Satakunnan ammattikorkeakoulu

OPINNÄYTETYÖ

Janne Äijälä

Satakunnan ammattikorkeakoulu

Janne Äijälä

Mikroprosessoriohjattu 3D-ristinolla

Tekniikka Pori

Tietotekniikan koulutusohjelma

Ohjelmistotekniikka

2008

MIKROPROSESSORIOHJATTU 3D-RISTINOLLA

Äijälä, Janne
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Maaliskuu 2008
Ekholm, Ari
UDK: 621.3.049.77, 621.38
Sivumäärä: 44

Asiasanat: ohjelmointi, sulautetut järjestelmät, elektroniikka, tietotekniikka

---

Tämän opinnäytetyön aiheena oli suunnitella ja toteuttaa kolmiulotteisen ristinollapelin prototyyppi käyttäen mikro-ohjaimia. Suunnittelussa korostettiin käytettävyyden tärkeyttä, mikä vaikutti sekä komponenttivalintoihin että laitteiston kehitysprosessiin.

Opinnäytetyön teoreettisessa osuudessa käsiteltiin käytettyjen mikro-ohjainten tarjoamia tekniikoita, liitäntöjä ja niiden toimintaperiaatteita, sekä selvitettiin laitteistolle ja ohjelmistolle asetetut rajoitukset ja suunnitteluvaatimukset. Lisäksi pohdittiin erilaisten valmistustekniikoiden vaikutusta piirilevyn ominaisuuksiin, sekä tutkittiin valmistustekniikoiden käytännön rajoituksia.

Käytännön työstä kertovassa osuudessa käsiteltiin työssä tarvittavien osien valmistamista ja työstölaitteiden käyttöä sekä laitteiston kasaamista. Lisäksi tässä osuudessa kerrotaan yksityiskohtaisesti lopullisen ohjelmiston tekemisestä sekä laitteiston asettamista käytännön rajoituksista ohjelmaa kirjoitettaessa.

Kehitysprosessin virstanpylväinä voitiin pitää erilaisten laiterajapintojen luomista ja eri teknologiaa edustavien laitteiden liittämistä toisiinsa onnistuneesti, mikä vaati tekijältään tarkkaa tietämystä käytetyistä teknologioista. Mikro-ohjaintekniikka rajoitettuine resursseineen asettaa haasteita ohjelmoijille sekä edellyttää hyvää ohjelmointikielen hallintaa.

Valmistettu prototyyppi testattiin onnistuneesti ja havaitut epäkohdat ovat vähäisiä ja korjattavissa varsinaiseen tuotantomalliin. Yhteenvetona voitiin todeta uuden teknologian käytön olevan välttämätöntä kehitettäessä käyttäjäystävällisiä viihdejärjestelmiä.

MICROPROCESSOR CONTROLLED 3D TICTACTOE

Äijälä, Janne
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information Technology
Software Engineering
March 2008
Ekholm, Ari
UDK: 621.3.049.77, 621.38
Number of pages: 44

Key words: programming, embedded systems, electronics, information technology

---

The purpose of this thesis was to design and implemement a three-dimensional tictactoe game using microcontrollers. The importance of usability was emphasized in the design and it affected both the selection of components and the development process of the hardware.

The theoretical part of the thesis deals with the technologies and interfaces offered by the microcontrollers, their operating principles and the limitations and design requirements set by both the hardware and software. The effects of different manufacturing techniques on the characteristics of a circuit board were also considered and practical limitations of the manufacturing techniques were analyzed.

The practical part of this thesis is about fabrication of the parts needed in the project, using the machining tools and contruction of the hardware. In addition, this part of the thesis elaborately deals with writing of the final program and the practical limitations set by the hardware.

The creation of different device interfaces and successful interconnection of different technologies can be considered the milestones of the development process, which required detailed knowledge about the technologies used. Microcontroller technology with its limited resources sets challenges for programmers and calls for a good knowledge of a programming language.

Fabricated prototype was successfully tested and detected anomalies were insignificant and can be fixed for the proper production model. In brief, it could be pointed out that it is essential to use new technology while developing user friendly entertainment systems.

SISÄLLYS

1 LYHENNELUETTELO

A/D-muunnin	Analogia-digitaalimuunnin on muunninpiiri, joka muuntaa analogisen tulojännitteen digitaaliseksi arvoksi. Tämä digitaalinen arvo on suhteutettu piirin referenssi- eli vertailujännitteeseen.
ASCII-merkistö	American Standard Code for Information Interchange on ensimmäisen kerran vuonna 1968 standardoitu tietokonemerkistö, joka muodostuu 128:sta merkistä.
CAN-väylä	Controller Area Network on alunperin ajoneuvojen ja teollisuuslaitteiden tiedonsiirtoon tarkoitettu kaksijohdinväylä
D/A-muunnin	Digitaali-analogiamuunnin on piiri, joka muuntaa digitaalisen tuloarvon analogiseksi lähtöjännitteeksi.
D-kiikku	D-kiikku on porttipiiri, jonka lähdöt päivittyvät reunaliipaistulla kellopulssilla.
D-veräjä	D-veräjä on porttipiiri, jonka lähdöt päivittyvät Chip Enable-tulon ollessa aktiivisena.
Debuggaus	Debuggaus on toimintaa, jolla järjestelmän toimintavirheitä haetaan systemaattisesti.
Flash-muisti	Flash-muisti on puolijohdemuistia, joka voidaan sekä tyhjentää että ohjelmoida uudelleen sähköisesti.
I/O-linja	Input/Output-linja on kaksisuuntainen kommunikaatiolinja.
I²C	Inter-integrated Circuit on Philipsin kehittämä, hitaaseen tiedonsiirtoon tarkoitettu sarjaväylä. Väylä käyttää tiedonsiirtoon kahta johdinta.
JTAG	Joint Test Action Group on liitäntä toimivan laitteen toiminnan debuggausta varten.
PS/2-liitäntä	PS/2 on yleisin näppäimistö/hiiriliitäntä aivan viime päiviin saakka. Liittimenä liitännässä on kuusipinninen Mini-DIN.

PWM	Pulssinleveysmodulointi on modulointitapa, jolla voidaan vaikuttaa kanttiaaltosignaalin tehollisarvoon muuttamalla signaalin pulssisuhdetta.
RISC-arkkitehtuuri	Reduced Instruction Set Computer on prosessoriarkkitehtuuri joka perustuu vain vähän, mutta tehokkaita käskyjä sisältävään käskykantaan.
RS-232	Recommended Standard 232 on asynkroninen sarjaliitäntä.
SPI	Serial Peripheral Interface Bus, synkroninen sarjaväylä, joka vaatii kellolinjan ja kahden datalinjan lisäksi valintalinjan jokaiselle liitetylle orjalaitteelle.
TTL-jännitetaso	Transistor-Transistor Logic, TTL-porttipiireille ominainen jännitetaso käyttöjännitteen ja signaalijännitteiden osalta.
TTY	Unix-käyttöpääte, jota alettiin kutsua Teletypeksi erään valmistajan, Teletypen mukaan.
USB-liitäntä	Universal Serial Bus, tällä hetkellä yleisin oheislaiteliitäntä kotitietokoneissa. Sarjaliitäntä, jossa differentiaaliset jännitetasot.

2 JOHDANTO

Tässä opinnäytetyössä käsitellään kolmiulotteisen ristinollapelin prototyypin suunnittelu- ja toteutusprosessia. Työ on lopulta tarkoitus asettaa näytteille Satakunnan Ammattikorkeakoulun tekniikka-aiheiseen aulanäyttelyyn. Työn alullepanija on Satakunnan Ammattikorkeakoulu, joka halusi teettää projektin opinnäytetyönä.

Työn suunnitteluvaiheessa haluttiin panostaa laitteen käytettävyyteen ja hyödyntää uutta teknologiaa mahdollisuuksien mukaan sekä tutkia sen liittämismahdollisuuksia kustannustehokkaaseen ja yleisesti saatavilla olevaan mikro-ohjaintekniikkaan. Pelin käyttöliittymänä haluttiin käyttää kosketuskalvoja asennettuna neljään ruutuihin jaettuun pelitasoon. Jokainen pelitaso toimii omana yksikkönään pelin isäntälaitteen ohjaamana.

Laitteen rakenteesta johtuen työtä ja toteutettavaa oli paljon, ja projekti tehtiin parityönä. Tässä opinnäytetyössä keskitytään käsittelemään pelitasojen toiminnallista rakennetta sekä pelitasojen ja isäntälaitteen välistä kommunikaatiota. Itse isäntälaitteen toimintaa käsittelee Pasi Vuorinen opinnäytetyössään.

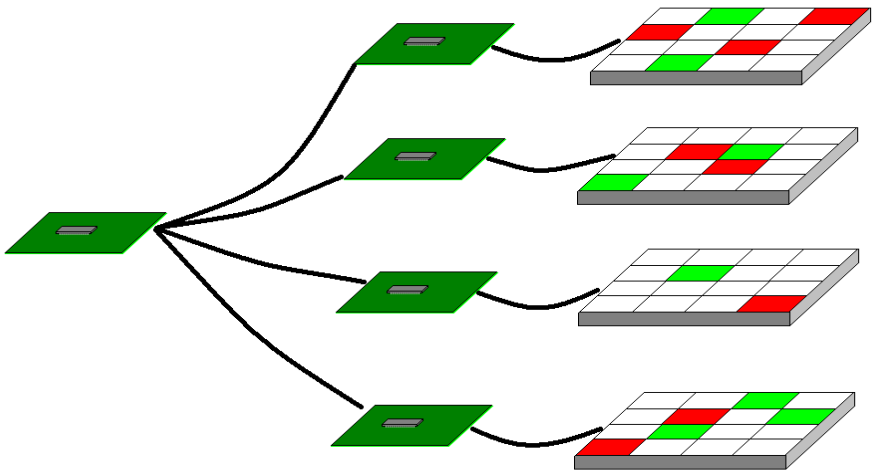
Tässä opinnäytetyössä kuvataan kehitysprosessin kulkua alkaen ensimmäisistä ajatuksista ja päättyen pelin testaukseen. Näiden ääripäiden välissä käsitellään laitteiston suunnittelu, osien valmistus ja kokoonpano, komponenttivalinnat ja niiden hankinta, ohjelmistokehitys, kohdatut ongelmat sekä työn aikana tehdyt havainnot ja päätelmät.

3 TAUSTATIETOA

3.1 Ristinolla pelinä

Ristinollapelin ideana on muodostaa omista pelimerkeistä riittävän mittainen rivi, jonka vastustaja pyrkii omilla pelimerkeillään estämään. Peliä pelataan sijoittamalla omia pelimerkkejä ruudukkoon vuorotellen vastustajan kanssa. Yleisesti käytettyjä pelimerkkejä ovat risti ja nolla, joista pelin nimikin on lähtöisin. Normaalisti peliä pe-lataan kaksiulotteisella pelilaudalla, jolloin voittamiseen hyväksyttyjä rivejä ovat vaaka-, pysty- ja viistosuuntaiset rivit. Opinnäytetyön tapauksessa mukaan tulee myös kolmas ulottuvuus, joka lisää hyväksyttyjen rivien määrää huomattavasti monimutkaistaen alunperin hyvin yksinkertaista ja nopeatempoista peliä. Verrattuna kaksiulotteiseen ristinollaan, vaatii kolmiulotteinen versio pelaajalta suurempaa keskittymiskykyä, ja vastustajan pelitaktiikan hahmottaminen on laajempien pelimahdollisuuksien johdosta vaikeampaa.

Toteutettu laite rakentuu neljästä kosketuskalvolla päällystetystä pelitasosta, joilla ristiä ja nollaa indikoidaan kaksivärisillä ledeillä, jolloin pelaajan valitsemat ruudut hohtavat joko punaisina tai vihreinä.



Kuva 1: Laitteen rakenne

3.2 Atmel AVR mikro-ohjaimet

3.2.1 Historiaa

AVR-tuoteperhe on yhdysvaltalaisen puolijohdevalmistaja Atmel:in markkinoille kehittämä jatkuvasti laajeneva mikro-ohjainmallisto. Ensimmäiset AVR-perheeseen kuuluvat ohjaimet ilmestyivät vuonna 1996 ja mullistivat alaa olemalla ensimmäisiä kuluttajahintaluokan mikro-ohjaimia, joissa oli integroituna uudelleenohjelmoitavaa flash-ohjelmamuistia normaalin kertaalleen ohjelmoitavan sijaan.

Alunperin AVR-tuotteissa käytetyn perusarkkitehtuurin kehittivät norjalaiset Alf-Egil Bogen ja Vegard Wollan ja myivät sen lopulta Atmelille, jonka leivissä he kehittivät sitä edelleen. Projekti kulki nimellä Alf and Vegard RISC architechture eli lyhennettynä AVR, joksi nimi vakiintui valmiiden piirien tullessa tuotantoon. /1/

Ajan mittaan on valmistettu erilaisia AVR-sukupolvia. Ensimmäiset mallit kuuluivat AT90x-sarjaan, joka sittemmin korvattiin Attiny-sarjalla ja malliston yläpäätä laajennettiin monipuolisemmalla ja tehokkaammalla Atmega-sarjalla. Viime vuosina mallistoa on lisätty erilaisilla sovelluskohtaisilla ratkaisuilla, kuten CAN-väyliä ja USB-liitännän omaavia malleja sekä erityisesti ajoneuvokäyttöön ja erilaisiin valaisusovelluksiin räätälöityjä erikoismallisarjoja.

3.2.2 Mikro-ohjaimen tekniikkaa

Teknisesti tarkasteltuna AVR-mikro-ohjaimet ovat kahdeksanbittisiä yhdelle piirille integroituja RISC-prosessorilla varustettuja mikro-ohjaimia. Prosessorin perusarkkitehtuuri edustaa Harvard-mallia, jossa käskyille ja datalle on omat fyysiset linjansa, toisin kuin esimerkiksi von Neuman-mallissa, jossa käskyt ja data käyttävät samaa yhteistä väylää. Von Neumann-arkkitehtuurin mukainen prosessori voi tehdä ainoastaan yhtä asiaa kerrallaan: joko lukea käskyjä tai suorittaa datan luku- tai kirjoitustoimintoja. Koska Harvard-arkkitehtuurissa on erilliset linjat käskyille ja datalle, voidaan lukea käskyjä ja käsitellä dataa samanaikaisesti, josta seuraten samalla kellotaajuudella voidaan suorittaa enemmän käskyjä monimutkaistamatta itse prosessorin rakennetta. Tämä onkin kyseisen arkkitehtuurin valtti mikro-ohjainmarkkinoilla, koska piirejä käytetään usein kannettavissa laitteissa, joissa tehoa halutaan säästää käyttämällä pieniä kellotaajuuksia.

Arkkitehtuurin ohella toinen AVR:n valtti on sen tehokas käskyrakenne. Lähes kaikki sen RISC-käskykokoelman käskyt suoritetaan yhdellä kellojaksolla, joka on huomattava etu verrattaessa esimerkiksi Microchipin kilpailevaan PIC-perheeseen, jossa käsky suoritetaan nopeimmillaan neljällä kellojaksolla.

