TURUN AMMATTIKORKEAKOULU
ÅBO YRKESHÖGSKOLA

Opinnäytetyö

Äänisignaalin ohjausjärjestelmä

Sampsa Salo

Tietotekniikka

2009

TURUN AMMATTIKORKEAKOULU

OPINNÄYTETYÖN TIIVISTELMÄ

Koulutusohjelma: Tietotekniikan koulutusohjelma	
Tekijä: Sampsa Salo	
Työn nimi: Äänisignaalin ohjausjärjestelmä	
Suuntautumisvaihtoehto: Sulautetut järjestelmät	Ohjaaja: Yliop. Jari-Pekka Paalassalo
Aika: Syksy 2009	Sivumäärä: 29

Tässä opinnäytetyössä suunniteltiin ja toteutettiin sähkökitaristeille yms. muusikoille suunnattu efektien valintalaite. Laitteen tarkoitus on yksinkertaistaa efektilaitteiden käyttöä sekä mahdollistaa useamman laitteen samanaikainen käyttö konserttitilanteessa.

Laitteessa on muokattavia muistipaikkoja, joihin voi tallentaa haluamansa efektiyhdistelmät. Muistipaikkoja vaihdetaan yhdellä painalluksella, kun normaalisti jokaista efektilaitetta täytyisi ohjata manuaalisesti. Äänisignaalia ohjataan releiden avulla efektilaitteiden ollessa aina päällä.

Työssä suunniteltiin kaksi erillistä laitetta, 19 ”:n räkkiin sopiva liitäntäyksikkö sekä jaloilla käytettävä ohjausyksikkö. Liitäntäyksikkö sisältää kaikki efektilaitteiden kytkemiseen tarvittavat liittimet, releet ja niiden ohjaamiseen käytetyt komponentit. Ohjausyksikkö taas sisältää ledeistä ja painikkeista koostuvan käyttöliittymän. Molemmat laitteet sisältävät mikrokontrollerit ja ne liitetään toisiinsa RJ45-kaapelilla. Tiedonsiirtoon käytetään SPI-väylää. Kaapelissa kulkee datalinjojen lisäksi ohjausyksikön käyttöjännite. Laitteiden välistä tiedonsiirtoa varten tehtiin oma tiedonsiirtoprotokolla.

Ohjausyksikkö todettiin häiriöalttiiksi testauksen yhteydessä. Ongelma saatiin korjattua ohjelmiston avulla. Äänisignaalin todettiin pysyvän parempi laatuisena, kun efektilaitteet ohitetaan täysin releiden avulla eikä äänisignaalia kierrätetä turhaan efektilaitteiden läpi niiden ollessa pois käytöstä.

Hakusanat: mikrokontrolleri, SPI-väylä, äänisignaali, rele

Säilytyspaikka: Turun ammattikorkeakoulun kirjasto

TURKU UNIVERSITY OF APPLIED SCIENCES

ABSTRACT OF THESIS

Degree Programme: Degree Programme of Information Technology

Author: Sampsa Salo

Title: Audio-signal control system

Specialization line: Embedded Systems	Instructor: Jari-Pekka Paalassalo, Lic.Tech., Principal Lecturer
Date: Autumn 2009	Total number of pages: 29

The goal of this thesis was to design and implement an effect selection device for electric guitars and other similar instruments. The purpose of the device is to simplify control of the effect devices and to enable the use of multiple effect devices at once in a live situation.

The device has editable memory slots, which can be used to store different effect combinations. You can change between the memory slots with a single push of a button, while normally you would have to switch every effect device on or off manually. The audio-signal is controlled with relays, while the effect devices are always on.

Two devices were designed, a 19" rack sized interface unit and a control unit. The interface unit includes the connectors for the effect devices, relays and other components needed to control them. The control unit includes a user interface, which consists of 9 LEDs and buttons. Both of the devices contain microcontrollers and they are connected to each other with an RJ45-cable. The cable is used for data transmission and to supply the operating voltage for the control unit. A data transfer protocol was designed for communication between the devices. Data is transferred via SPI-bus.

The control unit was found to be susceptible to interference during testing. The problems were fixed with software. It was noted that the audio-signal quality stays better when the effect devices are fully bypassed with the relays, instead of passing the signal through the effect devices' own bypass-circuitry.

Keywords: microcontroller, SPI-bus, audio-signal, relay

Deposit at: Library of Turku University of Applied Sciences

Sisällys

1 Johdanto

Tämän opinnäytetyön tavoitteena oli suunnitella ja toteuttaa sähkökitaristeille yms. muusikoille suunnattu efektien valintalaite. Laitteen tarkoitus on yksinkertaistaa efektilaitteiden käyttöä sekä mahdollistaa useamman laitteen samanaikainen käyttö konserttitilanteessa.

Laitteessa on muokattavia muistipaikkoja, joihin voi tallentaa haluamansa efektiyhdistelmät. Muistipaikkoja vaihdetaan yhdellä painalluksella, kun normaalisti jokaista efektilaitetta täytyisi ohjata manuaalisesti. Äänisignaalia ohjataan releiden avulla efektilaitteiden ollessa aina päällä.

Tavoitteena oli suunnitella kaksi laitetta, 19 ”:n räkkiin sopiva liitäntäyksikkö sekä jaloilla käytettävä ohjausyksikkö. Liitäntäyksikkö sisältää kaikki efektilaitteiden kytkemiseen tarvittavat liittimet ja komponentit. Ohjausyksikkö taas sisältää ledeistä ja painikkeista koostuvan käyttöliittymän. Molemmat laitteet sisältävät mikrokontrollerit, ja ne liitetään toisiinsa RJ45-kaapelilla.

Toteutusta varten oli muutamia vaatimuksia: Laitteiden välisen kaapelin tuli olla vähintään 10 m pitkä. Käyttäjä ei saa huomata suurempaa viivettä muistipaikkaa vaihtaessaan. Muistipaikkoja tuli olla vähintään 16 ja laitteeseen täytyi pystyä liittämään 7 efektilaitetta sekä yhden ulkoisen laitteen ohjauksen (esim. vahvistimen kanavan vaihto). Laitteiden välisen kaapelin irtoaminen ei saanut vaikuttaa releiden tilaan.

