikrokontrollerille kommunikoimalla sarjaliikenne väylän yli. Viestin kirjoitettavan arvon koko on tärkeä vaatimus protokollalle ja haluttiinkin saavuttaa suurin mahdollinen arvo mitä voitaisiin saavuttaa. Alkuperäisessä protokollassa arvo oli 0 – 9 999, jota ei haluta laskea pienemmäksi. Yhden tavun antaessa vain maksimi arvon 0 - 255 tuo edellä mainittu vaatimus haastetta protokollalle.

6.6 Omat funktiot

Toisinaan tarvitaan mikrokontrolleriin tehdä omia toimintoja, joita olisi mahdotonta toteuttaa protokollan käskyillä. Tätä varten pitää jättää mahdollisuus käyttäjän omien funktioiden kutsulle, joita pitää pystyä muokkaamaan mieluisekseen.

7 USIP Protokolla

7.1 Rakenne

Protokollan rakennetta suunnitellessa päädyimme vaihtamaan edellisessä protokollassa olleen moni komennon yksittäiskomento rakenteeseen. Näin saadaan yksinkertaistettua kirjasto ohjelmaa sekä lyhennettyä viestin kokoa. Protokollan rakenne jakautuu viiteen tavuun. Toisinkuin vanhassa protokollassa, nykyisessä ei ole aloitus tavua taikka bittiä. Tähän päädyttiin, koska muuten olisi jouduttu lisäämään niin sanottuja ESC tavuja viestin väliin aina kun tavun arvo on sama kuin aloitus tai lopetus tavun arvo.

Kun aloitus ja lopetus eivät ole käytössä, on ohjelman joka lukee viestiä, odotettava että puskuriin on tullut viisi tavua ennen kuin reagoi viestiin. Ensimmäisen tavun saavuttua on hyvä ottaa aikaa, joka menee muiden tavujen saapumiseen. Jos aika ylittää tietyn rajan, voidaan päätellä, ettei seuraavan tuleva tavu välttämättä ole enää samasta viestistä. Tämmöiseksi ajaksi pääteltiin 10 ms, kun yhden tavun kirjoittaminen vie 1ms.

Kuvasta 7 voi nähdä protokollan rakenteen graafisessa muodossa. Jokainen laatikko on yksi tavu viestissä. Ensimmäinen tavu on Osoitetavu. Osoitetavu ilmoittaa mitä halutaan ohjata taikka mitata. Toinen tavu on tieto- taikka osoitintavu. Tämän tavun funktio vaihtuu sen mukaan, mikä oli osoitetavun arvo. Osoite tavun ollessa muu kuin porttiohjaus on kakkostavussa ohjaustietoa. Kolmas ja neljäs tavu on vain tuomaan ohjaustietoja orjalaitteelle. Viides tavu on CRC tavu eli tarkistesummatavu. Ohjelma vertaa tätä tavua vasten tutkiessaan viestin eheyttä.

Kuva 7. USIP protokollan viesti.

Joissain toiminnoissa tarvitaan suurempaa lukua kuin 255, joka on tavun maksimiarvo. Näitä toimintoja on esimerkiksi servomoottorin kulma/millisekunti tieto taikka funktioiden ohjaus arvot. Tällöin kaksi taikka useampi tavu yhdistetään toisiinsa mahdollistaen suuremman ohjausarvon mikrokontrollerille. Tavut kuitenkin lähetetään yksitellen protokollan mukaisesti.

7.2 Osoite numeristo

Ensimmäinen tavu ilmoittaa osoitteen taikka portin, jota halutaan ohjata. Koska tutkimuskeskus käyttää ainoastaan Arduinon-mikrokontrollereita protokolla suunniteltiin paljolti niiden ympärille. Tällä hetkellä suurin mikrokontrolleri Arduino Mega 2560 sisältää 54 GPIO porttia ja 16 analogista porttia joiden mukaan protokollan osoite numeristo on rakennettu.

Numeristo jakautuu siten että osoite tavun numero 1 – 70 viittaa porttinumeroon vähentämällä osoite numerosta yhden saadakseen halutun porttinumeron. Esimerkiksi osoitetavu 10 tarkoittaa porttinumero yhdeksään tulevaa käskyä. Osoite numeristo alkaa ykkösestä koska ei haluttu että nolla voisi olla ensimmäisenä lukuna, mutta Arduino-mikrokontrollerien portti numerointi alkaa nollasta. Tämä toimii kaikille muille perinteisille käskyn luku ja kirjoitusmuodoille paitsi analogian lukemiselle. Arduino-mikrokontrollereissa on erilliset analogia portit joilla ainoastaan voidaan lukea analogista 0-5V jänniteviestiä. Näihin portteihin päästääkseen kiinni on osoite numeron oltava haluttu analogia portti johon on lisätty arvo 55.

Osoite numerot 71 – 124 ovat servomoottori ohjauksia. Haluttuun porttiin pääsee ohjaamaan kun kirjoittaa halutun porttinumeron ja lisää siihen 71. Analogia porteilla ei ole mahdollista ohjata Servomoottoreita joten vain perinteisillä GPIO portit ovat mukana osoite numerostossa.

Osoite numerot 125 – 255 ovat käytössä käyttäjän omille funktioille.

7.3 Analoginen ja digitaalinen kirjoitus ja luku

Osoitetavun ollessa 1 – 70 välillä Arduino-mikrokontrollerille kerrotaan halusta ohjata taikka lukea portin arvoa. Osoitetavun kertoessa portin numero, toisena tuleva tavu kertoo mitä portille halutaan tehdä. Jos toisen tavun arvo on 10, halutaan lukea portin digitaalinen arvo. Jos arvo on 20, halutaan kirjoittaa digitaalinen arvo. 30 on analogia arvon lukeminen ja 40 analogia arvon kirjoittaminen.

Kun portille kirjoitetaan uusi arvo, arvo saadaan kolmannesta tavusta. Digitaalisessa kirjoittamisessa arvo 16 kirjoittaa portin arvoon 0 ja 17 arvoon 1. Muut arvot eivät vaiku-

ta portin arvoon. Kirjoittaessa analogia-arvoa, tulee kolmannen tavun arvo suoraan kirjoitettavaksi analogia-arvoksi.

7.4 Servo-ohjain

Osoite tavun arvot 71 – 124 ovat servomoottori ohjauksia. Halutun portin saa kutsuttua kun lisää portin numeroon luvun 71. Ohjatessa servomoottoreita, toinen tavu on ohjattavan servomoottorin nopeus. Yhteenlaskettuna, kolmas ja neljäs tavu ovat kulma taikka luvun ollessa yli 512, se muuttuu ajoajaksi millisekunteina.

Yleensä servomoottorit pitää yhdistää ohjelman alussa jotta servomoottoriohjain osaa ohjata oikeata servoa käskyn tullessa. Arduinolle tehdyssä kirjasto-ohjelmassa tämä yhdistäminen on tehty automaattiseksi. Protokollan lähettäessä ensimmäisen kerran servomoottori ohjauskäskyn ohjelma yhdistää tämän portin servomoottoriohjaimeen. Ohjelma myös tallentaa servomoottorin asennon ja ei pyri liikkumaan jos tuleva ohjauskäsky on sama kuin nykyinen tila.