Koska AVR-piirit ovat yhdelle piirille integroituja mikro-ohjaimia, on niissä prosessorin lisäksi kaikki tarvittavat oheis- ja liitäntäpiirit itsenäisen kokonaisuuden luomiseksi. Jokaisessa AVR-piirissä onkin RISC-prosessorin lisäksi tarvittava määrä ohjel-

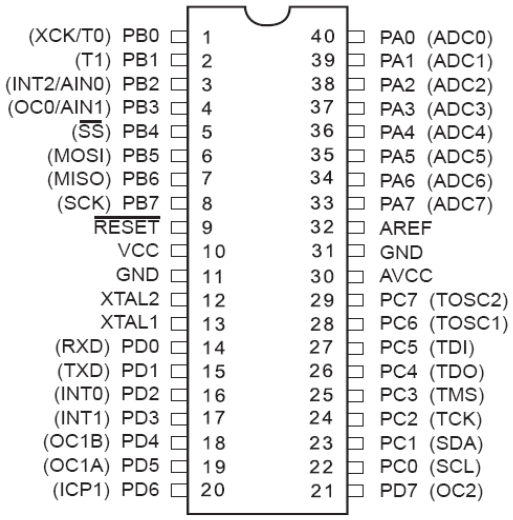
mamuistia ja keskusmuistia. Lisäksi useimmissa malleissa on pieni määrä uudelleenohjelmoitavaa eeprom-muistia erilaisiin käyttösovelluksen vaatimiin tallennuksiin. RISC-prosessori tukee myös keskeytysvektoreita, jolloin ohjelman aikakriittiset osat voidaan suorittaa ajallaan ilman ohjelman vaivalloista optimointia. Lisäksi piirit on varustettu sisäisellä kellokiteellä, joten ulkoista kellopulssia tai kidettä ei tarvita ellei kellotaajuuden tarvitse olla suuri tai erityisen tarkka. Jokaisessa piirissä on myös tietty määrä ohjelmoitavia I/O linjoja mahdollisine sisäisine ylösvetovastuksineen. Näiden kaikille malleille yhteisien piirteitten lisäksi eri malleissa on lukuisia erilaisia laitteistolla toteutettuja ominaisuuksia, joista enemmän seuraavassa kappaleessa.

3.2.3 Oheislaitteet ja liitännät

Piirien yleisimpiin oheislaitemoduuleihin lukeutuvat erilaiset tiedonsiirtoliitännät, kuten RS-232, SPI ja I²C, josta Atmel käyttää nimeä TWI, Two Wire Interface. Näillä liitännöillä on mahdollista liittää järjestelmä useimpiin saatavilla oleviin oheislaitteisiin (tietokoneisiin, muistipiireihin, erilaisiin anturiyksiköihin toisiin mikro-ohjaimiin, lähes mihin tahansa). Tietoliikennelaitteiden lisäksi yleisiä oheisliitäntöjä ovat a/d-muuntimet, analogiset vertailijat ja PWM eli pulssinleveysmodulaatiomoduulit, joilla voidaan instrumentoida ja säätää piirin ulkopuolisia suureita, kuten mitata analogisia jännitteitä ja säätää analogista lähtöjännitettä. Professionaalisempaa kehitystyötä ajatellen on oleellista myös mainita joidenkin Atmega-mallien sisältämä JTAG-liitäntä. JTAG-liitäntää voidaan käyttää paitsi piirin ohjelmointityökaluna, myös ohjelman toiminnan tarkasteluun. JTAG-liitännällä on ominaisuus nimeltä Boundary-Scan, joka siirtää piirin kaikkien I/O-linjojen ja rekisterien sisällön mahdolliseen debuggeriin.

AVR-tuoteperhe on alunperin suunniteltu ohjelmoitavaksi C-kielellä. Saatavana on ilmaisia kehitysympäristöjä ja kääntäjiä sekä useita kaupallisia sovelluksia. Lisäksi Atmel tarjoaa käytettäväksi ilmaisen AVR-Studio-nimisen kokonaisuuden, jossa on perinteisten tekstieditorin ja C-kääntäjän lisäksi myös simulaattori ja JTAG-debug-

geri, jotka mahdollistavat lähes reaaliaikaisen ohjelman toiminnan tarkastelun.



Kuva 2: Erään AVR:n pinnikytkentä

3.3 RS-232 sarjaliitäntä

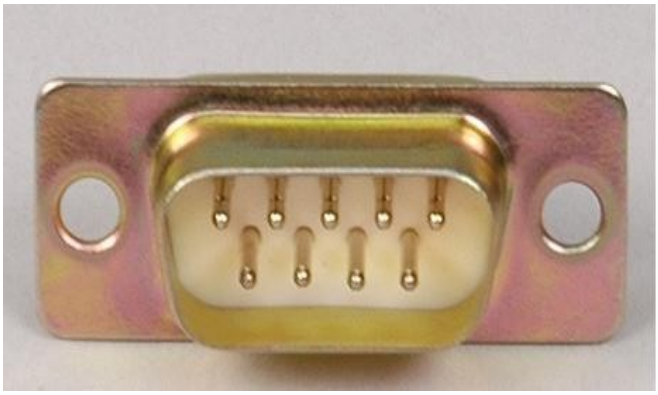
3.3.1 Yleistä

RS-232 on standardi, joka määrittelee sarjamuotoisen tiedonsiirron binäärimuodossa kahden päätelaitteen välillä. Alunperin EIA:n vuonna 1969 standardoima RS-232C tunnetaan myös ITU-T standardina V.24. Standardi määrittelee myös käytettävät liittimet, niiden kytkennän, signaalien jännitetasot ja jännitteiden muutosnopeuden. /2/

Päätelaitteista käytetään nimityksiä DTE, data terminal equipment ja DCE, data circuit-terminating equipment. DTE on useimmiten tietokone taikka muu laite, jota DCE palvelee. Siirrettävä data kulkee TxD eli lähetys ja RxD eli vastaanottolinjoissa. Molemmilla päätelaitetyypeillä on omat liittimien kytkentäjärjestykset, joten ns. suora välikaapeli riittää kytkemään kaksi eri tyyppistä laitetta toisiinsa. Liitettäessä kaksi

samantyyppistä laitetta, tarvitaan nk. ristiinkytketty kaapeli, jolloin linjat kytkeytyvät oikein.

Alkuperäinen käyttötarkoitus RS-232:lle oli käyttöpäätteet eli TTY:t, joilla voitiin käyttää unix-järjestelmää tai tulostimia. Liitäntää on käytetty laajalti erilaisissa tietoliikenne ja automaatiosovelluksissa, ja se on vanhimpia edelleen käytössä olevia liitäntöjä PC-koneissa. Yleisimmäksi liittimeksi on vakiintunut 9-pinninen DB9-liitin.



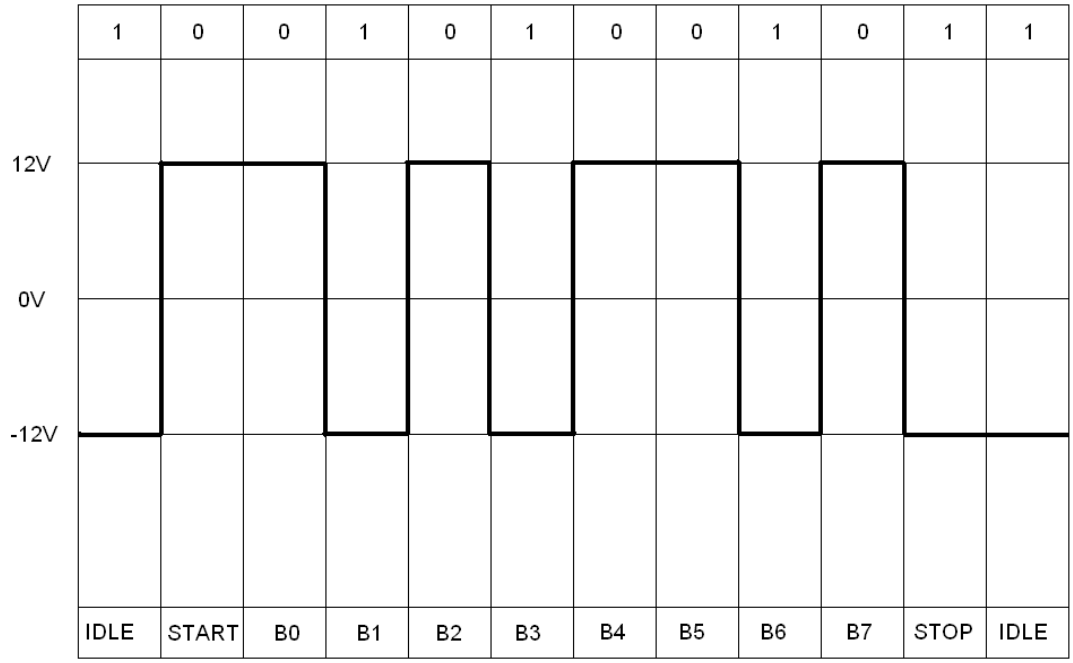
Kuva 3: DB9-liitin

3.3.2 Johtimet ja tiedonsiirto

Yksinkertaisimmassa sovelluksessa tarvitaan ainoastaan kolme johdinta, lähetys-, vastaanotto- ja maajohtimet. Näin toteutettu liitäntä kykenee siirtämään dataa molempiin suuntiin, mutta liitännän tilaa ei pysty päättelemään ilman ohjelmallista protokollatoteutusta molempiin päätelaitteisiin. Tämän vuoksi standardiin kuuluu erilaiset kättelylinjat, joilla linjan tilaa pystytään seuraamaan ja toteuttamaan ns. vuonohjaus, jolloin päätelaitteet kertovat toisilleen, onko mahdollista siirtää uutta dataa.

Liitännän kehysrakenne soveltuu tavupohjaisen datan, kuten ASCII-merkkien lähetykseen. Itse kehys koostuu aloitusbitistä, yleensä 6-9 bitistä dataa, mahdollisesta pariteettibitistä ja lopetusbiteistä, joita voi olla yhdesta kahteen kappaletta. Standardi

ei kuitenkaan määrittele merkkien koodausta eikä virheenkorjausta, joten ne jäävät toteutettaviksi ohjelmallisesti. Varsinainen data siirretään linjalle vähiten merkitsevä bitti ensimmäisenä.



Kuva 4: Yhden merkin siirto

Vuonohjauksen toteutukseen standardi tarjoaa kaksi 0-aktiivista nk. kättelylinjaa, CTS, clear to send ja RTS, request to send. DTE informoi DCE:tä valmistautumaan datan vastaanottoon asettamalla RTS-signaalin 0-tilaan. Mikäli DCE on kykenevä vastaanottamaan uutta dataa, vastaa se kutsuun asettamalla CTS-signaalin 0-tilaan ja data voidaan siirtää. Vuonohjauksella varustettu liitäntä vaatii siis minimissään viisi johdinta.

RS-232 on asynkroninen liitäntä, josta seuraten molempien päätelaitteiden lähetys- ja vastaanottopiirit on oltava tahdistettuna tarkasti samalla lähetysnopeudella. Mikäli päätelaitteiden nopeudet poikkeavat toisistaan, voidaan vastaanotetut bitit tulkita väärin. Tämän välttämiseksi joissakin laitteissa on omat johtimet kellosignaalin välittämistä varten.

Näiden yleisempien signaalien lisäksi standardiin kuuluu useita muitakin ohjaussignaaleja, jotka eivät ole oleellisia liitännän normaalikäytössä eikä nykyään käytössä olevissa db9-liittimissä ole edes riittävästi johtimia niiden kytkemiseen.

Käytettävät jännitetasot ovat ±15V, joista ±3V on nk. häiriönpoistoalue jota ei huomioida. Negatiivinen jännite kuvaa 1-tilaa ja positiivinen 0-tilaa. PC-koneissa RS-232- eli sarjaportti onkin ainoa laite jota varten virtalähteessä on lähtö -12V:n jännitteelle. Nykyisin USB-liitäntä onkin korvannut sarjaliitännät varsinkin kannettavissa tietokoneissa sen matalamman jännitetason ja yksinkertaisemman liittimen vuoksi. Ohjelmiston yksinkertaisuudessa sarjaportti on kuitenkin ylivoimainen.

3.4 I²C eli TWI-liitäntä

3.4.1 Yleistä

I²C on Philipsin kehittämä useaa isäntälaitetta tukeva hitaaseen tiedonsiirtoon tarkoitettu väylä. Sovelluskohteita ovat mikro-ohjaimet, muistipiirit, erilaiset anturiyksiköt, AD- ja DA-muuntimet, tietokoneiden hitaat oheislaitteet kuten lämpöanturit ja kulutuselektroniikka yleensä.Väylä käyttää tiedonsiirtoon kahta ylösvetovastuksilla käyttöjännitteeseen kytkettävää kaksisuuntaista signaalijohdinta, SCL eli serial clock ja SDA eli serial data. Ylösvetovastukset ovat pakollisia, koska väylään liitettävien laitteiden liitäntäpinnit ovat avoinkollektori-lähtöjä, joilla väylän voi vetää alas mikä tahansa siihen kytketty laite.

Toimiva I²C järjestelmä koostuu yhdestä tai useammasta master- eli isäntälaitteesta sekä slave- eli orjalaitteista. Tiedonsiirto väylällä tapahtuu tavupohjaisesti ja isäntälaite muodostaa väylän kellosignaalin SCL-johtimeen, jonka avulla tiedonsiirto tahdistetaan. Jokaisella orjalaitteella on oma 7 bittiä pitkä osoitteensa, joka on oltava

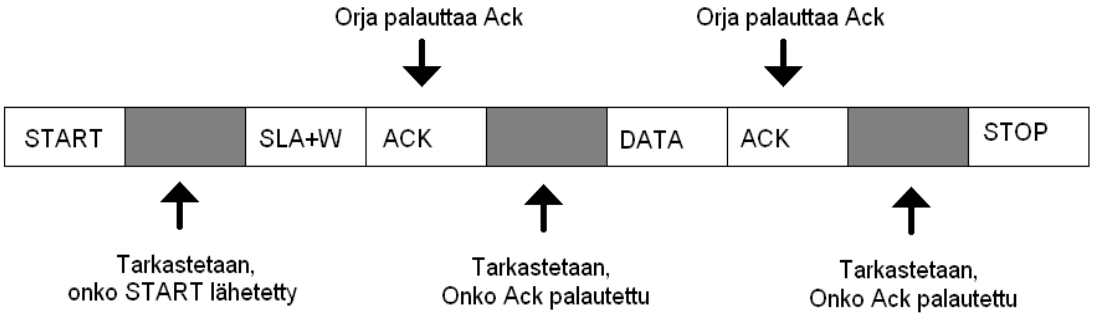
välillä 1-127, 0 on varattu broadcast-osoitteeksi jolla osoitetun datan kaikki orjalaitteet vastaanottavat tarvittaessa. /3/

Normaalissa yhden isännän järjestelmässä isäntä- ja orjalaitteet toimivat molemmat kahdessa eri moodissa, eli moodeja on yhteensä neljä. Master Transmitter-moodissa isäntälaite syöttää dataa SDA-linjalle. Tällöin orjalaite on Slave Receiver-moodissa ja valmistautuu tarvittaessa vastaanottamaan isäntälaitteen lähettämää dataa. Näin toteutetaan datan kirjoitusoperaatio. Lukuoperaatiossa isäntälaite on puolestaan Master Receiver-moodissa ja valmiina vastaanottamaan Slave Transmitter- moodissa olevan orjalaitteen lähettämää dataa. Kaikissa moodeissa isäntälaite muodostaa kellosignaalin SCL-linjalle.