2 Ympäristö

2.1 Atmel AVR-mikrokontrollerit

AVR on puolijohdevalmistaja Atmelin vuonna 1996 kehittämä mikrokontrolleriperhe. AVR oli yksi ensimmäisistä tuoteperheistä, joka sisälsi uudelleenohjelmoitavan flash-ohjelmamuistin kertaalleen ohjelmoitavan muistin sijaan. [1]

AVR-mikrokontrollereissa käytetyn perusarkkitehtuurin kehitti kaksi norjalaista opiskelijaa Alf-Egil Bogen ja Vegard Wollan. He myivät arkkitehtuurin Atmelille ja jatkoivat sen kehittämistä siellä. AVR nimen uskotaan tulleen sanoista Alf and Vegards RISC (eli Alfin ja Vegardin RISC). [1]

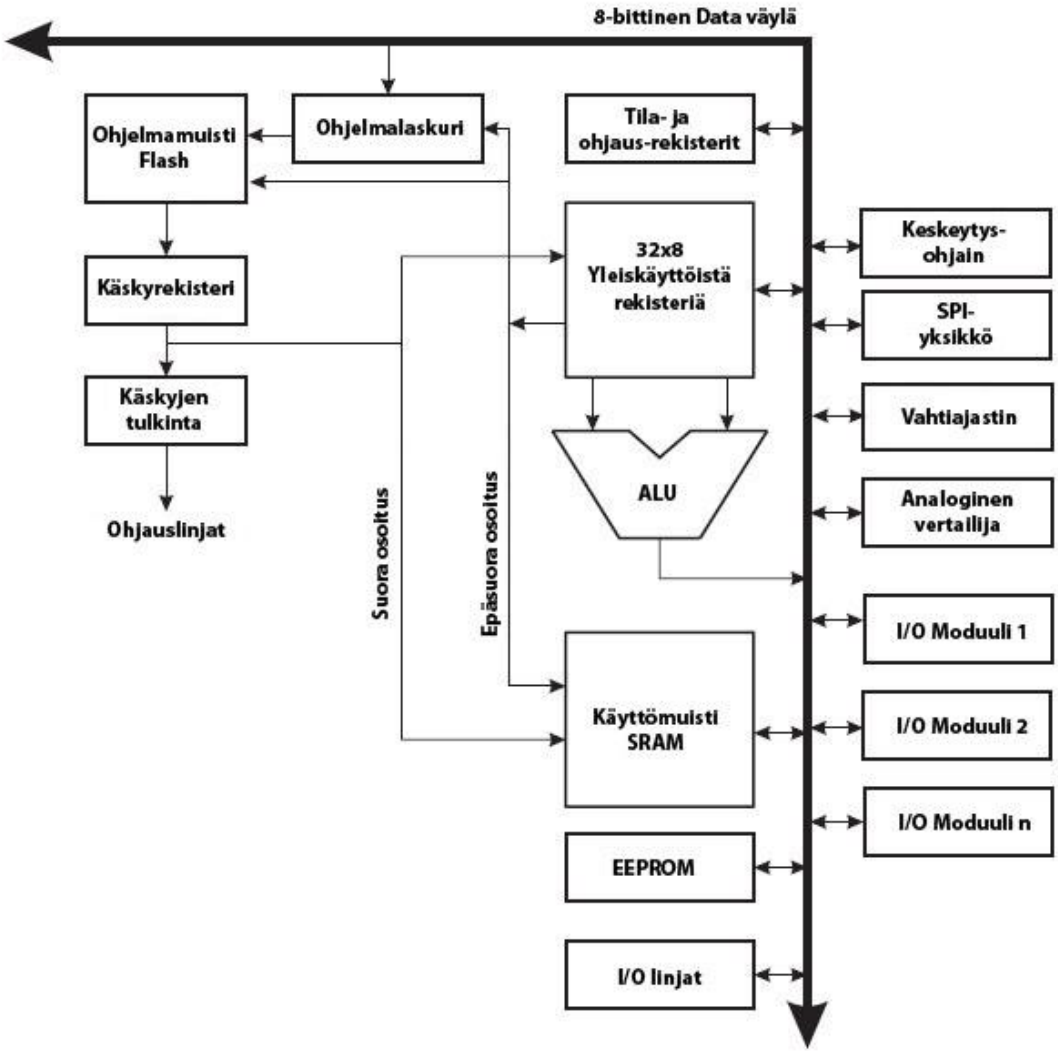
AVR-perheessä on nykyään yli 50 erilaista mikrokontrolleria. Kaikissa malleissa on sama suoritin ja muistirakenne. Suurimmat erot ovat mallien välillä muistien kapasiteetissa ja I/O-porttien määrässä. Mallit jaetaan neljään pääryhmään: tinyAVR, megaAVR, XMEGA ja Application Specific AVR. Taulukossa 1 on kolmen pääryhmän ominaisuuksien vertailu. Application Specific AVR eli sovelluskohtainen AVR-ryhmä sisältää tiettyyn tarkoitukseen suunniteltuja mikrokontrollereita, esim. USB- tai LCD-ohjaimia sisältäviä malleja. [1]

Taulukko 1. AVR-ryhmät. [1]

	tinyAVR	megaAVR	XMEGA
Ohjelmamuisti (kt)	1 - 8	4 - 256	16 - 384
Kotelon koko (pinniä)	8 - 32	28 - 100	44 - 100
Ominaisuudet	Rajoitetut oheislaitteet/liitännät	Laajennettu käskykanta, laajat oheislaitteet/liitännät	Laajennetut suorituskyvyn ominaisuudet, laajat oheislaitteet/liitännät sisältäen D/A-muuntimet

2.1.1 Arkkitehtuuri

AVR-mikrokontrollerit ovat yhdelle piirille integroituja, 8-bittisen RISC-suorittimen sisältäviä mikrokontrollereita. Suorituskyvyn maksimoimiseksi AVR käyttää Harvard-arkkitehtuuria, jossa on erilliset fyysiset väylät käskyille ja datalle. Ohjelmamuistin käskyt suoritetaan yksitasoisella liukuhihnalla. Samalla kun yhtä käskyä suoritetaan, haetaan seuraavaa jo etukäteen ohjelmamuistista. Tämä mahdollistaa käskyjen suorittamisen joka kellojaksolla. Ohjelmamuisti on uudelleenohjelmoitavaa flash-muistia. Kuvassa 1 on AVR-arkkitehtuurin lohkokaavio. [2]



Kuva 1. AVR-arkkitehtuuri. [2]

Suorituskyvyltään tehokas AVR ALU toimii suorassa yhteydessä kaikkien 32 yleiskäyttöisen työrekisterin kanssa. ALU tukee aritmeettisia ja loogisia operaatiota kahden rekisterin tai rekisterin ja vakion välillä. Myös yhden rekisterin operaatiot voidaan suorittaa. ALUn operaatiot voidaan jakaa kolmeen pääluokkaan: aritmeettisiin, loogisiin ja bittikohtaisiin operaatioihin. Tyypillisessä ALUn operaatiossa haetaan kaksi operandia rekisteritiedostosta, suoritetaan operaatio ja tallennetaan tulos rekisteritiedostoon, kaikki yhden kellojakson aikana. [2]