7.5 Vastaaminen

Isäntä laitteen kysyessä mittaustulosta taikka halutessaan arvon funktiolta, on mikrokontrollerin vastausviesti oltava protokollan mukainen. Vastaus alkaa samalla osoitenumerolla kuin alun perin isäntä kysyi. Toinen, kolmas ja neljäs tavu ovat varattu vastaus viestille mitättömin tavu ollessa toisena, toiseksi arvokkain kolmantena ja arvokkain tavu neljäntenä. Viides tavu on mikrokontrollerin luoma CRC varmenne.

7.6 Käyttäjän omat funktiot

Käyttäjä voi määritellä yli sata omaa funktiota tällä hetkellä. Osoitenumerot 125 – 255 toimivat funktioiden kutsumisessa. Käyttäjä ohjelmoi funktiot Arduinon ohjelmointityökalulla. Funktiot toimivat kuten normaali ohjelma ja voivat sisältää mitä tahansa C++/C# kielissä hyväksyttyjä toimintoja. Funktiot voivat myös vastata isännälle jos ne ohjelmoidaan niin.

7.7 Porttiasetusten tallennus

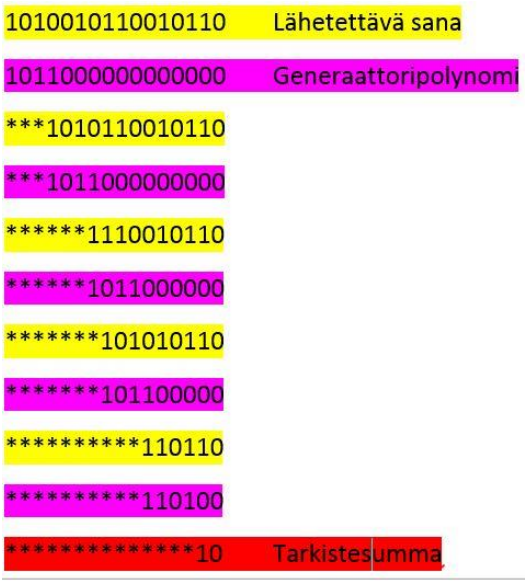
Arduino-mikrokontrollereissa portin normaali asento on tulo ja se voidaan ohjelmallisesti muuttamaan lähdöksi käyttämällä pinmode komentoa. Osoitenumeron kutsuessa suoraan porttinumeroita, toisella tavulla voidaan mikrokontrollerille kertoa portinasetusten vaihdosta. Toisen tavun arvon ollessa 11 taikka 31 portin asetukseksi tallennetaan tulo, numeroiden ollessa 21 taikka 41 tallentuu ne lähdöiksi. Kolmatta ja neljättä tavua ei käsitellä kun portin asetusta vaihdetaan.

7.8 CRC

CRC on tärkeimpiä uudistuksia vanhaan protokollaan nähden. Sillä saadaan varmistettua että viesti on luettaessa se sama viesti kuin se oli sitä lähetettäessä. CRC:llä tarkoitetaan cyclic redudancy checkiä eli syklistä tarkistuskoodia. CRC:n periaate on että jokaiselle koodisanalle mitä laitteet lähettävät lasketaan tarkistussumma erityisen generaattoripolynomin avulla. Lähetettävä sana jaetaan generaattoripolynomilla ja saatu jakojäännös on tarkistussumma. [9]

32-bittinen CRC on äärimmäisen varma (99.99999998 %), sen avulla huomataan melkein varmasti kaikki virheet. 64 bittinen CRC on käytännössä täysin varma. Protokollassa käytettiin 7-bittistä CRC:tä eli CRC-7:ää jotta tarkistesumma mahtuisi tavun sisään. [9]

Ennen CRC:n generaattoripolynomin laskentaa, viestin neljä tavua yhdistetään pitkäksi binäärijonoksi. Tätä binäärijonoa vasten aletaan tekemään binääri XOR vertailua generaattoripolynomia vasten aloittaen merkittävimmästä bitistä jonka jälkeen siirrytään seuraavaan merkittävimpään bittiin kunnes saadaan tarkistesumma. Esimerkki lyhemmästä laskennasta, kuin mitä protokollassa tehdään, on esitetty kuvassa 8. [9]



Kuva 8. Esimerkki CRC laskennasta

8 Arduino kirjasto-ohjelma

Tärkeä työn osa oli kirjoittaa kirjasto-ohjelma Arduino-mikrokontrollereille, jotta protokolla saataisiin toimimaan ja testattua. Ohjelman koodi on luettavissa liitteissä 1 ja 2. Liite 1 sisältää C++ koodikielen h-tiedoston ja liite 2 cpp-tiedoston. [10]

8.1 Kieli

Arduino-mikrokontrollerit ohjelmoidaan käyttämällä Arduinon omaa ohjelmointi työkalua Arduino SDK:ta. Tällä työkalulla voidaan ohjelmoida kaikki, vanhat sekä uudet Arduino-mikrokontrollerit. Arduino-mikrokontrollerit suunniteltiin aikoinaan helpottamaan maallikkojen työskentelyä ja harrastuksen aloittamista mikrokontrollerien parissa, joten perinteiset C++/C# kielet piti vaihtaa yksinkertaisempaan lähestymistapaan. Kehittäjät päätyivät käyttämään kirjasto funktioita joita voitaisiin kutsua ohjelmalla helpottamaan ja yksinkertaistamaan Arduinon ohjelmointia. C++ kielessä h-tiedosto toimii esittely tiedostona jossa kaikki muuttujat ja funktiot esitellään. Cpp-tiedosto sisältää varsinaisen koodin joka toteutetaan. [10]

GRC:llä ei ole vahvaa osaamista C++/C# kielistä joten kirjasto-ohjelma kirjoitettiin käyttämällä Arduinon omia funktioita hyväkseen ohjelmoinnissa joiden käyttöön oli totuttu GRC:n tutkimuskeskuksessa. Tämä nopeutti ja yksinkertaisti työtä, sillä Arduinon funktiot ovat tehty helpottamaan ohjelmointia.

8.2 Toiminta

Käytännössä ohjelma joka on asetettu orjaksi, odottaa viestiä vastaanotettavaksi UART-sarjaliikenne portista taikka I^2C kenttäväylä liitännästä. Saatuaan viisi tavua, joka on jokaisen viestin koko, Ohjelma ratkoo CRC tarkistesumman ja vertaa sitä vastaanottamaansa tarkistelukuun. Jos luvut täsmäävät, on viesti hyväksytty ja edetään viestin mukaisesti. Jos viesti sisältää pyynnön takaisin lähetyksestä vastaa ohjelma lopuksi protokollan mukaisella tavalla viestiin. Jos CRC tarkistesummat eivät täsmää, ohjelma lähettää virhe viestin takaisin isäntälaitteelle.

Kirjasto ohjelma pystyy tekemään mikrokontrollerista isäntälaitteen muille mikrokontrollereille. Ohjelmassa kutsumalla funktioita kuten DigitalRead taikka AnalogRead voidaan nimiensä mukaisesti lukea pinnin analogi taikka binääri arvo toisesta mikrokontrollerista. DigitalWrite ja AnalogWrite funktioiden kutsuilla taas voidaan kirjoittaa orjana toimivan mikrokontrollerin pinneihin.

9 Protokollan testaaminen

9.1 Testialusta

Testialustaksi valittiin aiemmin esitelty iRW robotoitu pyörätuoli. Valinta kohdistui iRW koska se oli prototyypeistä edistynein sekä monimutkaisin. iRW pitää pystyä ohjaamaan ohjaussauvalla, mobiililaitteella sekä käsitankoa työntämällä ja vetämällä. iRW ydin on Rasperry Pi-tietokone. Raspberry Pin työnä on ohjata kaikkia iRW:n toimintoja ja vaihtaa ohjaustapaa käyttäjän toiveiden mukaisesti.