3.4.2 Tiedonsiirtotapahtuma

Isäntälaite aloittaa tiedonsiirtotapahtuman lähettämällä START-komennon, joka tapahtuu vetämällä SDA-linja alas SCL-linjan ollessa ylhäällä. Tällöin orjalaitteet alkavat kuunnella linjaa. Seuraavaksi isäntälaite vetää scl-linjan alas ja siirtää SDA-linjalle haluamansa orjalaitteen 7-bittisen osoitteen yksi bitti kerrallaan. Ensimmäisenä SDA-linjalle siirtyy osoitteen eniten merkitsevä bitti, jonka jälkeen isäntälaite päästää SCL-linjan takaisin ylös. Kaikki orjalaitteet lukevat bitin siirtorekisteriinsä. Sama toistuu, kunnes osoitteen kaikki seitsemän bittiä on lähetetty. Tämän jälkeen isäntälaite lähettää kahdeksantena bittinä Read/Write bitin, joka kertoo onko tapahtuma datan kirjoitus- vai lukuoperaatio. Jokainen väylällä oleva orjalaite vertaa vastaanottamaansa osoitetta omaansa, mikäli osoitteet vastaavat toisiaan, jää laite Read/Write-bitistä riippuen vastaanottamaan tai lähettämään dataa ja palauttaa väylälle ACK-signaalin vetämällä SDA-linjan alas yhdeksännen SCL-kellojakson aikana. ACK-signaali kertoo isäntä-laitteelle orjalaitteen olevan valmis jatkamaan. Orjalaite voi myös palauttaa NACK-signaalin mikäli se ei jostain syystä ole valmis. Mikäli osoitteet eivät täsmää, irrottautuu orjalaite väylästä kunnes isäntälaite vapauttaa väylän lähettämällä STOP-komennon, joka tapahtuu päästämällä SDA-linja ylös SCL-linjan olles-

sa ylhäällä. Väylä on jälleen vapaa ja uusi tapahtuma voidaan aloittaa. Vaihtoehtoisesti STOP-komennon sijasta voidaan antaa REPEAT START-komento, jolloin uusi siirtotapahtuma alkaa ilman väylän vapauttamista. /3,5/



Kuva 5: Tyypillinen tiedonsiirtotapahtuma

3.4.3 Ominaisuuksia, vaatimuksia ja rajoituksia

Useimmat I²C-laitteet tukevat standardia 100 kbit/s tiedonsiirtonopeutta, mutta myös 400 kbit/s nopeus on yleinen uudemmissa laitteissa. Nopeimpia sovelluksia ovat 1 mbit/s Fast mode plus ja 3.4 mbit/s High speed mode.

Väylään liitettävien laitteiden määrää rajoittaa paitsi lyhyt 7-bittinen osoiteavaruus, myös linjalle annettu maksimi kapasitanssi, joka on 400 pF. Kapasitanssin vaikutuksia voidaan kuitenkin minimoida käyttämällä suuruudeltaan sopivia ylösvetovastuksia. Atmelin suosittelema signaalijohtimien ylösvetovastus käyttöjännitteeltään 5V:n järjestelmään on 4.7 kilo-ohmia. Mikäli signaalijohtimet ovat huomattavan pitkät tai laitteita on liitetty useita, voidaan ylösvetovastuksia pienentää pysyen kuitenkin järkevissä rajoissa, jotta piirien avoinkollektori-lähdöt eivät vahingoittuisi. Atmel suosittelee myös mikro-ohjaimissaan käytettävän vähintään kymmenkertaista kellotaajuutta SCL-linjan kellotaajuuten verrattuna, jotta signaalit voitaisiin tulkita oikein.

3.5 Kosketuskalvot

3.5.1 Yleistä kosketuskalvoista

Kosketuskalvo on osoitinlaite, joka tulkitsee käyttäjän kosketuksen ja sen paikan kalvolla. Sitä voidaan käyttää monissa tietotekniikan sovelluksissa hiiren sijasta ja se luo moniin kannettaviin laitteisiin näppäimiä luonnollisemman käyttöliittymän. Tällaisia laitteita ovat esimerkiksi älypuhelimet, PDA-laitteet ja GPS-navigaattorit. Kehittyneimmät kosketuskalvojärjestelmät kykenevät myös useamman yhtäaikaisen kosketuksen käsittelyyn sekä tunnistavat kalvoa painavan voiman suuruuden. Lähes poikkeuksetta kosketuskalvo asennetaan jo tehtaalla suorapintaisen näyttöelementin päälle muodostaen kosketusnäytön, mutta irrallisiakin kalvoja on rajallisesti saatavilla. Useimmiten nämäkin kalvot jälkiasennetaan jo aiemmin hankittuihin näyttöihin.

Suurin osa nykyisistä kosketuskalvoteknologioista on alunperin kehitetty jo 1970- ja 1980-luvuilla. Teknologiat olivat hyvin patentoituja ja vasta viime vuosina vapautuneet yleiseen käyttöön, josta johtuen tuotteiden hinnat ovat käytön laajentuessa laskeneet.

Pelkän kalvon lisäksi tarvitaan ohjain, joka muuntaa kalvon muodostaman signaalin ohjattavan laitteen ymmärtämiksi koordinaateiksi. Ohjaimien yleisimmät liitännät ovat RS-232 eli sarjaliitäntä, SPI-liitäntä sekä PS/2.

3.5.2 Yleisimmät kosketuskalvotyypit

Asennettaessa kosketuskalvoa näytön päälle on otettava huomioon, että kalvotyypistä riippumatta suurin mahdollinen läpinäkyvyys on 70-85%, joka aiheuttaa kuvan sumenemista.

Resistiiviset kosketuskalvot ovat tällä hetkellä yleisimpiä johtuen niiden kilpailukykyisestä hinnasta, ja saatavilla on suuriakin kokoja. Pöly ja muu ulkoinen lika ei myöskään vaikuta itse kalvon toimintaan. Tyypillisiä heikkouksia ovat vain keskinkertainen läpinäkyvyys ja vaurioitumisen mahdollisuus kalvon osuessa terävään esineeseen.

Resistiivinen kosketuskalvo koostuu kahdesta erittäin ohuesta resistiivisestä metallikerroksesta, jotka ovat hiuksenhienosti toisistaan erillään. Painettaessa kalvoa, koskettavat metallikerrokset toisiaan, jolloin muodostuvasta signaalista voidaan päätellä kosketuspaikka.

Kapasitiivinen kosketuskalvo on päällystetty materiaalilla, joka johtaa kosetuskalvon pinnan yli tasaisen sähkövirran. Virta kulkee sekä pysty- että vaakasuunnassa. Kosketettaessa kalvoa sormella tai johtavalla esineellä vaikuttaa ihmiskehon kapasitanssi kalvon pinnalla kulkevaan sähkövirtaan, josta voidaan päätellä kosketuspaikka.

Kapasitiiviset kosketuskalvojen läpinäkyvyys on erittäin hyvä ja ulkoiset tekijät eivät suuresti vaikuta sen toimintaan. Kapasitiivisten kalvojen haittapuolina on monimutkaisen ohjauselektroniikan aiheuttama kallis hinta.

Surface Acoustic Wave eli SAW-tyyppiset kosketuskalvot käyttävät kosketuksen havaitsemiseen kosketuspinnan yli lähetettävää ultraääniaaltoa. Kalvoa kosketettaessa osa aallosta vaimenee, jolloin kosketusken paikka voidaan laskea aallon muutoksesta. SAW-tyyppiset kosketuskalvot ovat mekaanisesti herkkiä ja ulkoinen lika ja sormenjäljet haittaavat toimintaa. /4/

4 LAITTEISTO

4.1 Laitteiston suunnittelu

4.1.1 Suunnittelun lähtökohdat

Työssä käytettävän laitteiston ja elektroniikan suunnittelussa oli otettava huomioon tietyt lähtökohdat ja vaatimukset. Jokaisella piirikortilla pitää olla tarvittavat liitännät, piirikortille asennetussa mikro-ohjaimessa oltava riittävästi ohjelmamuistia sekä sen on mahdollistettava oikein ajoitetun ohjelman ohjelmointi ja suoritus. Työssä päädyttiin käyttämään Atmelin ATmega32-mikro-ohjainta sen tarjoamien liitäntöjen monipuolisuuden ja suurehkon ohjelmamuistin takia.

4.1.2 I/O-linjojen lisääminen

Työssä ensimmäinen ratkaisua vaativa ongelma oli mikro-ohjaimien I/O-pinnien vähäisyys. Pelitasoilla valittujen ruutujen tilaa kuvaavia kaksivärisiä ledejä on jokaisella tasolla 16 kappaletta, joista jokaisella kaksi oman ohjauksen vaativaa anodia. Käytetyissä mikro-ohjaimissa on ainoastaan 32 I/O-pinniä, joista osa käytetään muihin liitäntöihin. Ainoaksi järkeväksi ratkaisuksi jäi jonkinlaisen oheislogiikan käyttö, jolloin muutamalla data- ja ohjauslinjalla voidaan laajentaa lähtöpinnien määrää. Lopulliseksi ratkaisuksi päätyi kahdeksanbittiset D-kiikut, joita tarvittiin yhteensä neljä kappaletta jokaiseen pelitasojen ohjainlevyyn. Alunperin tarvittavat 32 I/O-linjaa saatiin näin vähennettyä 13:een ja mikro-ohjaimien I/O-pinnejä jäi näin ollen vielä vapaaksi. D-kiikkujen ohjaukseen käytetään viittä I/O-linjaa, neljää enable-linjaa ja yhtä kello-linjaa. Data tarvitsee ainoastaan yhden kahdeksan bittiä leveän linjan.

D-kiikun ollessa enabloituna, päivittyy datalinjan sisältö kiikun lähtöön aina kellolinjan nousevalla reunalla ja lähtö pysyy muuttumattomana kunnes se uudelleen päivitetään. Näin saadaan pelitasolle toimiva pelitilanteen indikointi ilman, että pelitason ohjaimen tarvitsee ohjata ledejä jatkuvasti eikä havainnointi vaikeudu, vaikka pelitason ohjain suorittaisikin vaikkapa pitkäkestoista keskeytysrutiinia.

Valittaessa oikeanlaisia porttipiirejä kytkentään, huomio kiinnittyi kunkin kiikkupiirin perään kytkettävien ledien ottamaan virtaan. Käytettyjen 74HC574-piirien suurin antovirta on 35mA lähtöä kohti, joka riittää mainiosti normaalille ledille, jolle yleensä suositellaan 20mA:n virtaa. Piirin suurin yhtäaikainen antovirta on piirin valmistajasta riippuen 60-75mA. Otettaessa huomioon, että piirin perään kytketyistä ledeistä ainoastaan neljä voi pelitilanteessa palaa yhtäaikaa, voidaan ledien läpi kulkevaan virtaan vaikuttaa etuvastuksien resistanssia muuttamalla. Tässä päädyttiin käyttämään 180 ohmin vastuksia, joilla ledien läpikulkeva virta saatiin asettumaan noin 15mA:iin ja kiikkupiirien virranantokykyä ei ylitetty.

4.1.3 Muut sovitinpiirit

Kosketuskalvojen ohjaimet kytketään pelitasojen ohjainlevyihin RS232-liitännällä, jonka jännitetasot eivät ole yhteensopivat mikro-ohjaimen TTL-tasoisen signaalin kanssa. Tarvitaan jännitesovitin, tässä tapauksessa MAX232, joka on MAXIM:n valmistama RS232-jännitetasosovitin kahdelle RS232-linjalle, joista tässä yhteydessä tarvitaan vain toista. MAX232 tarvitsee ainoastaan neljä ulkoista kondensaattoria, joita se käyttää jännitteen korottamiseen charge pump-menetelmällä ja toimii 5V:n jännitteellä.

RS232-liitäntää varten pelitasojen ohjainlevyillä on 9-pinninen D-liitin. Normaalien lähetys- ja vastaanottolinjojen lisäksi tässä tapauksessa kosketuskalvon ohjaimelle siirretään käyttöjännite D-liittimen 1-pinnin kautta, jotta vältytään ylimääräisiltä käyttöjännitejohtimilta.

4.1.4 Virtalähde ja häiriöt

Jokaisella piirilevyllä on oma käyttöjännitteen suodatus ja regulointi. Tämä ominaisuus valittiin, koska pelitasojen ledien yhteenlaskettu virrankulutus on niin suuri, että yhdellä regulaattorilla toteutettu virtalähde olisi kuormittunut pitempiaikaisessa käytössä liiaksi. Tämän lisäksi häiriöiden mahdollisuus kasvaa kasvatettaessa etäisyyttä reguloinnin ja kuorman välillä. Tällä menettelyllä voidaan regulaattoreissa käyttää pienempiä jäähdytysprofiileja ja suodatuksessa käytettäviä kondensaattoreita voidaan pienentää.

Itse regulaattorina käytetään kenties yleisintä 78-sarjan 5V:n versiota. Se tarvitsee ulkoisiksi komponenteikseen ainostaan muutamia kondensaattoreita suodattamaan suurimpia häiriöitä ja on lämpötila- ja oikosulkusuojattu.

Koska kytkennät ovat digitaalisia ja mikro-ohjaimet toimivat verraten suurella taajuudella, on jokaisen digitaalisen piirin käyttöjännitepinneihin lisättävä pienet häiriönpoistokondensaattorit mahdollisimman lähelle itse piiriä. Nämä kondensaattorit saavat olla tyypiltään joko keraamisia tai muovikondensaattoreita niiden elektrolyyttikondensaattoreita parempien korkeataajuusominaisuuksiensa takia.

4.1.5 Liitäntä käyttöliittymälle

Jotta pelin pelaaminen olisi jossakin määrin jouhevaa, päätettiin laitteistoon lisätä muutamia nappeja, joilla voi vaikkapa aloittaa uuden pelin. Nappulat on sijoitettu erilliselle pienelle kortille, joka on kytketty viidellä johtimella isäntälaitteeseen. Käyttöjännitteen lisäksi johtimissa kulkee kaksi datalinjaa ja yksi keskeytyslinja, jotka on kytketty isäntälaitteen mikro-ohjaimeen. Painettaessa nappia, generoituu ohjaimessa ulkoinen keskeytys, jolloin nappien tila luetaan ja toimitaan sen mukaan.