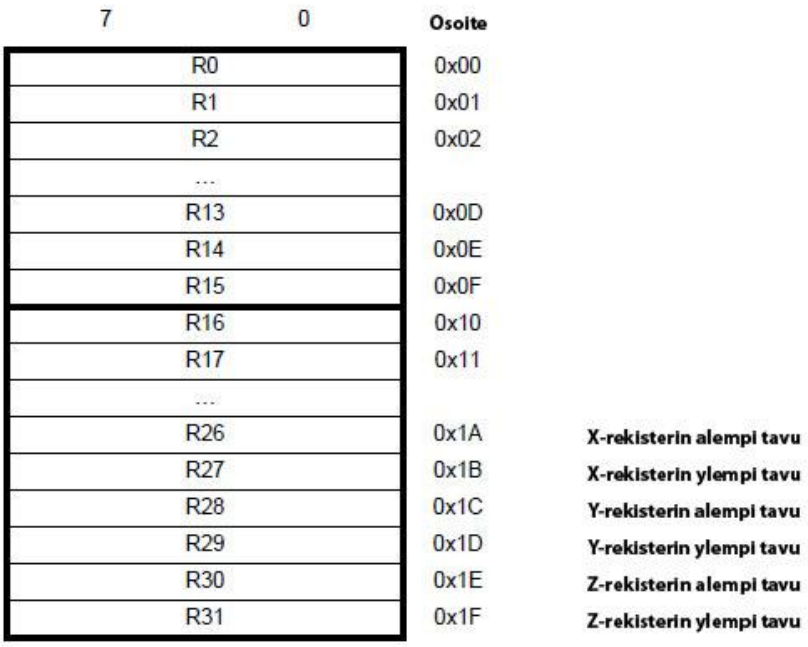
Tila-rekisteri sisältää aina tiedot viimeisimmästä aritmeettisesta operaatiosta. Näitä tietoja voidaan käyttää ohjelman suoritusprosessin muuttamiseen, joka mahdollistaa ehdollisten operaatioiden suorittamisen. Koska tila-rekisteri päivittyy automaattisesti jokaisen aritmeettisen operaation jälkeen, vältytään usein vertailukäskyjen suorittamiselta, joka taas johtaa nopeampaan ja kompaktimpaan koodiin. [2]

Nopeakäyttöinen rekisteritiedosto koostuu 32:sta 8-bittisestä yleiskäyttöisestä rekisteristä, ja se on optimoitu AVR:n parannetulle RISC-käskykannalle. Kyseisten rekisterien hakuaika on vain yhden kellojakson mittainen, joka taas mahdollistaa ALUn yksisyklisen toiminnan. Vaaditun suorituskyvyn ja joustavuuden saavuttamiseksi rekisteritiedosto tukee monia erilaisia tiedonsiirtotapoja, jotka on esitelty taulukossa 2. [2]

Taulukko 2. Rekisteritiedoston tiedonsiirtotavat. [2]

Ulos	Sisään
Yksi 8-bittinen operandi	Yksi 8-bittinen tulos
Kaksi 8-bittistä operandia	Yksi 8-bittinen tulos
Kaksi 8-bittistä operandia	Yksi 16-bittinen tulos
Yksi 16-bittinen operandi	Yksi 16-bittinen tulos

Yleiskäyttöisistä rekistereistä kuusi voidaan yhdistää kolmeksi 16 bittiseksi osoittimeksi, joita kutsutaan X-, Y- ja Z-rekistereiksi (kuva 2). Näitä käytetään epäsuorina muistialueen osoittimina, mikä mahdollistaa tehokkaan osoitteiden laskemisen. Yhtä näistä osoittimista voidaan käyttää myös flash-ohjelmamuistin hakutaulukon osoittimena. X-, Y- ja Z-rekistereillä on toimintoja eri osoitustiloissa, kuten automaattinen lisäys ja vähennys sekä kiinteä siirtymä. [2]



Kuva 2. AVR:n yleiskäyttöiset työrekisterit. [2]

2.1.2 Muistit

AVR-mikrokontrollereista löytyy kolmea erilaista muistia. Päämuistialueet ovat flash-ohjelmamuisti ja SRAM-datamuisti. Näiden lisäksi löytyy vielä pitkäaikaista EEPROM-muistia ohjelman asetusten yms. tiedon tallentamiseen. [2]

Flash-ohjelmamuistia löytyy ATmega-sarjan mikrokontrollereista 4 - 256 kt. Muisti on jaettu 2 - 128 k x 16-bittiseen osaan, koska kaikki AVR-käskyt ovat 16 tai 32 bittiä pitkiä. Ohjelmiston turvallisuutta ajatellen ohjelmamuisti on jaettu kahteen osaan, esilataajan- ja sovellusohjelman-osioihin. Flash-ohjelmamuisti kestää vähintään 10 000 kirjoitusjaksoa. [2]

SRAM-muisti sisältää rekisteritiedoston, I/O-muistin, laajennetun I/O-muistin ja sisäisen datamuistin. Ensimmäiset 32 osoitetta ovat rekisteritiedostoon, sen jälkeen 64 osoitetta I/O-muistiin ja 160 osoitetta laajennettuun I/O-muistiin. SRAM-muistialueen loppu on datamuistia, jota löytyy esim. ATmega88-mikrokontrollerista 1 024 t. [2]

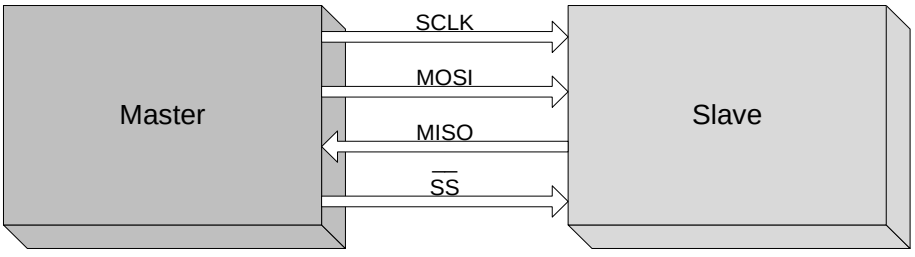
ATmega-sarjan mikrokontrollereissa on EEPROM-muistia 256 - 1 024 t. EEPROM-muisti on hitaampaa kuin SRAM-muisti ja sitä käytetään usein ohjelman asetuksien yms. tietojen pidempiaikaiseen tallentamiseen. Muisti sijaitsee erillisellä muistialueella, josta voidaan sekä lukea että kirjoittaa yksittäisiä tavuja tätä varten olemassa olevien rekisterien kautta. EEPROM-muisti kestää vähintään 100 000 kirjoitusjaksoa. [2]

2.1.3 Oheislaitteet

AVR-mikrokontrollereista löytyy paljon integroituja oheislaitteita. Esimerkiksi seuraavat oheislaitteet löytyvät ATmega88-mikrokontrollereista: Kolme laskuria, joista kaksi 8-bittisiä ja yksi 16-bittinen. Kuusi pulssinleveysmoduloitua kanavaa. Kuusikanavainen A/D-muunnin ja analoginen komparaattori. Tiedonsiirtoa varten löytyy SPI, USART sekä kaksijohtoinen I^2C yhteensopiva sarjaväylä TWI. [2]

2.2 SPI

SPI (Serial Peripheral Interface Bus) on Motorolan kehittämä synkroninen sarjaliikenneväylä. Väylä toimii full-duplex-tilassa, eli tietoa voidaan siirtää molempiin suuntiin samanaikaisesti. Laitteet keskustelevat isäntä/orja periaatteella, ja ainoastaan isäntä voi aloittaa tiedonsiirron. Kuvassa 3 näkyy SPI-väylä yksinkertaisimmassa muodossaan. [3]