Raspberry Pi oli kytkettynä Arduino Mega-mikrokontrolleriin UART-sarjaliikenne väylällä ja toimi väylän isäntänä. Mega oli elektronisesti kytketty kaikkiin antureihin ja toimilaitteisiin ja luki ja ohjasi niitä Raspberry Piin ohjeistamana. Mega oli myös kytketty

toisella sarjaliikenne väylällä Arduino Uno-mikrokontrolleriin joka ohjasi pyörätuolin tuolin korkeutta ja kaltevuutta. Arduino Unoakin ohjattiin Raspberry Piistä, Mega toimi protokollan viestien välittäjänä.

9.2 Tulokset

Testi suoritettiin ajamalla iRW pyörätuolilla ympäri kampus aluetta ja tutkimuskeskuksen tiloja. Testissä tehtiin nopeita käänteitä, äkkipysäytyksiä ja nopeita kiihdytyksiä joihin pitää olla varautunut lopullisessa tuotteessakin. Testin tulokset arvioitiin silmämääräisesti.

Ennen testejä ja testien aikana huomattiin monia pieniä puutteita kirjasto-ohjelmassa sekä iRW ohjausohjelmassa. Vikojen korjauksen jälkeen saimme testeistä hyviä tuloksia ja totesimme protokollan ja kirjasto-ohjelman toimivaksi iRW pyörätuolin kanssa.

10 Yhteenveto

Tämän opinnäytetyöntaustalla oli kehittää Taiwanissa sijaitsevalle Gerontechnology Research Centerille sarjaliikenne väylällä toimiva tiedonsiirto protokolla ja ohjelmoida kirjasto-ohjelma toimivaksi Arduino mikrokontrollerille. Työ tehtiin vaihtojaksoni aikana ja toteutettiin yhteistyössä muiden opiskelijoiden avulla, hyödyntäen heidän tietouttaan, jota protokolla tulisi kyetä tekemään. Protokollaa lähdettiin kehittämään vanhan protokollan päälle, joskin lopputulos vaikuttaa täysin erilaiselta kuin alkuperäinen protokolla.

Lopullinen ohjelmointi ja suunnittelutyö tehtiin itsenäisesti, suurimmalta osin hyödyntäen jo opittua asiaa ja lukemalla verkkomateriaaleja tiedon puutteen ilmentyessä. Testaaminen tehtiin pari työnä iRW pyörätuolin kehittäjän kanssa, joka myös vastasi protokollan hyväksynnästä. Ekojen testien vian korjaamisen jälkeen, protokolla läpäisi testit ja otettiin välittömästi käyttöön GRC:n tutkimuskeskuksessa.

Uuden protokollan hyödyiksi voidaan lukea sen joustavuus ja monikäyttöisyys. Vaikka työn otsikossa mainitaan geronteknologia robotit, voidaan sitä yhtä hyvin käyttää muissakin robotiikka taikka automaatio sovelluksissa. Yksinkertaisuutensa ansiosta proto-

kolla on myös helppo ottaa käyttöön, eikä vaadi käyttäjältä suuria resursseja hyödyntääkseen protokollaa.

Tulevaisuudessa on ajatus kehittää protokollaa enemmän ja pohtia, miten sitä voitaisiin hyödyntää kaupallistetuissa tuotteissa eikä ainoastaan prototyyppien valmistamisen nopeuttamisessa.

Lähteet

1. GRC - Development of an intelligent robotic wheelchair as the center of mobility, health care, and daily living of older adults Tekninen julkaisu http://designer.mech.yzu.edu.tw/Content.aspx?CatSubID=173 Luettu 28.7.2014

2. GRC - Wobot - The Prototype Development of an Interaction and Communication Platform by Using the Telepresence Technique Tekninen julkaisu http://designer.mech.yzu.edu.tw/Content.aspx?CatSubID=176 Luettu 28.7.2014

3. GRC TRiCmini+ – Telepresence Robot for Interpersonal Communication for Older Adults Tekninen julkaisu http://140.138.40.170/articlesystem/article/compressedfile/%282012-09-03%29%20Telepresence%20Robot%20for%20Interpersonal%20Communication%20for%20Older%20Adults.aspx?ArchID=1859 Luettu 28.7.2014

4. GRC – Gerontechnology Research Center in Yuan Ze University Verkkomateriaali http://grc.yzu.edu.tw/Files/20121031_Gerontechnology%20Research%20in%20Yuan%20Ze%20University.pdf Luettu 28.7.2014

5. Sparkfun - RS-232 vs. TTL Serial Communication Verkkoartikkeli https://www.sparkfun.com/tutorials/215. Luettu 28.7.2014

6. Society of robots - MICROCONTROLLER UART TUTORIAL. Verkkoartikkeli http://www.societyofrobots.com/microcontroller_uart.shtml. Luettu 28.7.2014

7. NXP - UM10204 I^2C-bus specification and user manual. Ohjekirja http://www.nxp.com/documents/user_manual/UM10204.pdf. Luettu 28.7.2014

8. NXP – AN10216-01 I^2C manual. Ohjekirja http://www.nxp.com/documents/application_note/AN10216.pdf. Luettu 28.7.2014

9. TKK - Virheiden käsittely tiedonsiirrossa. Koulumateriaali. http://www.netlab.tkk.fi/opetus/s38118/s00/tyot/37/index.shtml.html. Luettu 10.8.2014