4.1.6 Piirilevyjen suunnittelu ja valmistus

Kytkentäkaaviot ja piirilevyt laadittiin saksalaisen CadSoftin Eagle-ohjelmistolla, jolla suunnittelu on melko yksinkertaista. Ensin luodaan kytkentäkaavio Schematic Editorilla asettelemalla symboliset komponentit ohjelman työtilassa haluttuun järjestykseen. Itse kytkennän luominen tapahtuu liittämällä komponenttien jalat loogisesti toisiinsa johdotustyökaluilla. Kun kaikki komponentit on lisätty ja johdotettu sekä komponentit nimetty osuvasti, voidaan siirtyä piirilevyn asetteluun.

Kun kytkentäkaavio on saatu tyydyttävään kuntoon, voidaan aloittaa piirilevyn suunnittelu käynnistämällä Eaglen Board Editor. Suunnittelu kannattaa aloittaa asettamalla työtilan mittayksiköt sekä piirilevyn tyyppi ja rakenne oikein.

Kun työskentely-ympäristö on kunnossa, voidaan alkaa varsinainen suunnittelu asettelemmalla sijoitettavat komponentit siten, että kytkentä voidaan toteuttaa järkevästi ja mahdolliset liittimet ovat sellaisessa paikassa, jossa niitä voidaan esteettä käsitellä. Kun komponentit on aseteltu, kannattaa mielessään tarkistaa, ettei esimerkiksi komponenttipuolelta juotettavien komponenttien juottaminen muutu mahdottomaksi liian tiheän komponenttiasettelun vuoksi.

Seuraava ja työläin askel piirilevyn suunnittelussa on johdotusten tekeminen. Eaglen Board Editor sisältää myös Autoroute-toiminnon, joka osaa oikeanlaisilla säännöillä avustettuna luoda automaattisesti johdotuksia. Käytännössä Autoroute-toiminto ei toimi loogisesti ja ainoaksi vaihtoehdoksi jää johdotusten luonti käsin.

Tehtäessä johdotuksia eli ”vetoja”, tulee huomioida piirinlevyn valmistusmenetelmä. Työssä käytetyt piirilevyt valmistettiin jyrsimällä Bungard CCD-piirilevyjyrsimellä, joten huomiota tuli kiinnittää johdotusten paksuuteen ja johtimien välimatkaan. Koska jyrsiminen on mekaaninen työstötapa, ei voida käyttää kovinkaan ohuita johtimia,

koska molemminpuolinen jyrsiminen irrottaa helposti koko johtimen laminaatista eikä ohuen johtimen mekaaninen kestävyys ole muutenkaan kehuttava. Hyvin tiheästi sijoitetut juotostäplät ovat puolestaan erittäin haasteellisia juottaa, koska jyrsitty ura johtimien välissä on hyvin ohut ja tina muodostaa helposti siltoja kahden kuparipinnan välille. Mikäli kytkentä on johdotukseltaan hiemankaan monimutkaisempi, joutuu usein käyttämään kaksipuolista piirilevyä. Tällöin pitää huolehtia, että johdotukset ja komponentit ovat oikeilla tasoilla ja kaikki pintatasolla olevat juotostäplät voidaan myös juottaa.

Työssä valmistetut pelitasojen ohjainlevyt ovat kaksipuolisia, ja isäntälaitteen ohjainlevy on yksipuolinen johtuen sen yksinkertaisuudesta. Yksi- ja kaksi-puolisten piirilevyjen valmistamisen seuraavat askeleet poikkeavatkin hieman toisistaan.

Kun kaikki johdotukset on tehty ja piirilevy näyttää muutenkin hyvältä, pitää siitä luoda CAM-tiedostot jyrsintäohjelmistoa varten. Eagle sisältää oman Cam Processorin, jolla voidaan tuottaa erilaisia tiedostoja CAM-ohjelmia varten. Jokaista työstövaihetta varten pitää luoda omat CAM-tiedostot. Kaksipuoliset piirilevyt tarvitsevat siis tiedoston pintakuparin jyrsintään, pohjakuparin jyrsintään, reikien poraukseen ja piirilevyn sahaukseen. Yksipuolisissa pintakuparin tiedot jäävät luonnollisesti pois. Tiedostot tallennetaan tietyssä tiedostomuodossa, jyrsintätiedot HPGL-formaatissa ja poraustiedot EXCELLON-formaatissa. HPGL-formaatti sisältää yksikäsitteiset mittayksiköt, joten ne toistuvat oikein ohjelmasta toiseen. EXCELLON puolestaan voidaan tulkita eri ohjelmissa eri tavalla, joten oikeat mittayksiköt ja skaalat pitää olla tiedossa tai ne pitää kokeilla kunnes oikea löytyy. Yleensä mittayksikkönä on tuumat neljällä desimaalilla ilmoitettuna.

4.1.7 Työstötiedostojen tekeminen

Kun tiedostot on luotu, on aika siirtyä varsinaisen CAM-ohjelmiston ääreen. Työssä käytettiin Bungardin piirilevyjyrsimeen kuuluvaa ISOCAM-ohjelmaa työstötiedosto-

jen tekoon. ISOCAM:lla avataan Eagle:lla tehdyt CAM-tiedostot ja asetellaan ne työskentelyalueelle haluttuun asentoon. Tässä vaiheessa on mahdollista poistaa työstövaiheista ylimääräisiä osia, kuten reikiä, turhia juotostäpliä ja niin edelleen. Kun oikeat mittakaavat ja työstövaiheiden kohdistukset on tarkastettu, on aika ajatella kolmiulotteisesti. Koska jyrsin jyrsii vain piirilevyn päälipintaa, on pohjakuparointeja työstettäessä piirilevy käännettävä ylösalaisin. Tämä pitää ottaa huomioon ISOCAM:ia käytettäessä ja pohjakuparia kuvaava työstökuva on ohjelmassa peilattava y-akselin suhteen (myös piirilevy pitää kääntää y-akselin ympäri).

Kun kaikki työstökuvat on kohdallaan, tehdään jokaisesta työstövaiheesta itse jyrsinlaitteen ymmärtämä työstötiedosto. Jokainen jyrsittävä työvaihe valitaan kokonaan ja asetetaan prosessoitavaksi valinnalla ”copy to milling”. Komennolla ”create milling data” tehdään valinnasta jyrsimen ymmärtämää jyrsinkoodia. Tässä vaiheessa voidaan valita käytettävät työkalut, työstötarkkuus ja työstön liikeratoihin liittyviä asetuksia. Kun jyrsinkoodi on luotu, voidaan tarkastaa työstöradat mahdollisten virheiden takia. Mikäli jälki tyydyttää, tallennetaan jyrsinkoodi valinnalla ”save milling data”.

Sahauskoodi muodostetaan valitsemalla piirilevyn sahausta kuvaava työstökuva, valinnalla ”create board cut data” voidaan luoda sahauskoodi. Luotu sahauskoodi voidaan nyt tallentaa ja siirtyä tämän vaiheen viimeiseen osaan.

Piirilevyn poraustiedot luodaan porauskuvasta valitsemalla kaikki reijät ja tallentamalla ne. Tallennettaessa valitaan tallennusformaatti joka Bungard-jyrsimelle on millimetreinä viidellä numerolla, joista kaksi desimaaleja. Poraustietoihin ei tässä tapauksessa tallenneta työkalutietoja. Tämän jälkeen siirrytään jyrsimään ja poraamaan.

Tässä vaiheessa tulisi huomioida, että työstettävä piirilevy kiinnitetään työstöpöytään poraamalla piirilevyyn reiät molempiin päihin. Työstöpöydässä on reiät vastaavissa paikoissa. Molemmista reiistä painetaan läpi tiukalla sovituksella oleva metallinen tappi.

Työstettäessä kaksipuolisia piirielevyjä, on oleellista että kiinnityspiste on ainakin y-akselilla sama kuin jyrsimen offset-koordinaatti sekä ISOCAM:n nollakoordinaatti. Näin varmistetaan työstön oikea kohdistuminen käännettäessä piirilevyä.

4.1.8 Piirilevyn jyrsintä käyttäen Bungard CCD-jyrsintä

Piirilevyn jyrsintätapahtuma aloitetaan käynnistämällä RoutePro-ohjelmisto sekä kytkemällä virta jyrsimen ohjausyksikköön. Ohjelman käynnistysvalikossa voidaan vaikuttaa jyrsimen offset-koordinaatteihin, piirilevyn paksuuteen sekä käytettävissä olevaan työkaluvalikoimaan.

Kun offset-koordinaattien on todettu vastaavan piirilevyn kiinnityspisteitä, voidaan aloittaa piirilevyn työstäminen avaamalla aikaisemmin tehty jyrsinkoodi. Tässä vaiheessa ohjelma ilmoittaa piirilevyn työstöalueen millimetrikoordinaatteina. Mikäli tiedot täsmäävät, voidaan alkaa työstäminen komennolla ”start routing”, jolloin ohjelma avaa työstöikkunan, josta selviää suoritettavat työstöliikeradat ja työn eteneminen.

Ennen työstön aloittamista jyrsin pyytää asennettavaksi tiettyä työstöterää. Kun terä on asennettu, säädetään työstösyvyys erillisellä työstökaraan kiinnitettävällä nylon-suulakkeella, jonka pituus on säädettävissä kiertämällä. Suulake liukuu pitkin työstettävän piirilevyn pintaa, jonka vuoksi on tärkeää siirtää työstettävä alue riittävän kauas piirilevyn kiinnitystapeista, jotta vältyttäisiin kontaktilta. Työstöterän kärki on erittäin ohut kartio, joten työstösyvyyttä säätämällä voidaan vaikuttaa leikkuu-uran leveyteen ja terän kestävyyteen. Työssä käytetyt leikkuu-uran leveydet olivat tilanteesta riippuen 0.1-0.3mm.

4.1.9 Reikien poraaminen ja levyn sahaus

Jyrsinnässä käytetty laite poraa myös reiät valmiiksi jyrsittyyn piirilevyyn. Ennen porauksen aloittamista tulee tarkistaa, että poraamiselle osoitettu offset-koordinaatti vastaa jyrsityn kappaleen koordinaatteja ja että piirilevy on työstöpöydällä oikeinpäin. Poraaminen aloitetaan avaamalla aikaisemmin luotu poraustiedot sisältävä tiedosto. Ohjelma ilmoittaa porausalueen koordinaatit ja niiden tulee sopia jyrsityn alueen sisään. Tarkastuksen jälkeen aloitetaan poraaminen komennolla ”start drilling”. Ohjelma avaa työstöikkunan, jossa näkyy porattavien reikien lisäksi edelliset jyrsintätyöt, joten reikien asemointi voidaan lopullisesti varmistaa.

Aloitettaessa poraaminen, ohjelma pyytää asentamaan laitteeseen oikean porausterän. Tässä tilanteessa on huomioitavaa asentaa seuraavaa reikää vastaava terä eikä ohjelman pyytämä, sillä jostakin syystä ohjelmien väliset työkalutiedot eivät pidä paikkaansa. Ohjelma poraa reiät pienimmästä suurimpaan, joten oikean terän valitseminen on melko helppoa.

Terän asentamisen jälkeen asennetaan karaan kiinnitettävä alumiininen poraussuulake, joka on kokoonpainuva ja jousikuormitteinen. Suulake painaa porattaessa piirilevyä kohti työstöpöytää, jolloin se pysyy varmemmin kohdillaan eikä ole vaaraa sen nousemisesta pois kiinnitystapeilta.

Kun kaikki reiät on porattu, voidaan piirilevy sahata irti. Sahausoperaatio aloitetaan avaamalla aikaisemmin luotu sahaustiedosto. Ennen sahauksen aloittamista on tarkistettava että piirilevy on oikein kohdistettu. Sahausta varten jyrsimeen asennetaan varta vasten sahaukseen tarkoitettu terä jossa on varressa leikkaava kuviointi. Tällä kertaa karaan ei kiinnitetä erillistä suulaketta, mutta sahattaessa on suositeltavaa pitää piirilevystä kiinni, ettei se pääse värähtelyn vaikutuksesta nousemaan pois paikaltaan. Sahauksen valmistuttua voidaan valmiin piirilevyn reunoja hieman pyöristää viilaamalla tai hiomalla, samalla silmämääräisesti tarkistaen työstöjälkeä.

4.1.10 Komponenttien hankinta ja levyjen kalustus

Komponenttien hankinnassa todellista päänvaivaa aiheuttivat ainoastaan kosketuskalvot. Irrallisten kalvojen ja ohjainten saatavuus Euroopassa on huono ja toimitusajat maahantuojalta tilatessa saattavat pientilauksissa kasvaa sietämättömän pitkiksi. Lopulta Yhdysvalloista löytyi tarvittavan kokoisia Fujitsun valmistamia kosketuskalvoja ja niihin sopivia ohjaimia suoraan hyllytavarana.

Muut työssä käytetyt komponentit olivat hyvin yleisiä ja ne tilattiinkin Elfalta Suomesta ja lyhyen tomitusajan jälkeen levyjen kalustus voitiin lopulta aloittaa. Pelitasojen ohjaimissa olevia d-kiikkuja ei kuitenkaan ollut saatavana, joten ne korvattiin lähes vastaavilla d-veräjillä, joiden käyttöönottamiseksi ainoastaan yhtä aliohjelmaa tuli hieman muuttaa

Kalustuksen edetessä komponentit juotettiin ja mitattiin yleismittarilla jokainen juotostäplä erikseen mahdollisten tinasiltojen varalta. Kyseinen aikaavievä toimenpide onkin välttämätön jyrsittyjä piirilevyjä valmistettaessa. Lopulta piirilevyjen valmistuttua oli aika ladata jokaiseen levyyn testiohjelma, jolla varmistettiin kaikkien ominaisuuksien toiminta. Tässä vaiheessa huomattiin piirilevyn valmistustekniikan aiheuttavan ylimääräistä kapasitanssia johtimien välillä. Etenkin pelitasojen ohjaimissa olevia d-veräjiä ohjaavan aliohjelman ajoituksia piti muuttaa jotta veräjäpiirien ajoittain virheellinen toiminta saatiin korjattua.

5 OHJELMISTO

5.1 Ohjelmistokehityksen teoriaa

5.1.1 Ohjelmiston suunnittelu

Ohjelmiston suunnittelussa oli otettava huomioon muutamia seikkoja. Isäntälaitteen ja pelitasojen ohjaimien välille oli luotava luotettava tiedonsiirtoprotokolla, joka esimerkiksi käyttäjän käyttövirheen takia ei aiheuttaisi virhetilannetta ja laitteen jumiutumista. Yhtäaikaa ajettavat palvelurutiinit eivät saisi olla toisiaan poissulkevia, jolloin saattaisi aiheutua virhetilanteita pelilogiikassa. Koska peli sijoitetaan lopulta julkiseen tilaan, tulee sen kyetä toimimaan pitkiäkin ajanjaksoja ongelmitta ja mahdollisten virhetilanteiden varalta tulee laitteiden resetoinnin ja alustuksen olla helppoa ja varmaa.