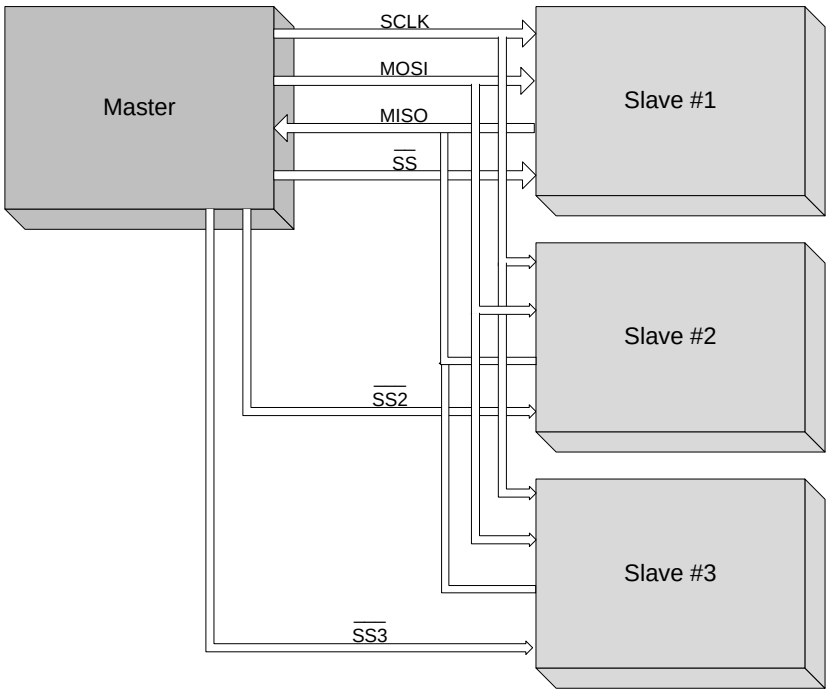
Kuva 3. Yhden isännän ja orjan SPI-väylä.

SPI-väylästä ei ole tarkkaa standardia, vaan monilla valmistajilla on hieman erilaisia ratkaisuja. Sarjakellon taajuus, polariteetti ja vaihe vaihtelevat laitteiden välillä. Osa laitteista ainoastaan vastaanottaa tietoa, ja osa taas osaa vain lähettää tietoa. Viestien pituudet vaihtelevat 8, 12 ja 16 bitin välillä. Näiden erojen takia mikrokontrollereissa on asetuksia, joiden avulla väylä saadaan yhteensopivaksi useiden eri laitteiden kanssa. [3]

2.2.1 Väylän toiminta

SPI-väylä sisältää 4 datalinjaa: SCLK (serial clock), MOSI (master output, slave input), MISO (master input, slave output) ja SS (slave select). SCLK on isännän generoima kellosignaali. MOSI on datalinja isännältä orjalle. MISO on datalinja orjalta isännälle. SS linjalla isäntä valitsee orjan, jonka kanssa haluaa aloittaa tiedonsiirron. [3]

SPI-väylässä voi olla yksi isäntä- ja yksi tai useita orja-laitteita. Jokainen orja-laite tarvitsee oman ohjauslinjansa (slave select), muut linjat ovat yhteisiä kaikkien orja-laitteiden kesken (kuva 4). Useissa orja-laitteissa on kolmitilaiset ulostulot ja niiden MISO-linja menee korkea impedanssiseen tilaan, ellei kyseisen laitteen SS-linja ole aktiivinen. Tämä mahdollistaa MISO-linjan jakamisen monen orjan välillä. [3]

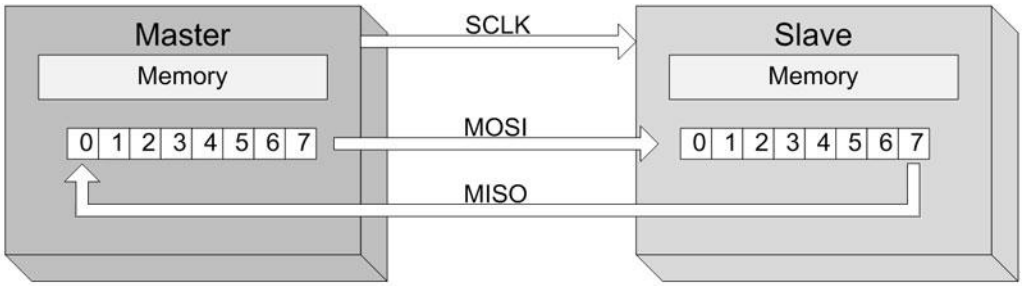


Kuva 4. Yhden isännän ja kolmen orjan SPI-väylä.

2.2.2 Tiedonsiirto

Tiedonsiirron mahdollistamiseksi isännän on asetettava SPI-väylän kello orjan tukemaan maksimitaajuuteen tai sitä pienempään arvoon. Orja-laitteen väyläkelloa ei lukita tiettyyn arvoon, vaan orja näytteistää kellosignaalia ja toimii sen tahdissa. Nopeudet liikkuvat usein 1 - 70 MHz:n alueella. Kellon vaihe ja polariteetti täytyy myös asettaa samaksi kuin orja-laitteessa. [3]

Ennen tiedonsiirron aloittamista isäntä laittaa siirtorekisteriin tavun, jonka haluaa lähettää orjalle. Seuraavaksi isäntä valitsee haluamansa orja-laitteen vetämällä tämän SS-linjan alas, jonka jälkeen se aloittaa kellopulssin luomisen. Jokaisen kellojakson aikana yksi bitti siirtyy isännältä orjalle ja orjalta isännälle. Tieto siirtyy aina molempiin suuntiin vaikka toisella laitteella ei olisikaan mitään hyödyllistä tietoa lähetettävänä. Tiedonsiirrossa käytetään tavallisesti kahta renkaaksi kytkettyä siirtorekisteriä (kuva 5). Siirtorekisterit ovat usein kahdeksan bitin mittaisia, ja toinen niistä sijaitsee isännässä ja toinen orjassa. [3]



Kuva 5. SPI-väylän siirtorekisterit.