10. Arduino – Language reference. Verkkomateriaali. http://arduino.cc/en/Reference/HomePage. Luettu 10.8.2014

USIP protokollan kirjasto ohjelma h-tiedosto

```
#ifndef UISP_H

#define UISP_H

#include "Arduino.h"

#include "Wire.h"

#include <EEPROM.h>

#include <VarSpeedServo.h>




                /*VarSpeedServo Servo1;
                // Call for servo library

                VarSpeedServo Servo2;
                // Call for servo library

                VarSpeedServo Servo3;
                // Call for servo library

                VarSpeedServo Servo4;
                // Call for servo library

                VarSpeedServo Servo5;
                // Call for servo library

                VarSpeedServo Servo6;
                // Call for servo library

                VarSpeedServo Servo7;
                // Call for servo library

                VarSpeedServo Servo8;
                // Call for servo library*/



void receiveEvent(int howMany);
```

```
void requestEvent();



const int TIME_OUT                = 100;

const byte DIGITAL_READ        = 0x0A;

const byte DIGITAL_READ_PM  = 0x0B;

const byte DIGITAL_WRITE       = 0x14;

const byte DIGITAL_WRITE_PM = 0x15;

const byte ANALOG_READ        = 0x1E;

const byte ANALOG_READ_PM   = 0x1F;

const byte ANALOG_WRITE      = 0x28;

const byte ANALOG_WRITE_PM  = 0x29;


const byte CRC_CODE              = 0x89;


const uint8_t NumberOfBytesR = 0x04;



void KUSIPSetup(byte mode, byte add, int speed);

int Idle();

int Action();


int DO();
```

```
int DI();

int AI();

int AO();

int ServoDrive();

int DECRC5(byte packet, byte packet1, byte packet2, byte packet3, byte packet4);

int CRC(byte pin, byte Data1, byte Data2, byte Data3);


long Func1(byte pin, byte data1, byte data2, byte data3);

long Func2(byte pin, byte data1, byte data2, byte data3);

long Func3(byte pin, byte data1, byte data2, byte data3);

long Func4(byte pin, byte data1, byte data2, byte data3);

long Func5(byte pin, byte data1, byte data2, byte data3);

long Func6(byte pin, byte data1, byte data2, byte data3);

long Func7(byte pin, byte data1, byte data2, byte data3);

long Func8(byte pin, byte data1, byte data2, byte data3);

long Func9(byte pin, byte data1, byte data2, byte data3);

long Func10(byte pin, byte data1, byte data2, byte data3);

long Func11(byte pin, byte data1, byte data2, byte data3);

long Func12(byte pin, byte data1, byte data2, byte data3);

long Func13(byte pin, byte data1, byte data2, byte data3);

long Func14(byte pin, byte data1, byte data2, byte data3);

long Func15(byte pin, byte data1, byte data2, byte data3);

long Func16(byte pin, byte data1, byte data2, byte data3);
```

```
long Func17(byte pin, byte data1, byte data2, byte data3);

long Func18(byte pin, byte data1, byte data2, byte data3);

long Func19(byte pin, byte data1, byte data2, byte data3);

long Func20(byte pin, byte data1, byte data2, byte data3);


void KUSIPwrite(byte Pin, byte Data1, byte Data2, byte Data3, byte address);

int KUSIPread(byte Pin, byte Data1, byte Data2, byte Data3, byte address);


void DigitalWrite(byte Pin, byte Data);

void AnalogWrite(byte Pin, byte Data);

void DigitalWrite(byte Pin, byte Data, byte address);

void AnalogWrite(byte Pin, byte Data, byte address);

int DigitalRead(byte Pin);

int DigitalRead(byte Pin, byte address);

int AnalogRead(byte Pin);

int AnalogRead(byte Pin, byte address);


void ServoWrite(byte Pin, int pos, byte spd);

void ServoWrite(byte Pin, int pos, byte spd, byte address);


void FuncWrite(byte Pin, byte Data1, byte  Data2, byte Data3);

void FuncWrite(byte Pin, byte Data1, byte  Data2, byte Data3, byte address);

long FuncRead(byte Pin, byte Data1, byte  Data2, byte Data3);
```

```
long FuncRead(byte Pin, byte Data1, byte  Data2, byte Data3, byte address);


#endif
```