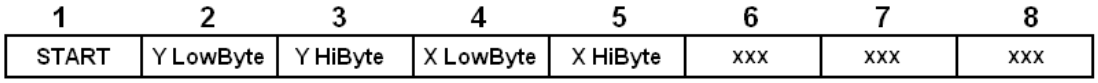
Työn tässä osassa keskitytään lähinnä pelitasojen ohjaimiin sekä niiden ja isäntälaitteen väliseen tiedonsiirtoprotokollaan. Protokollaa kehitettäessä oli päätettävä kaikista linjalla siirrettävistä komennoista ja laiteosoitteista.

Ohjelmat käännettiin ja kehitettiin pääosin Linuxissa käyttäen Kontrollerlab-nimistä kehitysympäristöä. Kontrollerlab käyttää ohjelman kääntämiseen AVR-GCC:tä, joka on ilmainen GNU-projektiin kuuluva kääntäjäkokoelma. Lisäksi asennettuna tulee olla kirjastoista AVR-Libc, AVR-Dude valmiin ohjelman lataamiseen mikro-ohjaimeen sekä AVR-Binutils. Kontrollerlab on jatkuvasti kehittyvä avoimeen lähdekoodiin perustuva graafinen kehitysympäristö, johon on integroitu tekstieditorin ja valmiin ohjelman latauksen lisäksi myös JTAG-debuggeri sekä RS232-pääte sarjaväylä-kommunikointia varten.

AVR-GCC on myös saatavana Windows-ympäristöön, nimellä WinAVR. Paketti sisältää kaikki tarvittavat sovellukset ohjelman kirjoittamiseksi, kääntämiseksi ja lataamiseksi mikro-ohjaimeen komentokehoitepohjaisesti. Paketin yhteydessä voidaan graafisena kehitysympäristönä käyttää esimerkiksi aikaisemmin mainittua AVR Studiota tai vaikkapa Context Editor-nimistä tekstieditoria, jossa on mukana myös liitäntä ulkoisille kääntäjille sekä ohjelman lataajille.

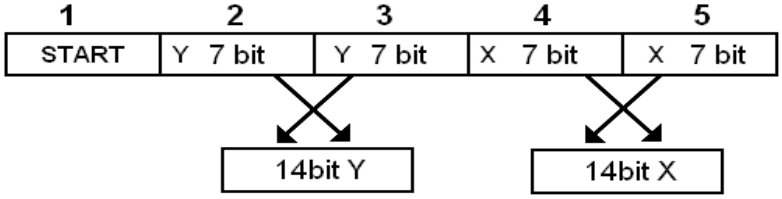
5.1.2 Ensimmäiset vaiheet ohjelmiston toteutuksessa

Siinä vaiheessa kun ensimmäiset tilatut komponentit olivat saapuneet, oli aika aloittaa ensimmäisten ohjelmapalasten kirjoittaminen. Ensimmäisenä suurennuslasin alle joutuivat kosketuskalvojen ohjaimet, joiden lähettämät datapaketit piti purkaa auki. Ratkaisuksi kirjoitettiin lyhyt ohjelma AVR-kehityslaudalle, joka tulosti näytölle jokaisen kosketuskalvolta vastaanotetun tavun desimaaliarvona. Ensimmäiseksi huomattiin ainoastaan yhden tavun olevan arvoltaan yli 127. Kosketustapahtuman yhteydessä ensimmäisen lähetetyn paketin ensimmäinen tavu oli arvoltaan 136 kasvaen yhdellä jokaisessa lähetetyssä paketissa kunnes kosketus lopetetaan. Näin voidaan huomioda ainoastaan ensimmäisen kosketuspaikan datapaketti, josta voidaan purkaa auki tarvittavat x- ja y-koordinaatit. Koordinaatit vaativat neljä tavua, joten loput kolme tavua ovat erilaista pakettien järjestysnumerointiin ja kosketusliikkeeseen liittyvää dataa. Tässä työssä ainostaan koordinaatit ja aloitustavu ovat laitteen toiminnan kannalta merkittäviä joten datapaketin viimeistä kolmea tavua ei käsitellä.



Kuva 6: Datapaketin rakenne tavuittain

Datapaketin toinen tavu sisältää y-koordinaatin seitsemän vähiten merkitsevää bittiä ja kolmas tavu puolestaan seitsemän eniten merkitsevää bittiä. Yhdistämällä nämä saadaan yhteensä 14-bittisellä resoluutiolla oleva y-koordinaatti. Vastaavasti paketin neljäs ja viides tavu sisältävät vastaavat tiedot x-koordinaatista. Näiden tietojen selvittyä voitiin alkaa kirjoittaa ohjelmamodulia kosketustapahtuman käsittelyyn.

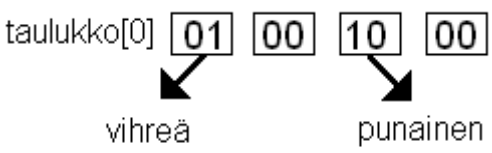


Kuva 7: Koordinaattien muodostus

5.1.3 Pelitapahtumien käsittely ohjelmassa

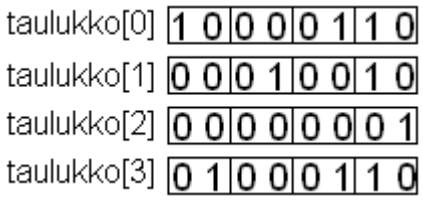
Koska kosketustapahtuman tapahtumisajankohtaa ei voida ennakoida, pitää sen käsittelyyn olla jatkuva mahdollisuus. Tämä tarkoittaa käytännössä keskeytysrutiinin käyttöä datapaketin vastaanotossa. Pelitason ohjaimen vastaanottaessa uuden merkin RS232-liitännällä, generoidaan keskeytys. Keskeytysrutiinin sisällä luetaan uusi merkki ja tarkistetaan sen arvo. Mikäli vastaanotetun tavun arvo ei ole 136, joka kuvaa kosketustapahtuman ensimmäisen datapaketin aloitustavua, poistutaan välittömästi keskeytysrutiinista. Mikäli tavun arvo on 136, jäädään toistosilmukkaan odottamaan seuraavia neljää tavua, joista lopulta lasketaan kosketustapahtuman koordinaatit. Tämän jälkeen päivitetään pelitason tiedot kosketetun ruudun osalta, ja asetetaan uutta dataa ohjelman sisällä kuvaava lippu. Lopulta poistutaan keskeytysaliohjelmasta normaaliin ohjelmansuoritukseen.

Pelitason ohjain käsittelee kokonaisuudessaan neljästä pysty- ja vaakarivistä koostuvaa pelitasoa, jonka sisältämä tieto on tallennettava jotenkin. Tässä tapauksessa tieto tallennetaan nelialkioiseen tavutaulukkoon, jolloin jokainen taulukon alkio kuvaa yhtä pelitason vaakariviä. Koska pelitasoilla ruutujen indikointi tapahtuu kaksivärisillä ledeillä, joiden värejä ohjataan erillisillä anodeilla, on luonnollista osoittaa kustakin taulukon sisältämästä tavusta yksi bitti kuhunkin anodiin. Tästä johtuen kutakin pelialueen ruutua voidaan tarkastella taulukon tavuista kahden bitin ryhmiä tutkimalla. Kussakin kahden bitin ryhmässä vihreää pelaajaa edustaa bittikuvio ”01” ja punaista ”10”. Näin isäntälaitteen pyytäessä pelitason tietoja, voidaan ne välittää I2C-väylää pitkin siirtämällä ainoastaan neljä tavua dataa.



Kuva 8: Esimerkki tavusta

Kosketustapahtuman generoiman keskeytyksen aikana koordinaattien laskennan jälkeen tarkastetaan tavutaulukosta, onko kosketettu ruutu jo valittuna. Mikäli ei ole, kirjoitetaan tavutaulukkoon oikeaa ruutua merkitsevään kahden bitin ryhmään pelivuorosta riippuen joko ”01” tai ”10”. Mikäli kosketettu ruutu on jo aikaisemmin valittuna, ei uutta dataa kuvaavaa lippua aseteta eikä kosketustapahtumaa huomioida, vaan pelaaja saa valita uudestaan kunnes tyhjä ruutu on valittu.



Kuva 9: Mahdollinen taulukon sisältö

5.1.4 Tiedonsiirto isännän ja orjalaitteiden välillä

Tiedonsiirto järjestelmän sisällä päätettiin jo aikaisessa vaiheessa hoitaa I2C- eli TWI-väylän avulla. TWI-väylä on Atmega32-mikro-ohjaimessa toteutettu laitteistopohjaisena. Itse TWI-moduuli toimii tilakonepohjaisesti ja sen siirtymiseen tilasta toiseen voidaan vaikuttaa sen ohjausrekistereitä ja niiden sisältämiä lippuja manipuloimalla. Näin toteutettuna sen käyttöönotto on ensikertalaiselle hankalahkoa, mutta menetelmä sallii moduulin erittäin joustavan käytön. TWI:n käyttämiseen master-tilassa on saatavilla joitakin esimerkkejä mutta slave-tilan toteuttaminen vaatii ATmega32:n datalehden perinpohjaista lukemista.

TWI-moduulin ohjaus koostuu viidestä eri rekisteristä, joita manipuloimalla voidaan luoda haluttu tiedonsiirtoprotokolla. TWBR, TWI Bit Rate Register sisältää kahdeksan ohjausbittiä, joilla voidaan konrolloida masterin tuottaman kellopulssin taajuutta yhdessä TWSR, TWI Status Registerin, sisältämien jakajanohjausbittien kanssa.

Maskaamalla muut TWSR:n sisältämät bitit esiin saadaan selville TWI-moduulin sen hetkinen tilakoodi. Jokaiselle mahdolliselle tiedonsiirron vaiheelle on oma koodinsa, joka on piirin datalehdessä ilmoitettu heksadesimaalina. Näin voidaan tilakoneelle ohjelmallisesti luoda haluttu käyttäytyminen eri tilojen välillä.

TWCR, TWI Control Register sisältää ohjausbitit ja liput, joilla voidaan ohjata koko tiedonsiirtotapahtumaa. Rekisteriä muokkaamalla voidaan muodostaa esimerkiksi START- tai STOP-tapahtumia. Mikäli tiedonsiirtoa ei jostain syystä haluta toteuttaa keskeytyspohjaisena, voidaan tarkastella TWI Interrupt-lipun asettumista merkkinä väylällä havaitusta tapahtumasta.

TWDR, TWI Data Register on kaksisuuntainen rekisteri, joka toimii tiedonsiirtorekisterinä. Transmitter-tilassa rekisteriin kopioidaan seuraavaksi lähetettävä tavu tai laiteosoite ja Receiver-tilassa siitä voidaan lukea viimeisin vastaanotettu data.

Laitteen ollessa orja-tilassa, sisältää TWAR, TWI Address Register seitsemänbittisen laiteosoitteen sekä GCE eli General Call Enable-lipun, jolla voidaan määrittää, vastaako kyseinen laite yleiskutsuun, joka osoitetaan laiteosoitteelle 0. Työssä kyseinen ominaisuus ei ollut tarpeen joten, se jätettiin hyödyntämättä.

TWI-väylän käyttökirjastoja kirjoitettaessa ilmeni, että väylän teoreettinen toiminta tulisi tuntea todella tarkasti. Tiedonsiirron jokaisessa vaiheessa rekisterien ohjauslippuja pitää lukea ja kirjoittaa. Lisäksi mahdolliset moodimuutokset, kuten vaihtaminen Transmitter-tilasta Receiver-tilaan tehdään kesken tiedonsiirtotapahtuman. Vastaavasti jokainen tiedonsiirtotapahtuma kuitataan kirjoittamalla TWI-keskeytyslippu 1-tilaan jokaisen keskeytysrutiinin jälkeen, jolloin seuraava tapahtuma aloitetaan.

Master-osion kirjaston kirjoittaminen oli huomattavasti helpompaa, sillä saatavilla on esimerkkejä muun muassa TWI-väyläisten eeprom-muistien käyttöä varten. Tällaisesta kirjastosta mallia ottaen oli helpompaa luoda väylän perusominaisuudet tarjoava kirjasto. Koska väylän isäntälaite ohjaa väylän tapahtumia, ei sen tarvitse tässä tapauksessa olla keskeytysohjattu.

Orjalaitteen osuus olikin tarkempaa työtä vaativa tapaus. Etukäteen ei voida ennustaa milloin väylällä on tapahtumia, joten orjalaitteen tapauksessa keskeytyspohjainen tapahtumaohjaus on käytännössä välttämätön. Keskeytys generoituu kaikissa väylätapahtumissa, joten kaikki mahdolliset tilakoodit pitää sisällyttää keskeytysrutiinin sisään. Keskeytyksen sisällä luetaan tilakoodi ja toimitaan sen mukaan: rekisterien lippuja lukemalla ja kirjoittamalla toteutetaan tilakoneelle haluttu toiminta tilojen välillä ja varmistetaan, että palataan aina lopulta perustilaan. Kun tilakoneen toiminta on toteutettu, on valmiina toiminnallinen runko, jonka sisään voidaan kirjoittaa täydellinen tiedonsiirtoprotokolla.

Tiedonsiirtoprotokollaa kehitettäessä oli ensimmäiseksi suunniteltava laitteessa tapahtuva pelikierto kokonaisuudessaan. Näin voitiin selvittää tarvittavat TWI-väylän yli siirrettävät komennot sekä tiedonsiirron suunnat. Koska TWI-väylän tapauksessa isäntälaite määrää jokaisen tiedonsiirron tapahtumisajankohdan, tulee pelitapahtumakin olla kokonaisuudessaan sen ohjaama. Työssä päätettiin tietynlaisesta pelikierrosta laitteiden välillä. Isäntälaite komentaa kutakin pelitason ohjainta ennalta määrätyllä rutiinilla.

5.1.5 Pelikierto teoreettisesti

Käynnistettäessä peli, isäntä lähettää resetointikomennon kaikille pelitason ohjainkorteille. Tällöin pelitaulukot alustetaan ja jäädään odottamaan isäntälaitteen seuraavaa komentoa. Isäntälaite kertoo jokaiselle pelitason ohjaimelle kumman pelaajan vuoro on valita jokin ruutu. Tällöin isäntälaite alkaa kyselemään vuorotellen kultakin pelitason ohjaimelta, onko uusia pelitapahtumia havaittu. Vuoronvaihtokomennon saatuaan pelitasojen ohjaimet puolestaan jäävät odottamaan kosketustapahtumaa, pelitapahtuman kyselyä tai mahdollista uutta vuoronvaihtokomentoa, mikäli pelitapahtuma on tapahtunut jollakin muista pelitasoista. Kosketustapahtuman tapauksessa kosketuksen koordinaatit luetaan, sijoitetaan koordinaattien osoittamaan tavutaulu-

kon osaan vuoron omaavaa pelaajaa kuvaava bittikuvio, ja asetetaan uutta dataa kuvaava lippu. Pelitapahtumakyselyn tullessa kukin pelitason ohjain palauttaa isäntälaitteelle uutta dataa kuvaavan lipun arvon. Lipun mahdollisia arvoja ovat epätosi, mikäli tapahtumia ei ole ollut tai tosi, mikäli pelaaja on valinnut ruudun. Isäntälaitteen vastaanottaessa lipun totuusarvoltaan tosi, pyytää se seuraavaksi kyseistä pelitason ohjainta lähettämään pelitaulukkonsa pyytäen sitä erikseen tavu kerrallaan. Pelitason ohjaimen lähetettyä kaikki tavut onnistuneesti, lähettää isäntälaite komennon lipun resetointia varten. Vasta nyt pelitason ohjain voi asettaa lipun arvon tilaan epätosi. Tällä kuittauksella varmistetaan datan perillepääsy.