Rekisterin sisältö siirretään usein eniten merkitsevä bitti edellä, ja samalla kellojaksolla rekisteriin siirretään uusi vähiten merkitsevä bitti. Kun koko rekisterien sisältö on vaihdettu keskenään, isäntä nostaa SS-linjan takaisin ylös ja laitteet tekevät vastaanotetulle tiedolle mitä haluavat (esim. kirjoittavat tavun muistiin). Jos tietoa tarvitsee siirtää lisää, siirtorekistereihin ladataan uudet arvot ja prosessi aloitetaan alusta. SPI-väylässä ei ole minkäänlaista rautapohjaista datavuonohjausta tai kuittausta, joten isäntä ei välttämättä saa koskaan tietää, vastaanottiko kukaan hänen lähettämäänsä viestiä. Tämä täytyy ottaa huomioon väylää käyttäviä ohjelmistoja suunnitellessa. [3]

2.2.3 Asetukset

SPI-väylän nopeus on mahdollista valita jakajien avulla. Väylän kellotaajuus muodostetaan järjestelmäkellosta joka jaetaan valitulla jakajalla, esim. 8 MHz:n järjestelmäkello ja 16 jakaja muodostaa 0,5 MHz:n väylän. Jakajia löytyy usein väliltä 4 – 256. Jotta väylä voisi toimia, on matalan ja korkean tilan kestettävä vähintään 2 kellojaksoa. [2]

SPI-väylällä on neljä erilaista tiedonsiirtomoodia, jotka ovat kellon polariteetin ja vaiheen yhdistelmiä (taulukko 3). CPOL (clock polarity) määrittää kellon polariteetin. Kun CPOL=0 ollaan normaalissa tilassa eli kellosignaali alkaa 0-tilasta, ja kun CPOL=1 kellosignaali invertoidaan (eli laskeva reuna = nouseva reuna kellon vaihetta ajatellessa). CPHA (clock phase) eli kellon vaihe määrittää luetaanko tieto väylästä kellopulssin laskevalla vai nousevalla reunalla, ja samoin kirjoitetaanko väylään nousevalla vai laskevalla reunalla. Kun CPHA=0 luetaan nousevalla reunalla ja kirjoitetaan laskevalla reunalla. [2]

Taulukko 3. SPI-moodit. [2]

SPI moodi	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

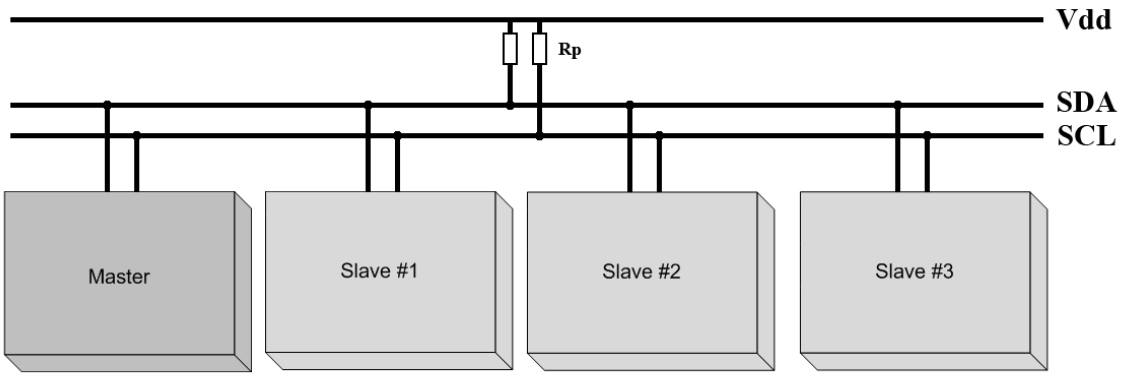
Useissa laitteissa tiedonlähetysjärjestys (DORD, data order) on myös valittavissa. Tällä asetuksella määritetään aloitetaanko tiedonsiirto vähiten vai eniten merkitsevästä bitistä. Kun DORD=0 eniten merkitsevä bitti lähetetään ensin. [2]

2.3 I^2C

I^2C on Philipsin kehittämä usean isännän sarjaliikenneväylä. Väylää käytetään hitaiden oheislaitteiden liittämiseen emolevyihin yms. sulautettuihin järjestelmiin. Useilla valmistajilla on markkinoilla täysin I^2C-yhteensopivia tuotteita (esim. Atmelilla TWI), ja nykyisin I^2C-väylän implementoinnista ei tarvitse maksaa lisenssimaksuja. [4]

2.3.1 Väylän toiminta

I^2C käyttää vain kahta kaksisuuntaista datalinjaa: sarjadataa (SDA) ja sarjakelloa (SCL). Datalinjat on vedetty ylös vastuksien avulla ja käytetyt jännitteet ovat tyypillisesti 5 tai 3,3 V. Esimerkki I^2C-väylästä näkyy kuvassa 6. [4]



Kuva 6. Yhden isännän ja kolmen orjan I^2C-väylä.

I^2C:n referenssimallissa on 7-bittinen osoiteavaruus ja osoitteista on varattu 16, joten yhdessä väylässä voi olla maksimissaan 112 laitetta. Väylässä voi olla vain kahden tyyppisiä laitteita, isäntiä ja orjia. Laitteet voivat vaihtaa rooleja viestien välissä. [4]

Useimmiten käytetyt nopeudet ovat 100 kb/s (standard mode) ja 10 kb/s (low-speed mode). Uudemmat revisiot sisältävät 10 bittisen osoiteavaruuden ja enemmän tiedonsiirtonopeuksia (400 kb/s fast mode, 1 Mb/s fast mode plus ja 3,4 Mb/s high speed mode). Ilmoitetut nopeudet sisältävät osoitteet ja kuittausbitit, joten todellinen tiedonsiirtonopeus on ilmoitettua pienempi. [4]

2.3.2 Tiedonsiirto

Isäntä käynnistää tiedonsiirron lähettämällä START-bitin ja haluamansa orjan 7-bittisen osoitteen. Tämän jälkeen lähetetään vielä yksi bitti, joka kertoo haluaako isäntä lähettää vai vastaanottaa (0 = lähetys, 1 = vastaanotto) dataa. Jos orja on väylässä, se vastaa ACK-

bitillä (0) ja isäntä jatkaa lähettämistä/vastaanottoa. Osoite ja data lähetetään eniten merkitsevä bitti edellä. Isäntä lähettää tavun kerrallaan, orjan kuitatessa ACK-bitillä jokaisen tavun jälkeen. Kun taas isäntä vastaanottaa, se kuittaa jokaisen orjan lähettämän tavun, viimeistä tavua lukuun ottamatta. Isäntä lopettaa tiedonsiirron lähettämällä STOP-bitin, tai toisen START-bitin halutessaan pitää väylä hallinnassaan ja aloittaa uuden tiedonsiirron. [4]

START-bitti ilmaistaan SDA-linjan muutoksella ylhäältä-alas, SCL-linjan ollessa ylhäällä. STOP-bitti taas ilmaistaan SDA-linjan muutoksella alhaalta-ylös, SCL-linjan ollessa ylhäällä. [4]

2.4 USART

USART (Universal Synchronous and Asynchronous Receiver/Transmitter) on sarjaliikennepiiri, joka muuttaa rinnakkaismuotoista dataa sarjamuotoiseen ja päinvastoin. USARTia käytetään yleensä laitteiden väliseen tiedonsiirtoon tietokoneen tai muun oheislaitteen sarjaportin kanssa. USART on kehittyneempi versio UARTista, sisältäen synkronisen tiedonsiirron. USART löytyy usein integroituna mikrokontrollereista. [5]

2.4.1 Väylän toiminta

UART-ohjain on tietokoneen sarjatietoliikennejärjestelmän tärkein komponentti. UART ottaa tavuja ja lähettää ne yksittäisinä bitteinä sarjamuodossa eteenpäin. Vastaanottajan UART kokoaa bitit takaisin kokonaisiksi tavuiksi. Tiedon siirtäminen yhtä johdinta pitkin on paljon kustannustehokkaampaa, kuin tiedon siirtäminen rinnakkaismuodossa useita johtimia pitkin. Tietoa voidaan siirtää half-duplex- tai full-duplex-tilassa. [5]