USIP protokollan kirjasto ohjelma cpp-tiedosto

```
#include "UISP.h"
#include "Arduino.h"
#include "Wire.h"
#include <EEPROM.h>
#include <VarSpeedServo.h>
//#include <avr/wdt.h>


                VarSpeedServo Servo1;
                // Call for servo library
                VarSpeedServo Servo2;
                // Call for servo library
                VarSpeedServo Servo3;
                // Call for servo library
                VarSpeedServo Servo4;
                // Call for servo library
                VarSpeedServo Servo5;
                // Call for servo library
                VarSpeedServo Servo6;
                // Call for servo library
                VarSpeedServo Servo7;
                // Call for servo library
                VarSpeedServo Servo8;
                // Call for servo library
                VarSpeedServo Servo9;
                // Call for servo library
                VarSpeedServo Servo10;
                // Call for servo library
                VarSpeedServo Servo11;
                // Call for servo library
                VarSpeedServo Servo12;
                // Call for servo library

int modeM;
int addM;
int speedM;

byte Message[5];
byte Packet[4];
byte UServo[12][2];
byte Servocount;
byte WireAmount;
byte PacketWrite;
byte EventReceived;

void KUSIPSetup(byte mode, byte add, int speed)
{
//Servo1.attach(11);
modeM = mode;
addM = add;
speedM = speed;

if (bitRead(modeM,1) == 1 )
{
                if (addM > 0)
                {               Serial.println("I2C Slave");
                                Serial.println(addM);
                                Wire.begin(addM);
                                Wire.onReceive(receiveEvent); // register event
                                Wire.onRequest(requestEvent); // register event
                }
```

```
                else
                {
                        Wire.begin();
                }
}
if (bitRead(modeM,0)== 1)
{
                #ifdef __AVR_ATmega32U4__
                        Serial1.begin(speedM);
                        //Serial.println("SS");
                #else
                        Serial.begin(speedM);
                #endif

}


                for (int i = 1; EEPROM.read(i) < 255 ; i = i + 2){
                int p = EEPROM.read(i);
                int j = EEPROM.read(i+1);

                        switch(j){

                        case 10:
                        pinMode(p, INPUT);
                        break;

                        case 20:
                        pinMode(p, OUTPUT);
                        break;

                        case 30:
                        pinMode(p, INPUT_PULLUP);
                        break;
                        }
                }
}
int Idle()
                                                                        // Serial idle
{
/*int i;
if (EventReceived == 1)
 {              Serial.println("ReceiveEvent");
                 i = DECRC5(Message[0], Message[1], Message[2], Message[3], Message[4]);
                if (i == 1){
                        Action();
                        Message[0] = 0;
                        Message[1] = 0;
                        Message[2] = 0;
                        Message[3] = 0;
                        Message[4] = 0;
                        WireAmount = 0;
                        PacketWrite = 0;
                        EventReceived = 0;
                }
}*/
int msg = 0;
int msg1;
int roskis;
                if (modeM == 0)
                {
                #ifdef __AVR_ATmega32U4__
                        if (Serial1.available() > 4)
                        {
```

```
                                        for (int i = 0; i < 5;  i++)
                                        {
                                                Message[i] = Serial1.read();
                                                Serial.println(Message[i]);
                                        }
                                        msg1 = DECRC5(Message[0], Message[1], Message[2],
Message[3], Message[4]);

                                        if (msg1 == 1)
                                        {
                                                msg = Action();
                                                if (msg != -1){
                                                        Seri-
al1.print(char(Packet[0]));
                                                        Seri-
al1.print(char(Packet[1]));
                                                        Seri-
al1.print(char(Packet[2]));
                                                        Seri-
al1.print(char(Packet[3]));
                                                        Seri-
al1.print(char(Packet[4]));
                                                        delay(1);
                                                }
                                        }
                                        else
                                        {
                                                //Serial.println("CRC Error");

                                                while(Serial1.available()){
                                                        roskis = Serial1.read();
                                                }
                                                Serial1.print(0xff);
                                                Serial1.print(0xff);
                                                Serial1.print(0xff);
                                                Serial1.print(0xff);
                                                Serial1.print(0x7A);
                                                delay(1);
                                        }
                                }
                #else
                                if (Serial.available() > 4)
                                {
                                        for (int i = 0; i < 5;  i++)
                                        {
                                                Message[i] = Serial.read();
                                        }
                                        msg1 = DECRC5(Message[0], Message[1], Message[2],
Message[3], Message[4]);

                                        if (msg1 == 1)
                                        {
                                                msg = Action();
                                                if (msg != -1){
                                                        Seri-
al.print(char(Packet[0]));
                                                        Seri-
al.print(char(Packet[1]));
                                                        Seri-
al.print(char(Packet[2]));
                                                        Seri-
al.print(char(Packet[3]));
                                                        Seri-
al.print(char(Packet[4]));
```

```
                                                                                delay(1);
                                                                        }
                                                                }
                                                                else
                                                                {
                                                                        while (Serial.available())
                                                                        {
                                                                                roskis = Serial.read();
                                                                        }
                                                                        Serial.print(0xff);
                                                                        Serial.print(0xff);
                                                                        Serial.print(0xff);
                                                                        Serial.print(0xff);
                                                                        Serial.print(0x7A);
                                                                        delay(1);
                                                                }
                                                        }
                        #endif
}
}
int Action()
                                                                // Action
{
long feedback = 0;
                        /*Serial.println("Action");
                        Serial.println(Message[0]);
                        delay(1);
                        Serial.println(Message[1]);
                        delay(1);
                        Serial.println(Message[2]);
                        delay(1);
                        Serial.println(Message[3]);*/
                        if (Message[0] > 0 && Message[0] < 71 ){                                        //      Nor-
mal DI/DO/AI/AO
                                        if (Message[1] > 9 && Message[1] < 20){
                                        feedback = DI();
                                        }
                                        if (Message[1] > 19 && Message[1] < 30){
                                        feedback = DO();
                                        }
                                        if (Message[1] > 29 && Message[1] < 40){
                                        feedback = AI();
                                        }
                                        if (Message[1] > 39 && Message[1] < 50){
                                        feedback = AO();
                                        }
                        }
                        else if (Message[0] > 70 && Message[0] < 125){                  // Servo
                                        feedback = ServoDrive();
                        }
                        else if(Message[0] > 124)
                        {
                                        switch(Message[0]){
                                        case 125:
                                        feedback = Func1(Message[0], Message[1], Message[2], Message[3]);
                                        break;

                                        case 126:
                                        feedback = Func2(Message[0], Message[1], Message[2], Message[3]);
                                        break;

                                        case 127:
                                        feedback = Func3(Message[0], Message[1], Message[2], Message[3]);
                                        break;
```

```
case 128:
feedback = Func4(Message[0], Message[1], Message[2], Message[3]);
break;

case 129:
feedback = Func5(Message[0], Message[1], Message[2], Message[3]);
break;
case 130:
feedback = Func6(Message[0], Message[1], Message[2], Message[3]);
break;

case 131:
feedback = Func7(Message[0], Message[1], Message[2], Message[3]);
break;

case 132:
feedback = Func8(Message[0], Message[1], Message[2], Message[3]);
break;

case 133:
feedback = Func9(Message[0], Message[1], Message[2], Message[3]);
break;

case 134:
feedback = Func10(Message[0], Message[1], Message[2], Message[3]);
break;

case 135:
feedback = Func11(Message[0], Message[1], Message[2], Message[3]);
break;

case 136:
feedback = Func12(Message[0], Message[1], Message[2], Message[3]);
break;

case 137:
feedback = Func13(Message[0], Message[1], Message[2], Message[3]);
break;

case 138:
feedback = Func14(Message[0], Message[1], Message[2], Message[3]);
break;

case 139:
feedback = Func15(Message[0], Message[1], Message[2], Message[3]);
break;

case 140:
feedback = Func16(Message[0], Message[1], Message[2], Message[3]);
break;

case 141:
feedback = Func17(Message[0], Message[1], Message[2], Message[3]);
break;

case 142:
feedback = Func18(Message[0], Message[1], Message[2], Message[3]);
break;

case 143:
feedback = Func19(Message[0], Message[1], Message[2], Message[3]);
break;
```

```
                                        case 144:
                                        feedback = Func20(Message[0], Message[1], Message[2], Message[3]);
                                        break;
                        }
                        }
                        //Serial.print("feedback ");
                        //Serial.println(feedback);
                        if (feedback != -1)
                        {

                        Packet[0] = Message[0];
                        Packet[1] = (byte) feedback;
                        Packet[2] = (byte) (feedback >> 8);
                        Packet[3] = (byte) (feedback >> 16);
                        if (modeM == 0){
                                        Packet[4] = CRC(Packet[0], Packet[1], Packet[2], Packet[3]);