Tässä välissä isäntälaite tallentaa pelitaulukon omiin muuttujiinsa ja tarkistaa onko jompikumpi pelaaja mahdollisesti voittaja, mutta sitä ei käsitellä työn tässä osassa. Otettuaan vastaan pelitaulukon, isäntälaite antaa vuoronvaihtokomennon kaikille pelitason ohjaimille, jolloin vastaavaa pelikiertoa jatketaan kunnes pelialue on kokonaisuudessaan valittu, voittaja on löydetty tai pelaaja keskeyttää ja aloittaa uuden pelin.

Pelitasojen ohjaimissa kaikki pelitapahtumiin liittyvä suoritetaan keskeytysrutiineissa, jolloin ohjain kykenee palvelemaan järjestelmää välittömästi. Keskeytysten ulkopuolella suoritetaan loppumatonta toistorakennetta, jossa päivitetään pelitaulukon sisältöä jatkuvasti pelitason indikointiledejä ohjaaviin d-kiikkuihin. Näin menetellen järjestelmän vastausaika on lyhyt eikä pelaaja huomaa pelitasojen ledeissä minkäänlaista poikkeamaa vaikka suoritettaisiin keskeytysrutiinia.

5.1.6 Virhetilanteiden varalta

Vaikka isäntälaitteen ja pelitason ohjaimien välinen tiedonsiirto on pyritty tekemään mahdollisimman toimintavarmaksi, haluttiin lisäksi säilyttää mahdollisuus resetoida pelitasojen ohjaimet laitteistotasolla isäntälaitteen ohjelmasta käsin. Tätä ominaisuutta ei laitteen testausvaiheessa tarvittu.

5.2 Toteutettu ohjelmisto esimerkein

5.2.1 Esimerkki yksinkertaisesta AVR-ohjelmasta

Ohjelmoitaessa AVR-mikro-ohjaimia C-kielellä, on ohjelman rakenne pääosin sama kuin kirjoitettaessa normaalia PC-ohjelmaa. Ohjelman alussa ilmoitetaan esikäsittelijälle #include komennolla mitä tiedostoja projektiin tulee sisällyttää. Gcc-ympäristössä pääohjelman oletetaan aina palauttavan int-tyyppisen palautusarvon. Erilaiset toiminnot toteutetaan kirjoittamalla tai lukemalla mikro-ohjaimen rekistereitä. Rekistereillä on AVR-GCC-ympäristössä symboliset nimet jotka vastaavat piirin datasheetissa nimettyjä rekistereitä. Pitää kuitenkin olla tarkkana mikro-ohjaimen mallista, sillä esimerkiksi vanhoissa jo markkinoilta poistuvissa versioissa rekisterinimet saattavat poiketa viimeisimmistä malleista. Lisäksi AVR-Libc kirjastossa on virheitä symbolisten vakioiden arvoissa, esimerkiksi joidenkin rekisterien sisältämien lippujen nimet eivät pidä paikkaansa kaikilla mikro-ohjaimilla vaan ne on korvattava numeerisilla arvoilla.

Perimmäisenä erona normaaliin PC-ohjelmointiin mikro-ohjaimet usein suorittavat jotakin jatkuvasti toistettavaa toimintoa, joka toteutetaan kirjoittamalla suoritettava ohjelmakoodi loputtoman while-toistorakenteen sisään. Myöskin muistinkäyttöä ja ohjelmakoodin kokoa tulee minimoida. Tähän yksinkertainen ohjenuora on käyttää muuttujia järkevästi ja välttää sisäkkäisten toisto- ja ohjausrakenteiden käyttöä mikäli mahdollista. Lisäksi käytettävien piirien datalehtien lukeminen vie aikaa vähintään yhtä paljon kuin itse ohjelmoiminen, mikäli ohjelma on jo valmiiksi suunniteltu paperilla. Ohessa esimerkkinä yksinkertainen ohjelma joka muuttaa B-portin pinnien tilaa jatkuvasti.

```
#include <avr/io.h> //sisällytetään otsikkotiedosto
```

```
int main (void){  // pääohjelman määrittely
DDRB=0xFF  //B-portin suunta ulos

        while(1){ // loputon silmukka
        PORTB=0xFF; //B-portin pinnien tila tosi
        PORTB=0x00; //B-portin pinnien tila epätosi
        }

Return(0); // pakollinen palautusarvo
}
```

5.2.2 Kosketustapahtuman käsittely

Ensimmäisenä voidaan käsitellä kosketustapahtuman hoitavaa osaa ohjelmasta. Tämä osa ajetaan keskeytysrutiinina ja se lukee kosketuskalvon ohjaimen RS232-liitännän yli lähettämiä tavuja ja tarkistaa vastaanotetun tavun arvon. Arvon ollessa 136, jäädään odottamaan neljän seuraavan tavun saapumista,luetaan ne nelialkioiseen taulukkoon ja lopulta muunnetaan ne x- ja y-koordinaateiksi ennen kesketyksestä poistumista. Mikäli ensimmäinen tavu ei olekaan arvoltaan 136, poistutaan keskeytysaliohjelmasta. Ohessa keskeytysaliohjelma kommentoituna.

```
ISR(USART_RXC_vect) //keskeytysaliohjelma uudelle merkille
{
        int laskuri=0; // esitellään laskuri-muuttuja
        char ReceivedByte; //esitellään muuttuja, johon uusi tavu luetaan
        ReceivedByte = UDR; // luetaan uusi tavu UDR-rekisteristä
        if (ReceivedByte==136){ //uusi datapaketti alkaa
                for (laskuri=0;laskuri<4;laskuri++){//luetaan koko paketti
                loop_until_bit_is_set(UCSRA,7);//odotetaan uutta tavua
                taulu[laskuri]=UDR; //luetaan uusi tavu taulukkoon
                }
        GetXy(); //ajetaan aliohjelma, joka muuntaa taulukon xy-koordinaateiksi
        }

}
```

Jotta kosketustapahtuma olisi täysin käsitelty, tulee x- ja y-koordinaatit selvittää ja varata pelitaulukosta oikea kahden bitin ryhmä kuvaamaan kosketettua pelitason ruutua. Bittikuvio tulee syöttää sen hetkisen pelaajan vuoron mukaan.

Kosketuskalvon ohjain lähettää koordinaatit 4096x4096 resoluutiossa eli kumpikin koordinaatti on välillä 0-4095, joten helpointa on muuntaa kumpikin koordinaatti ku-

vaamaan pelitasojen koordinaatteja välille 0-3. Näin voidaan suoraan viitata pelitaulukon alkioihin. Helpoiten muuntaminen tapahtuu jakamalla kumpikin koordinaatti 1024:llä.

Seuraavaksi tulee muunnettujen koordinaattien avulla muodostaa maski, jolla tarkistetaan pelitaulukosta, onko koordinaattien osoittama ruutu jo varattu. Mikäli maskauksen tulos osoittaa ruudun olevan tyhjä, täytetään se lisäämällä kyseiseen tavuun tietty lisäysarvo, joka muodostetaan siirtämällä pelaajan vuoroa kuvaavaa kahden bitin kuvioita x-koordinatin osoittama määrä vasemmalle. Koska kukin ruutu kuvataan kahdella bitillä, tulee tässä yhteydessä myöskin kertoa x-koordinaatti kahdella. Lopulta kun oikea tieto on lisätty pelitaulukkoon, asetetaan uutta dataa kuvaava lippu osoittamaan että uusi pelitapahtuma on käsitelty ja isäntälaite voi lukea pelitaulukon tiedot.

Mikäli valittu ruutu onkin varattu, mihinkään toimenpiteisiin ei ryhdytä ja kosketustapahtuma hylätään. Ohessa aliohjelma koordinaattien muuntamiseen ja pelitaulukon käsittelyyn.

```
void GetXy(void)
{
        unsigned char y=(taulu[1]*128+taulu[0])/1024; //muunnetaan y-
                                                      //koordinaatti
                                                      //välille 0-3
        unsigned char x=(taulu[3]*128+taulu[2])/1024; //samoin x

        unsigned char temp=(3<<(x*2));  //muodostetaan maski
        unsigned char temp2=(vuoro<<(x*2)); //muodostetaan lisäysarvo

        if (((matr[y]&temp)==0)){   //tarkistetaan, onko ruutu tyhjä
                matr[y]+=(temp2); //kirjoitetaan lisäysarvo pelitaulukkoon
                uusi2=1;  //uutta dataa saatavilla
        }
}
```

5.2.3 Asetuksien määrittely

Jotta työssä käytetyn ATmega32-mikro-ohjaimen keskeytyksiä tai RS232-kommunikointia voitaisiin käyttää, on kyseiset ominaisuudet ensin otettava käyttöön ja alustet-

tava asianmukaisesti. RS232-kommunikoinnin mahdollistamiseksi pitää USART-moduuli olla oikein alustettuna. Tässä tapauksessa pitää käynnistää moduulin vastaanottopiiri, enabloida vastaanotetun tavun generoima keskeytys, asettaa oikeat yhteysasetukset ja haluttu tiedonsiirtonopeus. Tämä tapahtuu muokkaamalla mikro-ohjaimen rekistereitä, joista on yksityiskohtainen kuvaus ja esimerkit mikro-ohjaimen valmistajan datasheetissa eli tuotelehdessä.

Vastaanottopiiriä ohjaava lippu löytyy UCSRB-rekisteristä ollen rekisterin bitti numero neljä. Rekisterin bitit numeroidaan vähiten merkitsevästä bitistä alkaen välillä 0-7. Keskeytyksen mahdollistava lippu löytyy samasta rekisteristä numerolla 7. Asettamalla nämä bitit 1-tilaan, on ominaisuudet käytössä. Keskeytykset pitää lisäksi erikseen sallia globaalisti asettamalla SREG-rekisterin eniten merkitsevä bitti 1-tilaan.

Tiedonsiirron asetukset löytyvät rekisteristä UCSRC. Valitaan siirrettävän merkin kooksi yksi tavu eli kahdeksan bittiä asettamalla rekisterin bitit numero kaksi ja kolme 1-tilaan.

Viimeiseksi asetetaan tiedonsiirtonopeutta kontrolloivan UBRR-rekisterin molempien tavujen arvo oikeaksi. Rekisterin oikea arvo selviää lukemalla mikro-ohjaimen valmistajan laatimaa datalehteä, jossa on taulukko rekisterin arvoista halutulla tiedonsiirtonopeudella ja käytetyllä kellokiteellä. Työssä käytettiin kellokiteenä 14.7456 Mhz:n kidettä, joka on optimoitu erityisesti sarjaliitäntäkäyttöä ajatellen ollen samalla nopein useimpia tiedonsiirtonopeuksia tukeva kide, jota käytetyissä mikro-ohjaimissa voi luotettavasti käyttää. Ohessa osa ohjelmaa, jossa tehdään edellä mainitut asetukset.

```
UCSRB = 0x90;//käynnistetään UART-vastaanotin ja keskeytysvalmius
UCSRC =0x06; //asetetaan 8-bittinen tiedonsiirto

UBRRH = 0; // nopeusrekisterin ylempi tavu,
UBRRL = 95; //ja alempi, 9600 bps  14.7456MHz kiteellä

SREG=0x80; //sallitaan globaalit keskeytykset
```

5.2.4 Isäntälaitteen lähettämät käskyt

Jotta viiden erillisen mikro-ohjaimen järjestelmässä pysyisi järjestys pelitapahtuman aikana, tulee isäntälaitteen ohjata kaikkia pelilogiikkaan liittyviä toimintoja. Isäntälaite käskyttää TWI-väylän kautta kaikkia pelitasojen ohjaimia ennalta määrätyillä yhden tavun mittaisilla käskyillä. Käskyn vastaanotettuaan pelitason ohjain siirtyy suorittamaan toimiaan viimeisen saamansa käskyn mukaan. Mikäli vastaanotettu käsky edellyttää datan lähettämistä pelitason ohjaimelta isäntälaitteelle, tapahtuu lähetys heti isäntälaitteen alustettua väylälle lukutapahtuman. Datasiirron jälkeen isäntälaite antaa uuden käskyn, jolloin pelitason ohjain vaihtaa jälleen moodia.