UART ei yleensä generoi tai vastaanota suoraan tiedonsiirrossa käytettyä ulkoista signaalia. Tavallisesti käytössä on erillinen piiri, joka muuttaa UARTin logiikkatasoiset signaalit siirrossa käytettäviin tasoihin. Ulkoisia signaaleita on monenlaisia. Usein käytettyjä standardeja ovat esim. EIA:n (Electronic Industries Alliance) RS-232, RS-422 ja RS-485. [5]

2.4.2 Tiedonsiirto

Asynkronisessa tiedonsiirrossa lähetetään ensin START-bitti jota seuraa 5 - 8 databittiä, pariteettibitti sekä 1 - 2 STOP-bittiä. Data lähetetään vähiten merkitsevä bitti edellä. START-bitin polariteetti on vastakkainen väylän lepotilaan nähden. STOP-bitti taas on sama kuin väylän lepotila ja se toimii viiveenä ennen seuraavan merkin lähetystä. Pariteettibitti tekee lähetettyjen 1-bittejen määrän parilliseksi tai parittomaksi. Parittomuus on luotettavampi menetelmä, sillä se takaa että lähetyksessä on ainakin yksi data transitio, joka taas mahdollistaa monien UARTien uudelleen synkronoitumisen. Jossain tapauksissa pariteettibitti jätetään pois. Jotta tiedonsiirto voisi toimia, laitteiden kellotaajuuksissa ei saa olla yli 10 % eroa. [5]

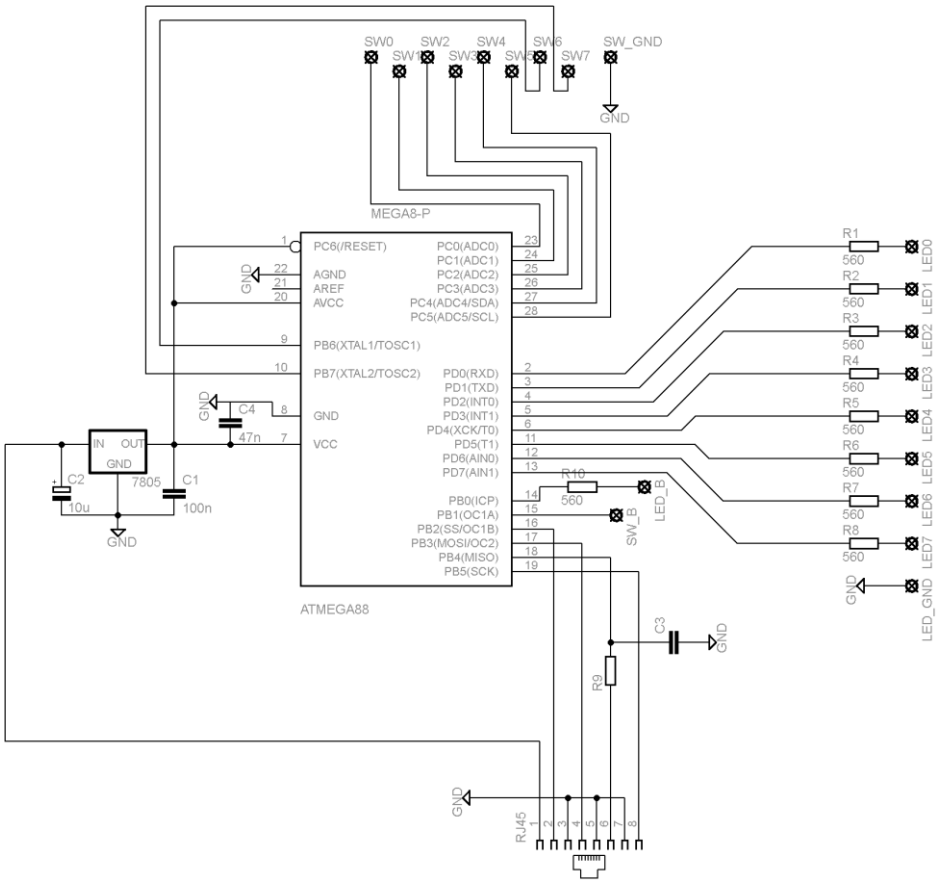
Synkronisessa tiedonsiirrossa kellodata erotetaan tietovirrasta, eikä start/stop-bittejä käytetä. Tämä parantaa tiedonsiirron tehokkuutta, sillä suurempi osa siirrettävistä biteistä on dataa. Asynkronisessa tiedonsiirrossa ei lähetä mitään väylän ollessa toimeettomana, kun taas synkronisessa väylässä lähetetään täytemerkkejä, jotta laitteet pysyvät synkronoituna. Tähän käytetään usein ASCII SYN -merkkiä, ja lähettävä laite hoitaa merkin lähetyksen usein automaattisesti. [5]

3 Toteutus

Laitteita varten täytyi suunnitella kolme erillistä piirilevyä. Ohjausyksikkö sisältää yhden ja liitäntäyksikkö kaksi piirilevyä. Laitteet yhdistetään toisiinsa RJ45-kaapelilla. Kaapelissa kulkee SPI-väylän MISO-, MOSI-, SCK- ja SS-linjat sekä 9 V:n käyttöjännite. Laitteissa käytetään Atmelin ATmega88-mikrokontrollereita.