                        }
                        /*Serial.println(Message[0]);
                        Serial.println(Packet[0]);
                        Serial.println((byte) feedback);
                        Serial.println(Packet[1]);
                        Serial.println((byte) (feedback >> 8));
                        Serial.println(Packet[2]);
                        Serial.println((byte) (feedback >> 16));
                        Serial.println(Packet[3]);*/
                        }
                        //return feedback;
}
int DI()
                                                                                // Digital input
 {
//Serial.println("DI");
int x = -1;
int y;
int z;
int j;
                        switch(Message[1]){

                        case DIGITAL_READ:
                                                // Digital read
                                        x = digitalRead(Message[0]-1);

                        break;

                        case DIGITAL_READ_PM:
                                        int i = EEPROM.read(0);

                                        if (i == 255){
                                                        i = 1;
                                                        EEPROM.write(0, i);
                                        }

                                        else {
                                        int z = 0 ;
                                        for (int n = 1; n <= i; n++){
                                                        z++;
                                                        if (Message[0]-1 == EEPROM.read((y*2)-1)){
                                                        y = 255;
                                                        break;
                                                        }
                                        }
                                        }
                                                        if (y != 255){
                                                        i++;
```

```
                    EEPROM.write(0, i);
                    i = i*2;
                    j = i - 1;
}
else{
                    i = z*2;
                    j = i - 1;
}

if (Message[0] > 54){

switch(Message[0]){
case 55:
pinMode(A0, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,10);
break;
case 56:
pinMode(A1, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,10);
break;
case 57:
pinMode(A2, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,10);
break;
case 58:
pinMode(A3, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,10);
break;
case 59:
pinMode(A4, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,10);
break;
case 60:
pinMode(A5, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,10);
break;
case 61:
pinMode(A6, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,10);
break;
#ifdef __AVR_ATmega32U4__
case 62:
pinMode(A7, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 63:
pinMode(A8, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 64:
pinMode(A9, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 65:
```

```
                pinMode(A10, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,20);
                break;
                case 66:
                pinMode(A11, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,20);
                break;
                #elif __ATmega2560__
                case 62:
                pinMode(A7, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,20);
                break;
                case 63:
                pinMode(A8, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,20);
                break;
                case 64:
                pinMode(A9, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,20);
                break;
                case 65:
                pinMode(A10, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,20);
                break;
                case 66:
                pinMode(A11, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,20);
                break;
                case 67:
                pinMode(A12, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,20);
                break;
                case 68:
                pinMode(A13, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,20);
                break;
                case 69:
                pinMode(A14, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,20);
                break;
                case 70:
                pinMode(A15, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,20);
                break;
                #endif
                }
                }
                else{
                pinMode(Message[0]-1, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,10);
                }
break;
```

```
                }
                return x;

}
int DO()
                                                                // Digital output
{
int y;
int z;
int j;
//Serial.println("DO");
//Serial.println(Message[0]);
//Serial.println(Message[1]);
//Serial.println(Message[2]);
//Serial.println(Message[3]);
                switch(Message[1]){

                case DIGITAL_WRITE:
                                                // Digital write
                        if (Message[2] == 17){
                                                digitalWrite(Message[0]-1, HIGH);
                        }
                        else if (Message[2] == 16){
                                                digitalWrite(Message[0]-1, LOW);
                        }
                break;
                case DIGITAL_WRITE_PM:
                        int i = EEPROM.read(0);

                        if (i == 255){
                                                i = 1;
                                                EEPROM.write(0, i);
                        }

                        else {
                        int z = 0 ;
                        for (int n = 1; n <= i; n++){
                                                z++;
                                                if (Message[0]-1 == EEPROM.read((y*2)-1)){
                                                                y = 255;
                                                                break;
                                                }
                        }
                        }
                                                if (y != 255){
                                                i++;
                                                EEPROM.write(0, i);
                                                i = i*2;
                                                j = i - 1;
                        }
                        else{
                                                i = z*2;
                                                 j = i - 1;
                        }

                        if (Message[0] > 54){

                        switch(Message[0]){
                        case 55:
                        pinMode(A0, OUTPUT);
                        EEPROM.write(j,Message[0]-1);
                        EEPROM.write(i,20);
                        break;
                        case 56:
```

```
pinMode(A1, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 57:
pinMode(A2, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 58:
pinMode(A3, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 59:
pinMode(A4, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 60:
pinMode(A5, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 61:
pinMode(A6, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
#ifdef __AVR_ATmega32U4__
case 62:
pinMode(A7, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 63:
pinMode(A8, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 64:
pinMode(A9, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 65:
pinMode(A10, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 66:
pinMode(A11, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
#elif __ATmega2560__
case 62:
pinMode(A7, OUTPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 63:
pinMode(A8, OUTPUT);
EEPROM.write(j,Message[0]-1);
```

```
                                        EEPROM.write(i,20);
                                        break;
                                        case 64:
                                        pinMode(A9, OUTPUT);
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,20);
                                        break;
                                        case 65:
                                        pinMode(A10, OUTPUT);
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,20);
                                        break;
                                        case 66:
                                        pinMode(A11, OUTPUT);
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,20);
                                        break;
                                        case 67:
                                        pinMode(A12, OUTPUT);
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,20);
                                        break;
                                        case 68:
                                        pinMode(A13, OUTPUT);
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,20);
                                        break;
                                        case 69:
                                        pinMode(A14, OUTPUT);
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,20);
                                        break;
                                        case 70:
                                        pinMode(A15, OUTPUT);
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,20);
                                        break;
                                        #endif
                                        }
                                        }
                                        else{
                                        pinMode(Message[0]-1, OUTPUT);
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,10);
                                        }
                        break;
                        }
                        return -1;
}
int AI()
                                                                        // Analog Input
 {
//Serial.println("AI");
int x = -1;
int y;
int z;
int j;
                        switch(Message[1]){

                        case ANALOG_READ:
                                                // Digital read
                                x = analogRead(Message[0]-55);
                        break;
```

```
case ANALOG_READ_PM:
                int i = EEPROM.read(0);

                if (i == 255){
                                i = 1;
                                EEPROM.write(0, i);
                }

                else {
                int z = 0 ;
                for (int n = 1; n <= i; n++){
                                z++;
                                if (Message[0]-1 == EEPROM.read((y*2)-1)){
                                y = 255;
                                break;
                                }
                }
                }
                                if (y != 255){
                                i++;
                                EEPROM.write(0, i);
                                i = i*2;
                                j = i - 1;
                }
                else{
                                i = z*2;
                                j = i - 1;
                }
                switch(Message[0]){
                                case 55:
                                pinMode(A0, INPUT);
                                EEPROM.write(j,Message[0]-1);
                                EEPROM.write(i,10);
                                break;

                                case 56:
                                pinMode(A1, INPUT);
                                EEPROM.write(j,Message[0]-1);
                                EEPROM.write(i,10);
                                break;

                                case 57:
                                pinMode(A2, INPUT);
                                EEPROM.write(j,Message[0]-1);
                                EEPROM.write(i,10);
                                break;

                                case 58:
                                pinMode(A3, INPUT);
                                EEPROM.write(j,Message[0]-1);
                                EEPROM.write(i,10);
                                break;

                                case 59:
                                pinMode(A4, INPUT);