Komento	Toimenpiteet pelitasolla
0x01	Vaihdetaan pelivuoro välittömästi punaiselle pelaajalle
0x02	Vaihdetaan pelivuoro välittömästi vihreälle pelaajalle
0x03	Pelitaulukko alustetaan ja pelivuoro asetetaan nollaksi
0x7F	Lukutapahtuman yhteydessä palautetaan Uutta dataa-lipun arvo isäntälaitteelle
0x80	Lukutapahtuman yhteydessä palautetaan pelitaulukon 1. rivi
0x81	Lukutapahtuman yhteydessä palautetaan pelitaulukon 2. rivi
0x82	Lukutapahtuman yhteydessä palautetaan pelitaulukon 3. rivi
0x83	Lukutapahtuman yhteydessä palautetaan pelitaulukon 4. rivi
0x84	Alustetaan Uutta dataa-lipun arvo epätodeksi

Kuva 10: Taulukko isäntälaitteen antamista käskyistä

5.2.5 TWI-väylän toteutus

Twi-väylää varten kirjoitettiin omat kirjastot sekä isäntälaitteelle että orjalaitteena toimiville pelitason ohjaimille. Isäntälaite muodostaa väylän kellosignaalin ja ohjaa väylän tapahtumia. Isäntälaitteen TWI-moduuli tulee alustaa Master Transmitter-tilaan sekä kellosignaalin muodustumiseen vaikuttavat rekisterit asettaa siten että halutut ominaisudet saavutetaan. Ohessa ote kellosignaalin asettamisesta.

```
#ifndef F_CPU
#define F_CPU 14745600UL //kiteen taajuus
#endif
```

```
#define SCL_CLOCK  100000L //haluttu väylätaajuus

extern void i2c_init(void){ //TWI-moduulin kellotaajuuden alustus
  TWSR = 0x00;         //ei jakajaa kellotaajuudelle
  TWBR = ((F_CPU/SCL_CLOCK)-16)/2; //pitää olla suurempi kuin 10
}                                                  //jottei toiminta vaarannu
```

Kun kellotaajuus on asetettu, voidaan aloittaa tiedonsiirto väylällä asettamalla isäntälaite Master Transmitter-tilaan ja lähettämällä Start-komento täydennettynä kirjoitusbitillä. Lähettämällä kirjoitusbitin sijasta lukubitti, siirtyy isäntälaite automaattisesti Master Receiver-tilaan. Ohessa esimerkki Start-komennon lähettävästä ja isäntälaitteen tarvittavaan tilaan alustavasta aliohjelmasta. Aliohjelma palauttaa palautusarvon tosi, mikäli tiedonsiirrossa on tapahtunut virhe.

```
extern unsigned char i2c_start(unsigned char address, unsigned char dir){

        uint8_t   twst;

        // lähetetään START-komento
        TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

        // odotetaan kunnes START lähetetty
        while(!(TWCR & (1<<TWINT)));

        // tarkastetaan TWI Status Registerin arvo.
        twst = TW_STATUS & 0xF8;
        if ( (twst != TW_START) && (twst != TW_REP_START)) return 1;
        //virhe havaittu

        TWDR = address*2 + dir;  // lähetetään osoite
        TWCR = (1<<TWINT) | (1<<TWEN);

        // Odotetaan kunnes lähetys tehty sekä ACK/NACK vastaanotettu
        while(!(TWCR & (1<<TWINT)));

        // tarkastetaan TWI Status Registerin arvo
        twst = TW_STATUS & 0xF8;
        if ( (twst != TW_MT_SLA_ACK) && (twst != TW_MR_SLA_ACK) )
        return 1; //virhe havaittu

        return 0;
}
```

Kun alustus Master Transmitter-tilaan on tehty, pitää väylälle kirjoittaa siirrettävä data. Ohessa ote aliohjelmasta, joka siirtää kirjoitettavan datan ja sen palautusarvo on tosi, mikäli havaitaan virheitä, muutoin epätosi.

```
extern unsigned char i2c_write(unsigned char data){
uint8_t   twst;
```

```
        // lähetetään dataa aiemmin annettuun osoitteeseen
        TWDR = data;

    TWCR = (1<<TWINT) | (1<<TWEN); //käynnistetään lähetys

        // odotetaan kunnes data lähetetty
        while(!(TWCR & (1<<TWINT)));

        // tarkastetaan TWI Status Registerin arvo
        twst = TW_STATUS & 0xF8;
        if (twst != TW_MT_DATA_ACK) return 1; //virhe havaittu
        return 0;

}
```

Kun halutut tiedonsiirrot on tehty, väylä pitää vapauttaa STOP-komennolla jonka jälkeen voidaan aloittaa uusi tiedonsiirtotapahtuma. STOP-komennon voi lähettää ainoastaan isäntälaite. Ohessa aliohjelma, joka lähettää STOP-komennon ja siten lopettaa tiedonsiirtotapahtuman ja vapauttaa väylän.

```
extern void i2c_stop(void){ //lähetetään STOP-komento

        TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
        while(TWCR & (1<<TWSTO)); // odotetaan kunnes väylä vapautuu

}
```

Pelitason ohjaimissa TWI-väylän käsittely hoidettiin eri tavalla. Koska pelitason ohjaimien tärkeimpänä tehtävänä on pelitapahtumien seuranta mahdollisimman viiveettömästi, käytännössä ainoaksi vaihtoehdoksi TWI-väylän käsittelyyn jäi keskeytyspohjainen järjestelmä. Lisäksi rajapintaa ohjelmoidessa tuli ottaa huomioon pelitason ohjaimien erilaiset toimintamoodit, joita isäntälaite ohjaa sekä vaihdot toimintamoodien välillä. Lopulliseksi ratkaisuksi jäi sijoittaa kaikki TWI-väylän toimintaan liittyvä toiminnallisuus keskeytysaliohjelman sisään toimintamoodeineen kaikkineen. Ainoastaan TWI-moduulin alustus Slave Receiver tilaan ja laiteosoitteen asetus hoidetaan keskeytysvektorin ulkopuolella omalla aliohjelmallaan.

Johtuen Atmelin tavasta toteuttaa TWI-moduuli, kaikki väylän tapahtumat generoivat keskeytyksen mikäli kekseytykset ovat enabloituina. Keskeytysaliohjelman sisällä luetaan TWI-moduulin tilakoodi ja päätetään jatkotoimista sen perusteella. Näin voidaan toteuttaa pelitason ohjaimen tilanvaihdot joustavasti ja suorittaa tarvittaessa

vuoronvaihto lähes reaaliajassa.

6 JOHTOPÄÄTÖKSET, TULOKSET, HAVAITUT ONGELMAT

Kehitysprosessin edetessä suunnitteluvaiheesta toteutusvaiheen ja testauksen kautta kohti loppuaan, voitiin tehdä perusteellinen yhteenveto projektin aikana tehdyistä havainnoista ja saavutuksista. Periaatetasolla ajatellen, toteutettu projekti oli yksinkertainen ja sen tavoitteet oli helppo muodostaa. Tehtäessä päätöksiä laitteen rakenteesta, oli päivänselvää, että työtä tulisi olemaan melkoisesti, ennenkuin ensimmäistäkään peliä pelattaisiin. Yksityiskohtaisempi suunnittelu osoitti, ettei yhdellä tai edes kahdella mikro-ohjaimella voitaisi toteuttaa halutunlaista kokonaisuutta.

Työssä haluttiin hyödyntää nykyaikaisen tekniikan tarjoamia mahdollisuuksia muun muassa luonnollisen käyttöliittymän ja helpon käytettävyyden muodossa. Vaatimuksien johdosta päädyttiin käyttämään resistiivisiä kosketuskalvoja pelitasoilla, jolloin voitiin yhdistää uuttaa tekniikkaa edustavia osoitinlaitteita perinteiseen mikro-ohjaintekniikkaan.

Kun työssä käytettävästä teknologiasta päästiin yksimielisyyteen, oli seuraava askel luonnollisesti laitteiston ja komponenttien hankkiminen. Lopulta kun kaikki laitteistot saatiin kasattua, ohjelmat kirjoitettua ja mikro-ohjaimet ohjelmoitua, voitiin aloittaa varsinainen testaaminen.

Testaaminen aloitettiin testaamalla ensin yksittäisiä ohjelmamoduuleja ja siirtyen lopulta koko järjestelmän testaukseen. Huomionarvoista testauksen alkuvaiheessa oli havaittujen ongelmien vähäisyys, sillä ainoina epäkohtina huomattiin muutamia tarkkaavaisuusvirheitä kirjoitetussa ohjelmassa ja ne korjaamalla päästiin pelaamaan ensimmäiset onnistuneet pelit.

Varsinaisiin ongelmiin törmättiin vasta myöhemmissä testausvaiheissa. Jostakin syystä testaamiseen käytetyssä työpöydässä esiintyi voimakkaita häiriöitä, mahdollisesti johtuen mikro-ohjaimien ohjelmointiin käytetyn tietokoneen ja laboratoriovirtalähteen muodostamasta maadoitussilmukasta. Näiden häiriöiden jälkeen laitteen toiminnassa alkoi esiintyä tavallisuudesta poikkeavaa käyttäytymistä ja kellokiteitä lakkasi toimimasta. Samaten häiriötä alkoi siirtymään sähköverkosta mikro-ohjaimien käyttöjännitteeseen.

Testausvaiheessa aiheutuneet vauriot olivat korjattavissa vaihtamalla mikro-ohjaimet sekä mahdollisesti vaurioituneet jänniteregulaattorit uusiin, eikä mitään peruuttamatonta vahinkoa siten syntynyt.

Yhteenvetona tehdystä työstä voidaan todeta sen lopulta saavuttaneen sille asetetut vaatimukset ja tavoitteet. Kehitysprosessia ajatellen sen monipuolisuus ja vaativuus olivat ominaisuuksia, jotka tekivät työstä mielenkiintoisen. Huolimatta työn paljoudesta niin elektroniikkasuunnittelun kuin ohjelmoinninkin parissa, ei yhtenäkään hetkenä motivaation puute aiheuttanut ongelmia. Laitteistoa yhdistettäessä oleelliseksi tekijäksi muodostui yksityiskohtainen tutustuminen kuhunkin käytettyyn tekniikkaan periaatetasoa syvemmälti. Työhön kuului paitsi teoreettista opiskelua ja ohjelmointia, myös kymmeniä tunteja fyysistä työskentelyä ja työstökoneiden käyttöä.

Teknisinä havaintoina voidaan todeta eri teknologiasukupolvien olevan keskenään liitäntäkelpoisia, mikäli teknologioiden väliset rajapinnat saadaan toteutettua ja laitteiden ominaisuuksia voidaan hyödyntää tehokkaasti.

Kaupallisesti ajatellen laitteistokustannukset kehitetyllä laitteella ovat liian suuret kuluttajamarkkinoita ajatellen. Mahdollisesti suuria eriä valmistettaessa voidaan räätälöidä komponentit sovelluksen mukaan, ja siten säästää valmistuskustannuksissa, muutoin laitteen kaupallista valmistamista ei voida pitää järkevänä.

LÄHTEET

1. Wikipedia, Atmel AVR [verkkodokumentti]. Saatavissa:
http://en.wikipedia.org/wiki/Atmel_AVR

2. Wikipedia, RS-232 [verkkodokumentti]. Saatavissa:
http://en.wikipedia.org/wiki/RS-232

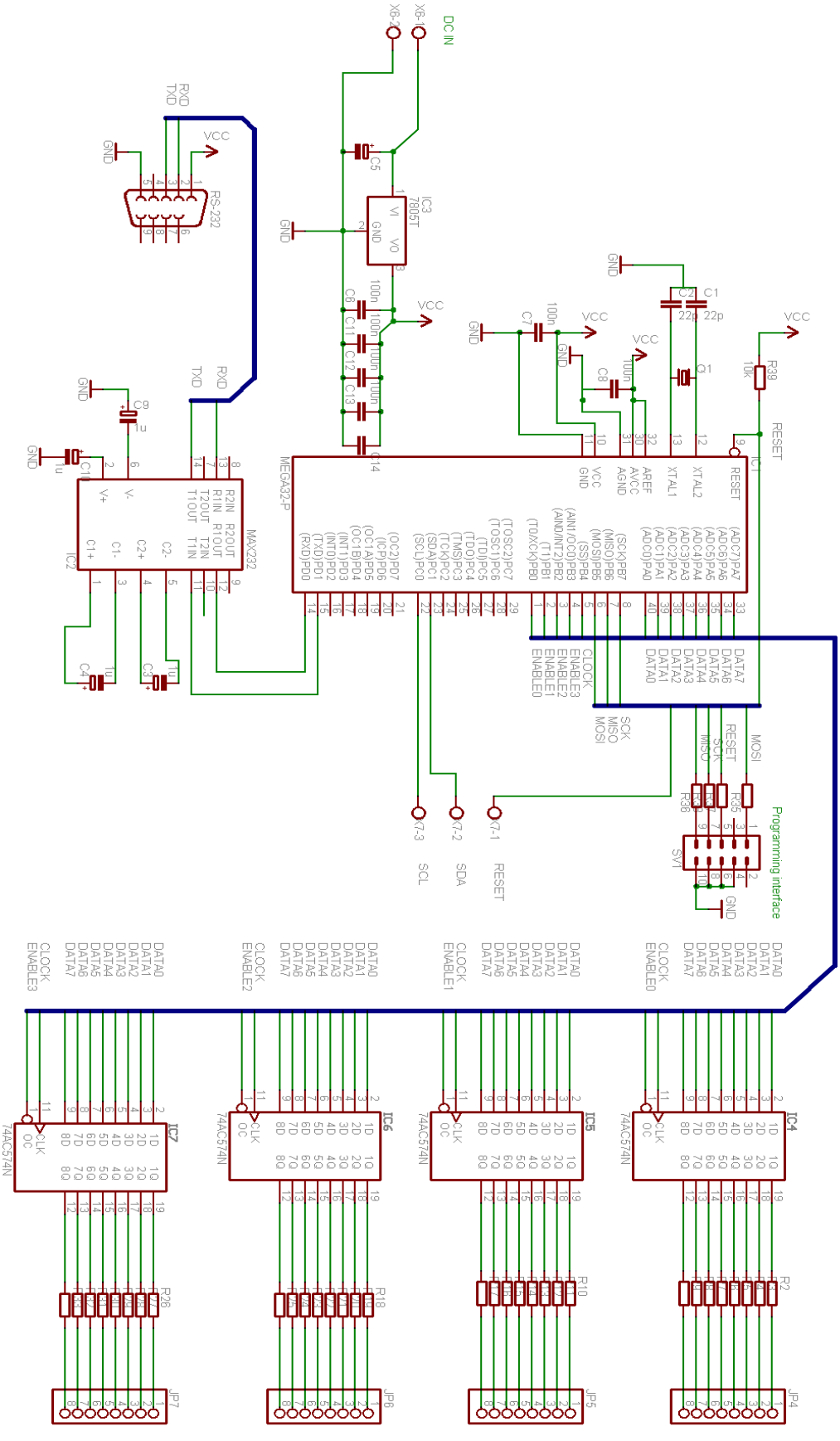
3. Wikipedia, I²C [verkkodokumentti]. Saatavissa:
http://en.wikipedia.org/wiki/I2c