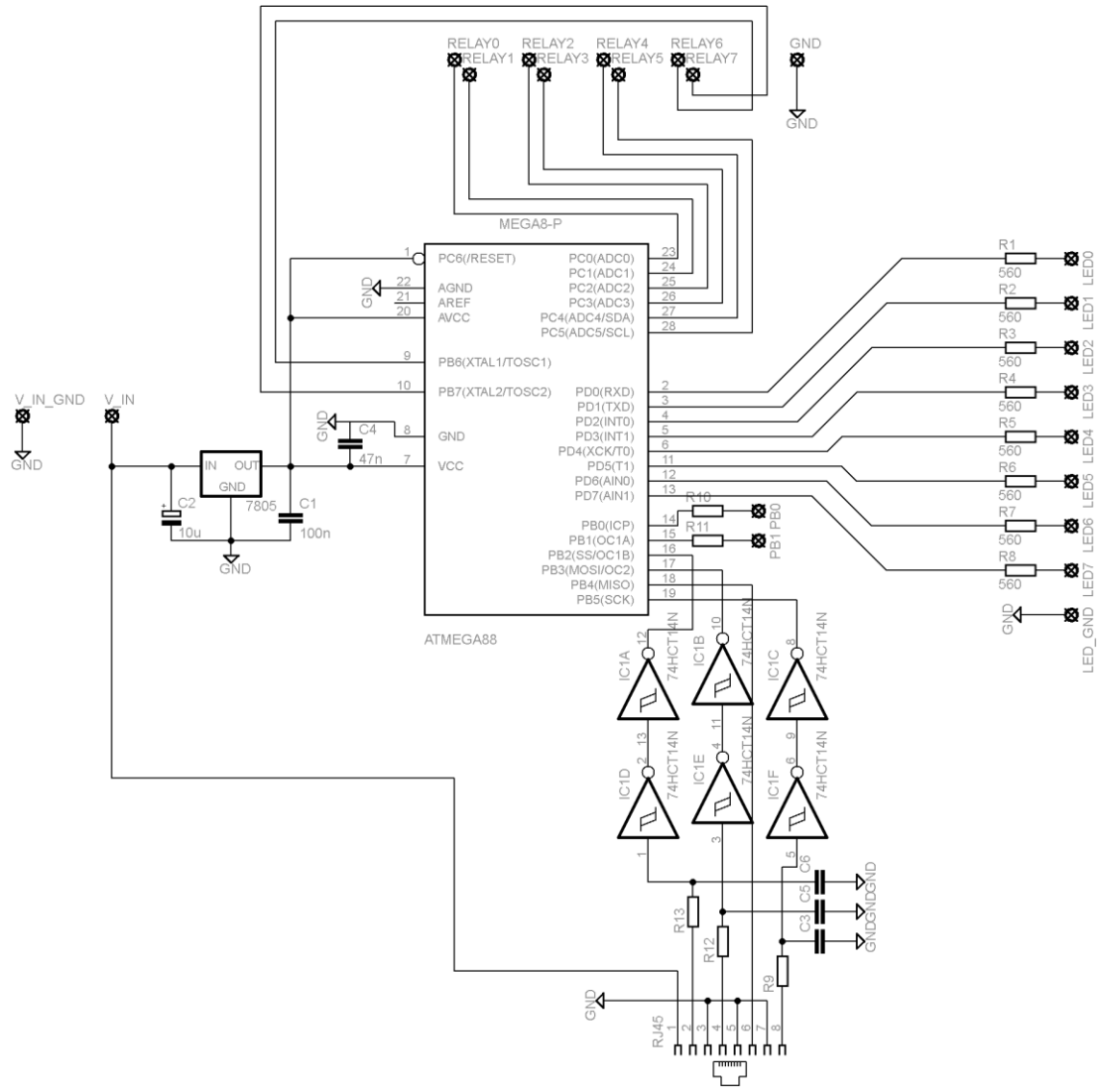
3.1 Kytkentäkaaviot

Ohjausyksikön piirilevy sisältää mikrokontrollerin, johon on kytketty 9 kytkintä, 9 lediä ja RJ45-liitin. RJ45-kaapelista tuleva käyttöjännite reguloidaan mikrokontrollerille sopivaan 5 V:iin. MISO-linjaan laitettiin alipäästösuodatin häiriönpoistoa varten. Ohjausyksikön kytkentäkaavio näkyy kuvassa 7.



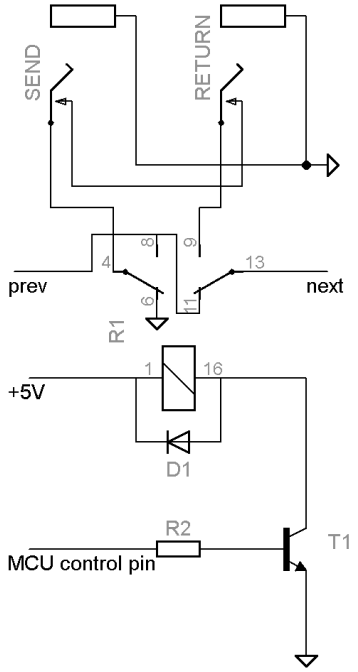
Kuva 7. Ohjausyksikön kytkentäkaavio.

Liitäntäyksikössä on 2 piirilevyä, 8 lediä, 18 6,3 mm:n monoliitintä, 6,3 mm:n stereoliitin, RJ45-liitin ja DC-liitin. Ensimmäinen levy sisältää mikrokontrollerin, joka ohjaa kahdeksaa lediä ja relettä. Laitteeseen tulee 9 V:n käyttöjännite ulkoisesta muuntajasta. Jännitteen väärinpäin kytkeminen on estetty diodin avulla. Jännite ohjataan RJ45-kaapelia pitkin ohjausyksikköön ja reguloidaan 5 V:iin mikrokontrolleria ja releitä varten. Sisääntuleviin datalinjoihin laitettiin alipäästösuotimet ja Schmitt-invertterit häiriönpoistoa varten. Schmitt-triggerit korjaavat vääristyneen signaalin tasaiseksi kanttiaalloksi. Tämä on erittäin tärkeää varsinkin kellosignaalin kohdalla. Inverttereitä laitettiin kaksi peräkkäin, jotta signaalien vaihe saatiin alkuperäiseen tilaan. Kyseisen piirilevyn kytkentäkaavio on nähtävillä kuvassa 8.



Kuva 8. Liitäntäyksikön ensimmäisen piirilevyn kytkentäkaavio.

Liitäntäyksikön toinen piirilevy sisältää releet sekä niiden ohjaamiseen tarvittavat komponentit. Äänisignaalia ohjaavia releitä on levyllä seitsemän kappaletta. Kahdeksatta relettä käytetään vahvistimen tai muun laitteen ohjaamiseen (esim. vahvistimen kanavaa voidaan vaihtaa sen avulla). Kuvassa 9 näkyy yksi seitsemästä äänisignaalin ohjaamiseen käytettävästä releasosta.



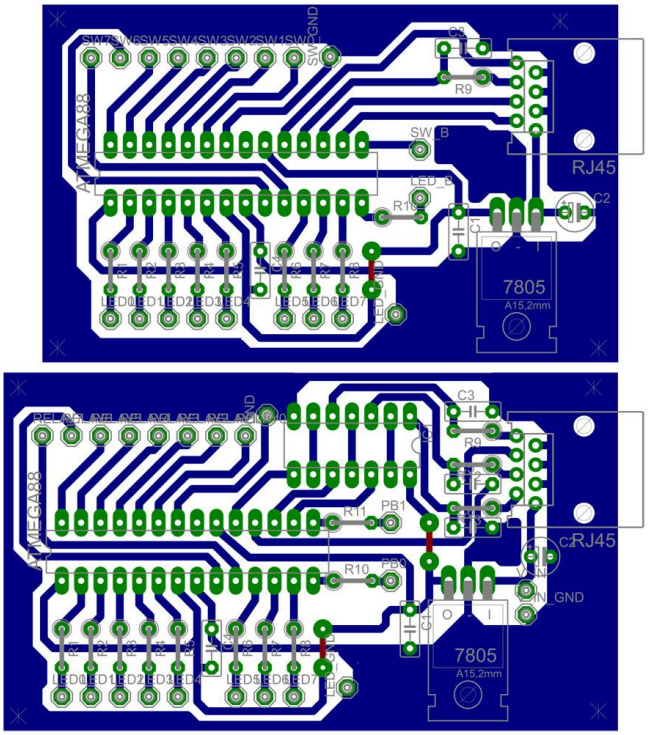
Kuva 9. Yksi äänisignaalin ohjaamiseen käytettävä releasoso.