                                EEPROM.write(j,Message[0]-1);
                                EEPROM.write(i,10);
                                break;

                                case 60:
                                pinMode(A5, INPUT);
                                EEPROM.write(j,Message[0]-1);
                                EEPROM.write(i,10);
                                break;
```

```
                case 61:
                pinMode(A6, INPUT);
                EEPROM.write(j,Message[0]-1);
                EEPROM.write(i,10);
                break;
                                #ifdef __AVR_ATmega32U4__
case 62:
pinMode(A7, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 63:
pinMode(A8, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 64:
pinMode(A9, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 65:
pinMode(A10, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 66:
pinMode(A11, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
#elif __ATmega2560__
case 62:
pinMode(A7, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 63:
pinMode(A8, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 64:
pinMode(A9, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 65:
pinMode(A10, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 66:
pinMode(A11, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 67:
pinMode(A12, INPUT);
EEPROM.write(j,Message[0]-1);
EEPROM.write(i,20);
break;
case 68:
```

```
                                        pinMode(A13, INPUT);
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,20);
                                        break;
                                        case 69:
                                        pinMode(A14, INPUT);
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,20);
                                        break;
                                        case 70:
                                        pinMode(A15, INPUT);
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,20);
                                        break;
                                        #endif
                                        }
                        break;
                        }
                        return x;

}
int AO()
                                                                        // Analog output
{
Serial.println("AO");
int y;
int z;
int j;
                        switch(Message[1]){

                        case ANALOG_WRITE:
                                                // Analog write
                                analogWrite(Message[0]-1, Message[2]);
                        break;
                        case ANALOG_WRITE_PM:
                                int i = EEPROM.read(0);

                                if (i == 255){
                                                i = 1;
                                                EEPROM.write(0, i);
                                }

                                else {
                                int z = 0 ;
                                for (int n = 1; n <= i; n++){
                                                z++;
                                                if (Message[0]-1 == EEPROM.read((y*2)-1)){
                                                y = 255;
                                                break;
                                                }
                                }
                                }
                                                if (y != 255){
                                                i++;
                                                EEPROM.write(0, i);
                                                i = i*2;
                                                j = i - 1;
                                }
                                else{
                                                i = z*2;
                                                j = i - 1;
                                }

                                pinMode(Message[0]-1, OUTPUT);
```

```
                                        EEPROM.write(j,Message[0]-1);
                                        EEPROM.write(i,20);
                        break;
                        }
                        return -1;
 }
int ServoDrive()
                                        // Servo
 {
int Position = (Message[3] << 8) | Message[2];
int x = 0;
int y = 0;
int z = 0;
int j = 0;
 //Serial.println(millis());
 //Serial.print("Position: ");
//Serial.println(Position);
//Serial.print("Servocount: ");
//Serial.println(Servocount);

 for (int i =0; i < 12; i++){
                        //Serial.print (UServo[i][0]);
                        //Serial.print (" ");
                        //Serial.println (UServo[i][1]);

                        if (UServo[i][0] == Message[0])
                        {
                                        x = 1;
                                        if (Position != UServo[i][1] || Position > 255)
                                        {
                                                        y = 1;
                                        }
                                        break;
                        }
z++;
}
//Serial.print("Z ");
//Serial.println(z);
                        if (z == 12)
                        {               j = Servocount;
                                        UServo[Servocount][0] = Message[0];
                                        UServo[Servocount][1] = Position;
                                        Servocount++;

                        switch(Servocount-1){
                        case 0:
                        //Serial.println("S 1 Detach");
                                        Servo1.attach(Message[0]-71);
                                        break;
                        case 1:
                        //Serial.println("S 2 Detach");
                                        Servo2.attach(Message[0]-71);
                                        break;
                        case 2:
                        //Serial.println("S 3 Detach");
                                        Servo3.attach(Message[0]-71);
                                        break;
                        case 3:
                        //Serial.println("S 4 Detach");
                                        Servo4.attach(Message[0]-71);
                                        break;
                        case 4:
                        //Serial.println("S 5 Detach");
                                        Servo5.attach(Message[0]-71);
```

```
                        break;
                case 5:
                //Serial.println("S 6 Detach");
                        Servo6.attach(Message[0]-71);
                        break;
                case 6:
                //Serial.println("S 7 Detach");
                        Servo7.attach(Message[0]-71);
                        break;
                case 7:
                //Serial.println("S 8 Detach");
                        Servo8.attach(Message[0]-71);
                        break;
                case 8:
                //Serial.println("S 8 Detach");
                        Servo9.attach(Message[0]-71);
                        break;
                case 9:
                //Serial.println("S 8 Detach");
                        Servo10.attach(Message[0]-71);
                        break;
                case 10:
                //Serial.println("S 8 Detach");
                        Servo11.attach(Message[0]-71);
                        break;
                case 11:
                //Serial.println("S 8 Detach");
                        Servo12.attach(Message[0]-71);
                        break;
                }
                }
                else
                {
                j = z;
                }


                //Serial.println(millis());
 if (z == 12 || (x == 1 && y == 1)){
switch(j)
{
case 0:
//Serial.println("SERVO1");
 //Serial.println(Position);
                //Serial.println(Message[0]-71);
//              Servo1.detach();
                //Servo1.attach(Message[0]-71);
                //Serial.print("attach ");
                //Serial.println(Servo1.attached());
                Servo1.slowmove(Position, Message[1]);
//              Servocount++;
                UServo[0][0] = Message[0];
                UServo[0][1] = Position;
break;
case 1:
//Serial.println("SERVO2");
                //Serial.println(Message[0]-71);
                //Servo2.detach();
                //Servo2.attach(Message[0]-71);

                Servo2.slowmove(Position, Message[1]);
//              Servocount++;
                UServo[1][0] = Message[0];
                UServo[1][1] = Position;
```

```
break;
case 2:
//Serial.println("SERVO3");
                    //Serial.println(Message[0]-71);
                    //Servo3.detach();
                    //Servo3.attach(Message[0]-71);
                    Servo3.slowmove(Position, Message[1]);
//                  Servocount++;
                    UServo[2][0] = Message[0];
                    UServo[2][1] = Position;
break;
case 3:
//Serial.println("SERVO4");
                    //Serial.println(Message[0]-71);
                    //Servo4.detach();
                    //Servo4.attach(Message[0]-71);
                    Servo4.slowmove(Position, Message[1]);
//                  Servocount++;
                    UServo[3][0] = Message[0];
                    UServo[3][1] = Position;
                    break;
case 4:
//Serial.println("SERVO5");
                    //Serial.println(Message[0]-71);
                    //Servo5.detach();
                    //Servo5.attach(Message[0]-71);
                    Servo5.slowmove(Position, Message[1]);
//                  Servocount++;
                    UServo[4][0] = Message[0];
                    UServo[4][1] = Position;
                    break;
case 5:
//Serial.println("SERVO6");
                    //Serial.println(Message[0]-71);
                    //Servo6.detach();
                    //Servo6.attach(Message[0]-71);
                    Servo6.slowmove(Position, Message[1]);
//                  Servocount++;
                    UServo[5][0] = Message[0];
                    UServo[5][1] = Position;
                    break;
case 6:
//S/erial.println("SERVO7");
                    //Serial.println(Message[0]-71);
                    //Servo7.detach();
                    //Servo7.attach(Message[0]-71);
                    Servo7.slowmove(Position, Message[1]);
//                  Servocount++;
                    UServo[6][0] = Message[0];
                    UServo[6][1] = Position;
                    break;
case 7:
//Serial.println("SERVO8");
                    //Serial.println(Message[0]-71);
                    //Servo8.detach();
                    //Servo8.attach(Message[0]-71);
                    Servo8.slowmove(Position, Message[1]);
//                  Servocount = 0;
                    UServo[7][0] = Message[0];
                    UServo[7][1] = Position;
                    break;
case 8:
//Serial.println("SERVO9");
                    //Serial.println(Message[0]-71);
```

```
                    //Servo9.detach();
                    //Servo9.attach(Message[0]-71);
                    Servo9.slowmove(Position, Message[1]);
//                  Servocount = 0;
                    UServo[8][0] = Message[0];
                    UServo[8][1] = Position;
                    break;
case 9:
//Serial.println("SERVO10");
                    //Serial.println(Message[0]-71);
                    //Servo10.detach();
                    //Servo10.attach(Message[0]-71);
                    Servo10.slowmove(Position, Message[1]);
//                  Servocount = 0;
                    UServo[9][0] = Message[0];
                    UServo[9][1] = Position;
                    break;
case 10:
//Serial.println("SERVO11");
                    //Serial.println(Message[0]-71);
                    //Servo11.detach();
                    //Servo11.attach(Message[0]-71);
                    Servo11.slowmove(Position, Message[1]);
//                  Servocount = 0;
                    UServo[10][0] = Message[0];
                    UServo[10][1] = Position;
                    break;
case 11:
//Serial.println("SERVO12");
                    //Serial.println(Message[0]-71);
                    //Servo12.detach();
                    //Servo12.attach(Message[0]-71);
                    Servo12.slowmove(Position, Message[1]);
//                  Servocount = 0;
                    UServo[11][0] = Message[0];
                    UServo[11][1] = Position;
                    break;
  }
}
return -1;
 }