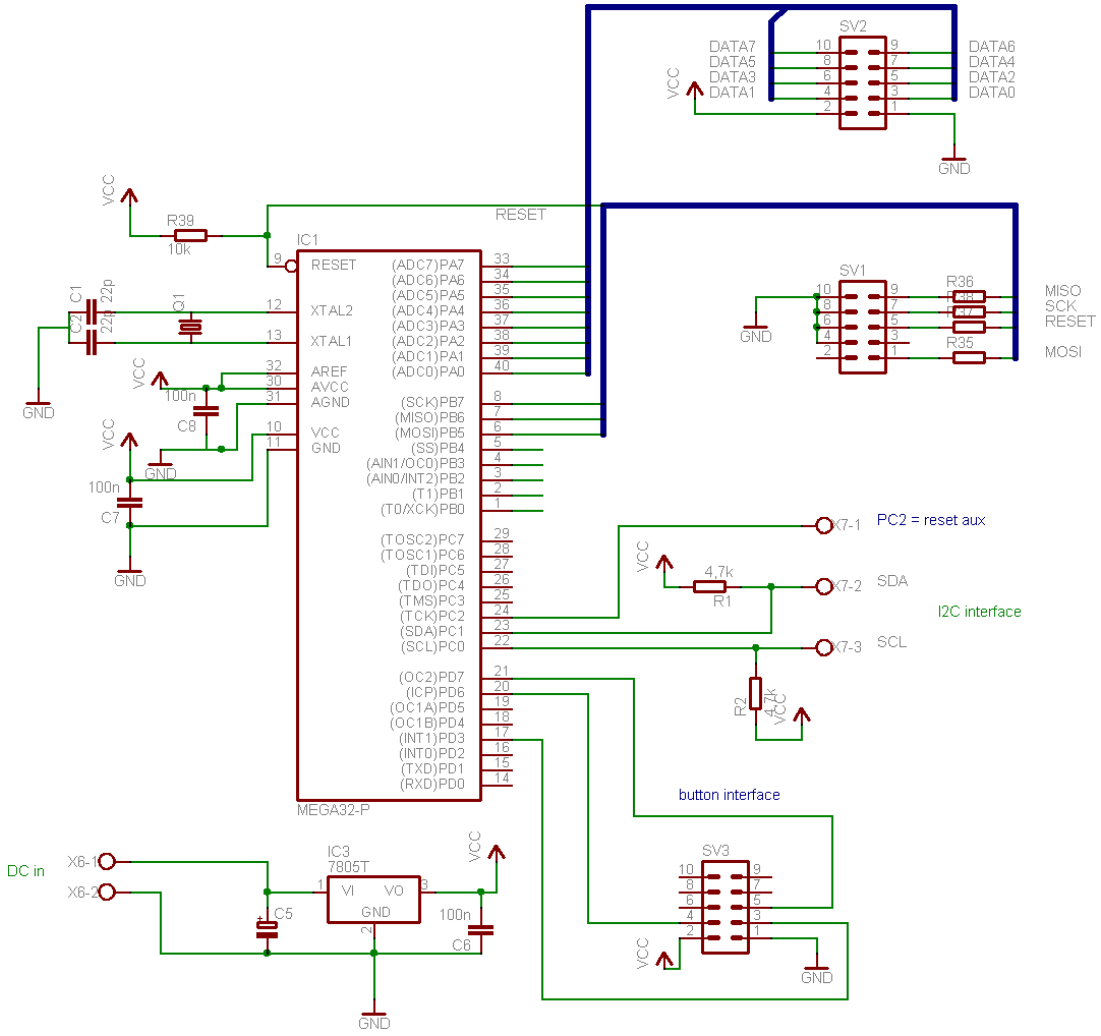
4. Wikipedia, Touchscreen [verkkodokumentti]. Saatavissa:
http://en.wikipedia.org/wiki/Touchscreen

5. Atmel Corporation, Atmega32 datasheet [verkkodokumentti]. Saatavissa:
http://www.atmel.com/dyn/resources/prod_documents/doc2503.pdf

6. Fujitsu Ltd, Touch panel datasheet [verkkodokumentti]. Saatavissa:
http://www.fujitsu.com/downloads/MICRO/fcai/touchpanels/4_wire_standard_ds_rohs.pdf

LIITTEET


DC IN
RS-232
IC3
7805T
MEGA32-P
MAX232
IC2
IC1
RESET
Programming interface
SCK
MISO
MOSI
SCL
SDA
RESET
CLOCK
ENABLE0
ENABLE1
ENABLE2
ENABLE3
74AC574N
IC4
IC5
IC6
IC7
JP4
JP5
JP6
JP7
R2
R10
R18
R26
R36
R35
SV1
GND
VCC


SV2
DATA7
DATA5
DATA3
DATA1
DATA6
DATA4
DATA2
DATA0
VCC
GND
RESET
R39
10k
IC1
RESET
XTAL2
XTAL1
AREF
AVCC
AGND
VCC
GND
(ADC7)PA7
(ADC6)PA6
(ADC5)PA5
(ADC4)PA4
(ADC3)PA3
(ADC2)PA2
(ADC1)PA1
(ADC0)PA0
(SCK)PB7
(MISO)PB6
(MOSI)PB5
(SS)PB4
(AIN1/OC0)PB3
(AIN0/INT2)PB2
(T1)PB1
(T0/XCK)PB0
(TOSC2)PC7
(TOSC1)PC6
(TDI)PC5
(TDO)PC4
(TMS)PC3
(TCK)PC2
(SDA)PC1
(SCL)PC0
(OC2)PD7
(ICP)PD6
(OC1A)PD5
(OC1B)PD4
(INT1)PD3
(INT0)PD2
(TXD)PD1
(RXD)PD0
MEGA32-P
C1
22p
C2
22p
Q1
100n
C8
100n
C7
SV1
R36
R35
MISO
SCK
RESET
MOSI
X7-1
PC2 = reset aux
4,7k
R1
X7-2
SDA
I2C interface
X7-3
SCL
R2
4,7k
button interface
SV3
DC in
X6-1
X6-2
IC3
7805T
VI
VO
GND
C5
100n
C6

tictac_slave.c:

```
#include <avr/io.h>
#include "twi_slave.h"
#include "lcd_pva.h"
#include "viive.h"
#include <avr/interrupt.h>
#include "d_kiikku.h"

unsigned char data, data2,mode,vuoro,uusi2; //muuttujien määrittely
int uusi=0;

char taulu[5],matr[4];  //koordinaattitaulukon ja pelitaulukon esittely

void Reset(void);
void GetXy(void);

#define SLAVE_ADDRESS 0x02  //laiteosoite

#define RESET 0x03  //käskyjen määrittelyt
#define RED 0x01
#define GREEN 0x02
#define UUSI 127
#define RESET_UUSI 132

#define READ0 128
#define READ1 129
#define READ2 130
#define READ3 131


ISR(TWI_vect){  //keskeytysaliohjelma TWI-väylän käyttöä varten

/*************************************************************
**** Nämä rivit  slave receiver-moodia varten                    ***
*************************************************************/

PORTC=TWSR;
if ((TWSR&0xF8) == 0x60){
      TWCR=TWCR|TWINT;
      }
      if ((TWSR&0xF8) == 0x68){
      //Arbitraatio kadotettu
      }
      if ((TWSR&0xF8) == 0x70){
      //General Call havaittu
      }

      if ((TWSR&0xF8) == 0x78){
      //Arbitrtaatio kadotettu, General Call havaittu
      }
      if ((TWSR&0xF8) == 0x80){
      //oma osoite havaittu + kirjoitus, Data vastaanotettu, ACK palautettu
      mode=TWDR;   //vaihdetaan moodi juuri vastaanotettuun


      TWCR=TWCR|TWINT; //Reset-komento saatu
       if (mode==RESET){
```

```
    Reset();
    }
    if (mode==RED){ //vuoro punaiselle
    vuoro=RED;
    }
    if (mode==GREEN){ //vuoro vihreälle
    vuoro=GREEN;
    }
    if (mode==RESET_UUSI){ //resetoidaan uutta dataa-lippu
      uusi2=0;
    }
    }

    if ((TWSR&0xF8) == 0x88){
    //oma osoite havaittu + kirjoitus, Data vastaaotettu, NACK palautettu
    mode=TWDR;



    TWCR=TWCR|(TWINT+TWEA);
    }
    if ((TWSR&0xF8) == 0x90){
    //general call havaittu, Data vastaanotettu, ACK palautettu
    TWCR=TWCR|TWINT;
    }
    if ((TWSR&0xF8) == 0x98){
    //oma osoite havaittu + kirjoitus, Data vastaaotettu, NACK palautettu
    }
    if ((TWSR&0xF8) == 0xA0){
    // Stop-komento havaittu
    TWCR=TWCR|(TWINT + TWEA);
}
/*****************************************************************
*** Nämä rivit slave transmitter-moodia varten                  ********
*****************************************************************/

if ((TWSR&0xF8) == 0xA8){
//oma osoite havaittu + luku, ACK palautettu
TWDR=data2; //lähetettävä data
if (mode==UUSI){
TWDR=uusi2;   //palautetaan uutta dataa kuvaavan lipun arvo
}
if (mode==READ0){
TWDR=matr[0];  //palautetaan pelitaulukon 1. rivi
}
if (mode==READ1){
TWDR=matr[1];  //2. rivi
}
if (mode==READ2){
TWDR=matr[2];  //3. rivi
}
if (mode==READ3){
TWDR=matr[3];  //4. rivi

}



TWCR=TWCR|TWINT; // NACK Tulisi olla vastaanotettuna
```

```
}
if ((TWSR&0xF8) == 0xB0){
//Arbitraatio kadotettu, oma osite havaittu  +  luku , ACK palautettu
}
if ((TWSR&0xF8) == 0xB8){
//Data lähetetty0, ACK otettu vastaan
}
if ((TWSR&0xF8) == 0xC0){
//Data lähetetty, NACK otettu  vastaan
  TWCR=TWCR|(TWINT + TWEA);//oma osoite tunnistetaan

}
if ((TWSR&0xF8) == 0xC8){
//Viimeinen tavu lähetetty, ACK vastaanotettu

}

}



ISR(USART_RXC_vect) //keskeytysaliohjelma UART:lle
{
   int laskuri=0;
   char ReceivedByte;
   ReceivedByte = UDR; // luetaan vastaanotettu merkki

 //start parsing data packets
if (ReceivedByte==136){ //datapaketti alkaa arvolla 136

  // TWCR &= ~(1<<TWIE);
   for (laskuri=0;laskuri<4;laskuri++){ //luetaan arvot taulukkoon
     loop_until_bit_is_set(UCSRA,7);//odotetaan kunnes uusi merkki vastaanotettu
     taulu[laskuri]=UDR; //luetaan uusi merkki
   }
 //  TWCR |=(1<<TWIE);
   GetXy(); //muunnetaan koordinaatit

}

}


int main(void){ //pääohjelma
 uusi2=0;
 UCSRB = 0x90 ;   // Käynnistetään UART:n vastaanottopiiri
 UCSRC =0x06; // Käytetään 8-bittisiä merkkejä

 UBRRH = 0;  //asetetaan UART-nopeus
 UBRRL = 95; //9600 bps  14.7456MHz kiteellä

 SREG=0x80;  //sallitaan globaalit keskeytykset




 unsigned char n;
  i2c_init_slave(SLAVE_ADDRESS); //alustetaan TWI-slave
```

```
while(1){ //ohjelmasilmukka

  set_data(matr[0],0x00); //käytetään d-kiikkuja visualisoimaan pelitaso
  set_data(matr[1],0x01);
  set_data(matr[2],0x02);
  set_data(matr[3],0x03);

  }
}



void Reset(void){ //Reset-aliohjelman toteutus
data=0;
vuoro=0;
matr[0]=0x00;
matr[1]=0x00;
matr[2]=0x00;
matr[3]=0x00;

}

void GetXy(void){ //koordinaattien muunnosohjelma
unsigned char x=(taulu[1]*128+taulu[0])/1024; //x-koordinaatti välillä 0-4095
unsigned char y=(taulu[3]*128+taulu[2])/1024; //y-koordinaatti

unsigned char temp=(3<<(x*2)); //maskausmuuttuja
unsigned char temp2=(vuoro<<(x*2)); //lisäysmuuttuja
if (((matr[y]&temp)==0)){   //tarkastetaan onko valittu ruutu tyhjä

matr[y]+=(temp2); //syötetään data
uusi2=1;  //asetetaan uutta dataa-lippu
}
}
```

twi_master.h:

```
/*********************************
***             TWI-master         **********
***     Janne Äijälä 2007         **********
*********************************

#include <avr/io.h>
#include <util/twi.h>


#ifndef F_CPU
#define F_CPU 14745600UL //käytettävä kellotaajuus
#endif

/* I2C clock in Hz */
#define SCL_CLOCK  100000L  //haluttu kellotaajuus TWI-väylälle



#define I2C_READ    1
#define I2C_WRITE   0

extern void i2c_init(void){ //väylän alustus
  TWSR = 0x00;                  // ei jakajaa
  TWBR = ((F_CPU/SCL_CLOCK)-16)/2;  //Bit Rate-rekisterin asetus
}


extern void i2c_stop(void){ //STOP-komennon lähetys

            TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);

            // odotetaan kunnes STOP suoritettu
            while(TWCR & (1<<TWSTO));

}

extern unsigned char i2c_start(unsigned char address, unsigned char dir){
      uint8_t   twst;

            // START-komennon lähetys
            TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

            // odotetaan kunnes START lähetetty
            while(!(TWCR & (1<<TWINT)));

            // Tarkastetaan TWI status rekisterin arvo ja maskataan pois prescaler-bitit.
            twst = TW_STATUS & 0xF8;
            if ( (twst != TW_START) && (twst != TW_REP_START)) return 1;

            // lähetetään laiteosoite
            TWDR = address*2 + dir;
            TWCR = (1<<TWINT) | (1<<TWEN);

            // odotetaan kunnes lähetys valmis ja kuittaus vastaanotettu
            while(!(TWCR & (1<<TWINT)));

            // Tarkastetaan TWI status rekisterin arvo ja maskataan pois prescaler-bitit.
```

```
        twst = TW_STATUS & 0xF8;
        if ( (twst != TW_MT_SLA_ACK) && (twst != TW_MR_SLA_ACK) ) return 1;

        return 0;
}

extern unsigned char i2c_rep_start(unsigned char address, unsigned char dir){
  return i2c_start( address,dir );
}


extern void i2c_start_wait(unsigned char address, unsigned char dir){
uint8_t   twst;


   while ( 1 )
   {
            // Lähetetään START-komento
            TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

        // Odotetaan kunnes lähetys valmis
        while(!(TWCR & (1<<TWINT)));

        // Tarkastetaan TWI status rekisterin arvo ja maskataan pois prescaler-bitit.
        twst = TW_STATUS & 0xF8;
        if ( (twst != TW_START) && (twst != TW_REP_START)) continue;

        // Lähetetään laiteosoite
        TWDR = address*2 + dir;
        TWCR = (1<<TWINT) | (1<<TWEN);

        // Odotetaan kunnes lähetetty
        while(!(TWCR & (1<<TWINT)));

        // Tarkastetaan TWI status rekisterin arvo ja maskataan pois prescaler-bitit.
        twst = TW_STATUS & 0xF8;
        if ( (twst == TW_MT_SLA_NACK )||(twst ==TW_MR_DATA_NACK) )
        {
           //laite ei valmis
              TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);

              // Odotetaan kunnes STOP-komento suoritettu ja väylä vapautettu
              while(TWCR & (1<<TWSTO));

           continue;
        }
        //if( twst != TW_MT_SLA_ACK) return 1;
        break;
    }
}

extern unsigned char i2c_write(unsigned char data){
uint8_t   twst;

        // lähetetään data aiemmin annettuun osoitteeseen
        TWDR = data;

      TWCR = (1<<TWINT) | (1<<TWEN);

        // Odota kunnes lähetys valmis
```

```
        while(!(TWCR & (1<<TWINT)));

        // Tarkastetaan TWI status rekisterin arvo ja maskataan pois prescaler-bitit.
        twst = TW_STATUS & 0xF8;
        if( twst != TW_MT_DATA_ACK) return 1;
        return 0;

}


extern unsigned char i2c_readAck(void){
TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
        while(!(TWCR & (1<<TWINT)));

   return TWDR;
}


extern unsigned char i2c_readNak(void){
        TWCR = (1<<TWINT) | (1<<TWEN);
        while(!(TWCR & (1<<TWINT)));

   return TWDR;
}
```