void receiveEvent(int howMany)                                          //      I2C
Receive event
{
//wdt_reset ();
//Serial.print("ReceiveEvent ");
//Serial.println(howMany);
//Serial.println(WireAmount);
//int i ;
//int buff;
//buff = Wire.read();
//Serial.println(buff);
//if (buff != -1){
                    while(Wire.available()){
                    //Serial.println(buff);
                    Message[WireAmount] = Wire.read();
                    //Serial.println(Message[WireAmount]);
                    WireAmount++;
                    //buff = Wire.read();
                    //Serial.println(WireAmount);
                    }
//                  }
//else
//{
```

```
                //                  Message[0] = 0;
                //                  Message[1] = 0;
                                    //Message[2] = 0;
                                    //Message[3] = 0;
//}
                if (WireAmount >= 3){
                                    /*i = DECRC5(Message[0], Message[1], Message[2], Message[3], Mes-
sage[4]);
                                    if (i == 1){
                                                        Action();
                                                        Message[0] = 0;
                                                        Message[1] = 0;
                                                        Message[2] = 0;
                                                        Message[3] = 0;
                                                        Message[4] = 0;
                                                        WireAmount = 0;
                                                        PacketWrite = 0;

                                    }*/
                                    Action();
                                    Message[0] = 0;
                                    Message[1] = 0;
                                    Message[2] = 0;
                                    Message[3] = 0;
                                    //Message[4] = 0;
                                    PacketWrite = 0;
                                    //EventReceived = 1;
                }
WireAmount = 0;
}

void requestEvent()
                                                        // I2C Request event
{
Wire.write(Packet, 4);
}

void KUSIPwrite(byte Pin, byte Data1, byte Data2, byte Data3, uint8_t address)
{
//byte data21     = (byte) Data2;
//byte data22 = (byte) (Data2 >> 8);

if (address == 0)
{                                   // byte crc = CRC(pin, Data1, data21, data22);
                #ifdef __AVR_ATmega32U4__
                                    Serial1.print(Pin);
                                    Serial1.print(Data1);
                                    Serial1.print(Data2);
                                    Serial1.print(Data3);
                                    Serial1.print(CRC(Pin, Data1, Data2, Data3));
                #else
                                    Serial.print(Pin);
                                    Serial.print(Data1);
                                    Serial.print(Data2);
                                    Serial.print(Data3);
                                    Serial.print(CRC(Pin, Data1, Data2, Data3));
                #endif

                }
                else{
                                    Wire.beginTransmission(address);        // transmit to device
                                    Wire.write(Pin);                                                    //  sends
bytes
                                    Wire.write(Data1);
```

```
                                Wire.write(Data2);
                                Wire.write(Data3);
                                //Wire.write(CRC(pin, Data1, data21, data22));
                                Wire.endTransmission();                                 //  stop
transmitting
                }
}

int KUSIPread(byte Pin, byte Data1, int Data2, int Data3, uint8_t address)
{

int feedback;
int x = 0;
int TimeS = millis();
// byte crc = CRC(pin, Data1, data21, data22);
if (address == 0)
{
                #ifdef __AVR_ATmega32U4__
                                Serial1.print(Pin);
                                Serial1.print(Data1);
                                Serial1.print(Data2);
                                Serial1.print(Data3);
                                Serial1.print(CRC(Pin, Data1, Data2, Data3));
                                delay(5);
                                // Wait to slave to do he's calculation and send the data
                                while(Serial1.available() >= 4)
                                {
                                                int t = millis();
                                                if (t >= (TimeS + TIME_OUT))
                                                {
                                                                x = 1;
                                                                break;
                                                }

                                }
                                if (x == 0){
                                                byte r1 = Serial1.read();
                                                byte r2 = Serial1.read();
                                                byte r3 = Serial1.read();
                                                byte r4 = Serial1.read();
                                                byte r5 = Serial1.read();
                                                r5 = DECRC5(r1, r2, r3, r4, r5);

                                                if (r1 == Pin && r5 != -1)
                                                {
                                                                feedback = r2  + (r3 << 8) + (r4 << 16);
                                                }
                                                else
                                                {
                                                                feedback = -1;
                                                }
                                }
                                else
                                {
                                                feedback = -1;
                                }
                #else
                                Serial.print(Pin);
                                Serial.print(Data1);
                                Serial.print(Data2);
                                Serial.print(Data3);
                                Serial.print(CRC(Pin, Data1, Data2, Data3));
                                delay(5);
                                // Wait to slave to do he's calculation and send the data
```

```
                        while(Serial.available() >= 4)
                        {
                                int t = millis();
                                if (t >= (TimeS + TIME_OUT))
                                {
                                        x = 1;
                                        break;
                                }

                        }
                        if (x == 0){
                                byte r1 = Serial.read();
                                byte r2 = Serial.read();
                                byte r3 = Serial.read();
                                byte r4 = Serial.read();
                                byte r5 = Serial.read();

                                r5 = DECRC5(r1, r2, r3, r4, r5);

                                if (r1 == Pin && r5 != -1)
                                {
                                        feedback = r2  + (r3 << 8) + (r4 << 16);
                                }
                                else
                                {
                                        feedback = -1;
                                }
                        }
                        else
                        {
                                feedback = -1;
                        }
                #endif
}
else
{
                        Wire.beginTransmission(address);                // transmit to device
                        Wire.write(Pin);                                //  sends
bytes
                        Wire.write(Data1);
                        Wire.write(Data2);
                        Wire.write(Data3);
                //      Wire.write(CRC(Pin, Data1, Data2, Data3));
                        Wire.endTransmission();                         //    stop
transmitting
                        delay(1);
                        Wire.requestFrom(address, NumberOfBytesR);
                        while(Wire.available() >= NumberOfBytesR)
                        {
                                int t = millis();
                                if (t >= (TimeS + TIME_OUT))
                                {
                                        x = 1;
                                        break;
                                }

                        }
                        if (x == 0){
                                byte r1 = Wire.read();
                                byte r2 = Wire.read();
                                byte r3 = Wire.read();
                                byte r4 = Wire.read();
                                //byte r5 = Wire.read();
```

```
                                        //r5 = DECRC5(r1, r2, r3, r4, r5);

                                        if (r1 == Pin /*&& r5 != -1*/)
                                        {
                                                feedback = r2  + (r3 << 8) + (r4 << 16);
                                        }
                                        else
                                        {
                                                feedback = -1;
                                        }
                                }
                                else
                                {
                                        feedback = -1;
                                }
}
return feedback;
}
int DECRC5(byte packet, byte packet1, byte packet2, byte packet3, byte packet4)
{
unsigned long test = packet;
int feedback;
test = (test << 8) + packet1;
test = (test << 8) + packet2;
test = (test << 8) + packet3;
unsigned long CompareCRC = CRC_CODE ;
CompareCRC = CompareCRC << 24;

  for (int i = 0; i < 25; i++)
  {
    if (bitRead(test, 31-i) == 1)
                {
     test ^= CompareCRC;
                }
                  CompareCRC >>= 1;
   }

   if (test == packet4)
    {
                                feedback = 1;
                }
                else
                {
                                feedback = test;
                }
return feedback;
}
int CRC(byte pin, byte Data1, byte Data2, byte Data3)
{
unsigned long test = pin;
int feedback;
test = (test << 8) + Data1;
test = (test << 8) + Data2;
test = (test << 8) + Data3;                                                     // needed only for
DECRC5
unsigned long CompareCRC = CRC_CODE ;
CompareCRC = CompareCRC << 24;
  for (int i = 0; i < 25; i++)
  {
    if (bitRead(test, 31-i) == 1)
                {
     test ^= CompareCRC;
                }
                CompareCRC >>= 1;
```

```
  }

return test;
}

void DigitalWrite(byte Pin, byte Data)
{
KUSIPwrite((Pin+1), DIGITAL_WRITE, (Data + 16), 0, 0);
}

void AnalogWrite(byte Pin, byte Data)
{
KUSIPwrite((Pin+1), ANALOG_WRITE, Data, 0, 0);
}

void DigitalWrite(byte Pin, byte Data, byte address)
{
KUSIPwrite((Pin+1), DIGITAL_WRITE, (Data + 16), 0, address);
}

void AnalogWrite(byte Pin, byte Data, byte address)
{
KUSIPwrite((Pin+1), ANALOG_WRITE, Data, 0, address);
}

int DigitalRead(byte Pin)
{
return KUSIPread((Pin+1), DIGITAL_READ, 0, 0, 0);
}

int DigitalRead(byte Pin, byte address)
{
return KUSIPread((Pin+1), DIGITAL_READ, 0, 0, address);
}

int AnalogRead(byte Pin)
{
return KUSIPread((Pin+1), ANALOG_READ, 0, 0, 0);
}

int AnalogRead(byte Pin, byte address)
{
return KUSIPread((Pin+1), ANALOG_READ, 0, 0, address);
}

void FuncWrite(byte Pin, byte Data1, byte  Data2, byte Data3)
{
KUSIPwrite((Pin+124), Data1, Data2, Data3, 0);
}

void FuncWrite(byte Pin, byte Data1, byte  Data2, byte Data3, byte address)
{
KUSIPwrite((Pin+124), Data1, Data2, Data3, address);
}

long FuncRead(byte Pin, byte Data1, byte  Data2, byte Data3)
{
return KUSIPread((Pin+124), Data1, Data2, Data3, 0);
}

long FuncRead(byte Pin, byte Data1, byte  Data2, byte Data3, byte address)
{
return KUSIPread((Pin+124), Data1, Data2, Data3, address);
}
```

```
void ServoWrite(byte Pin, int pos, byte spd){
KUSIPwrite((Pin+71), spd, byte(pos), byte(pos>>8), 0);
}

void ServoWrite(byte Pin, int pos, byte spd, byte address)
{
KUSIPwrite((Pin+71), spd, byte(pos), byte(pos>>8), address);
}
```