Releet sisältävät kaksi vaihtokytkintä. Äänisignaali ohjataan liittimille tai niiden ohi mikrokontrollerin ohjauslinjan tilan mukaan. Liittimet sisältävät kytkimet, jotka ohjaavat äänisignaalin eteenpäin, jos liittimiin ei ole kytketty mitään. Kun ohjauslinja on nollassa, rele on lepotilassa ja äänisignaali pääsee suoraan seuraavalle tasolle. Kun taas ohjauslinja on aktiivinen (+5 V), transistori maadoittaa releen ja äänisignaali ohjataan liittimien kautta seuraavalle tasolle.

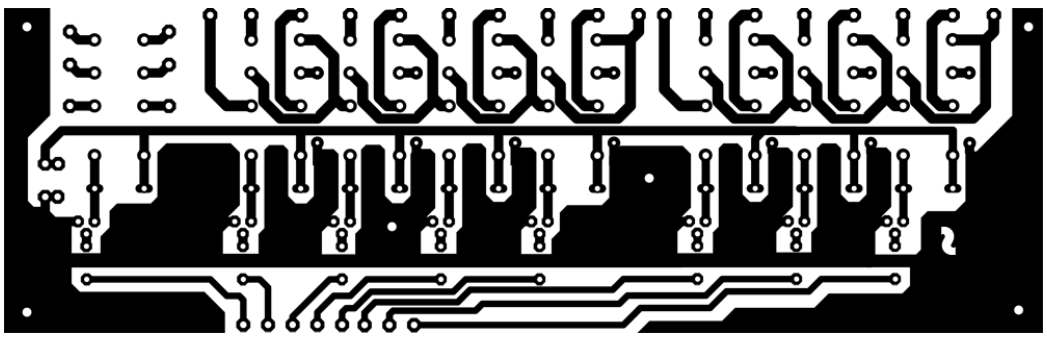
Releet on kytketty kahteen piiriin. Ensimmäisessä on kolme relettä ja toisessa neljä. Näitä voidaan käyttää erillisten äänisignaalien ohjaamiseen (esim. ensimmäinen kitaran ja vahvistimen väliin, ja toinen vahvistimen efektiluuppiin). Piirit voidaan yhdistää kytkemällä kaapeli ensimmäisen piirin ulostulosta toisen piirin sisääntuloon, jos kahdelle erilliselle piirille ei ole käyttöä.

3.2 Piirilevyjen valmistus

Komponenttien sijoituksessa täytyi ottaa huomioon muutamia seikkoja. Alipäästösuotimet täytyi saada mahdollisimman lähelle RJ45-liittimiä suodatuksen toiminnan varmistamiseksi. Ohituskondensaattorit sijoitettiin taas mahdollisimman lähelle mikronkontrollereita. Ylimääräinen tila täytettiin maatasolla. Kuvassa 10 on ohjaus- ja liitäntäyksikön mikrokontrollerit sisältävät piirilevyt komponenttipuolelta kuvattuna. Kuvassa 11 näkyy liitäntäyksikön relepiirilevy. Piirilevyjen suunnitteluun käytettiin Cadsoftin Eagle-ohjelmaa [6].



Kuva 10. Ohjausyksikön (yllä) ja liitäntäyksikön (alla) piirilevyt.

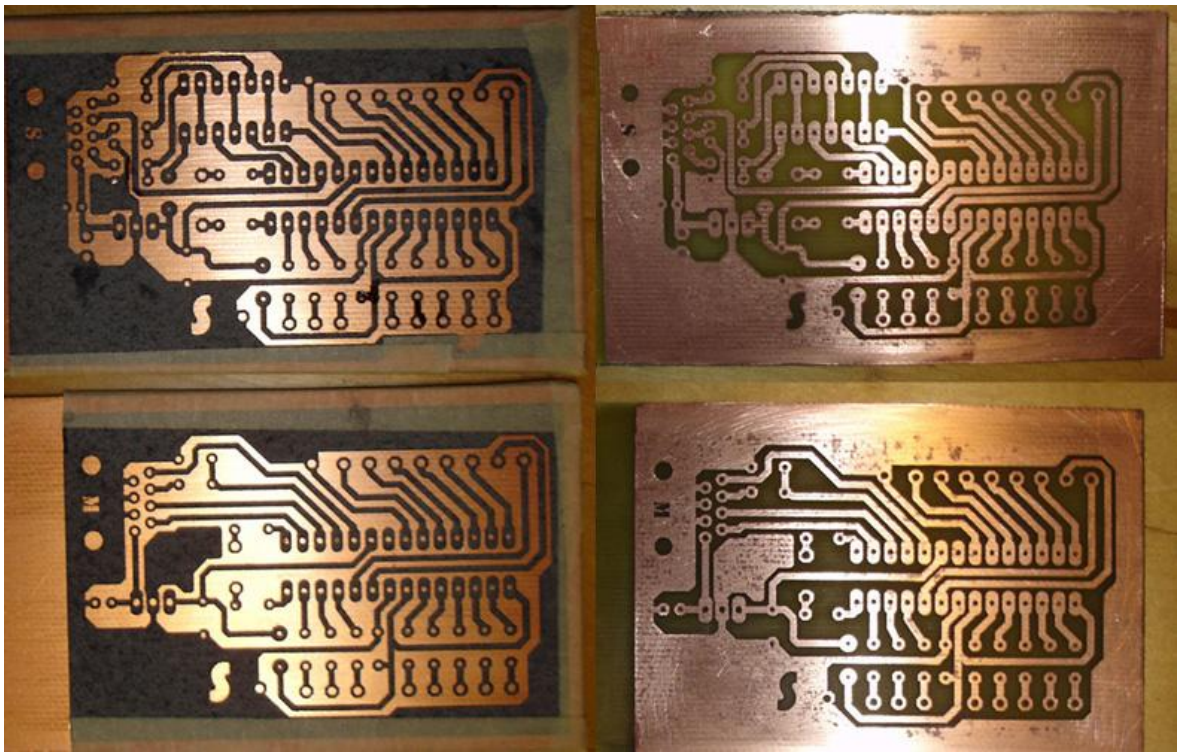


Kuva 11. Liitäntäyksikön relepiirilevy.

Piirilevyt valmistettiin lämpösiirtotekniikkaa hyväksikäyttäen: Ensin haluttu kuva tulostetaan peilikuvana lasertulostimella kiiltäväpintaiselle paperille (esim. valokuvapaperille). Seuraavaksi piirilevyn kuparipinta puhdistetaan hienolla hiekkapaperilla ja liuottimella. [7]

Tämän jälkeen tulostettu kuva laitetaan piirilevyn päälle mustepuoli kuparia kohti. Paperi silitetään piirilevyyn kiinni silitysraudalla tasaisesti lämmittäen. Kun piirilevy on jäähtynyt, se laitetaan veteen, ja muutaman minuutin kuluessa paperi irtoaa piirilevystä. Piirilevystä tarvitsee vielä poistaa kaikki ylimääräiset paperijäämät kevyesti vedellä hieromalla. [7]

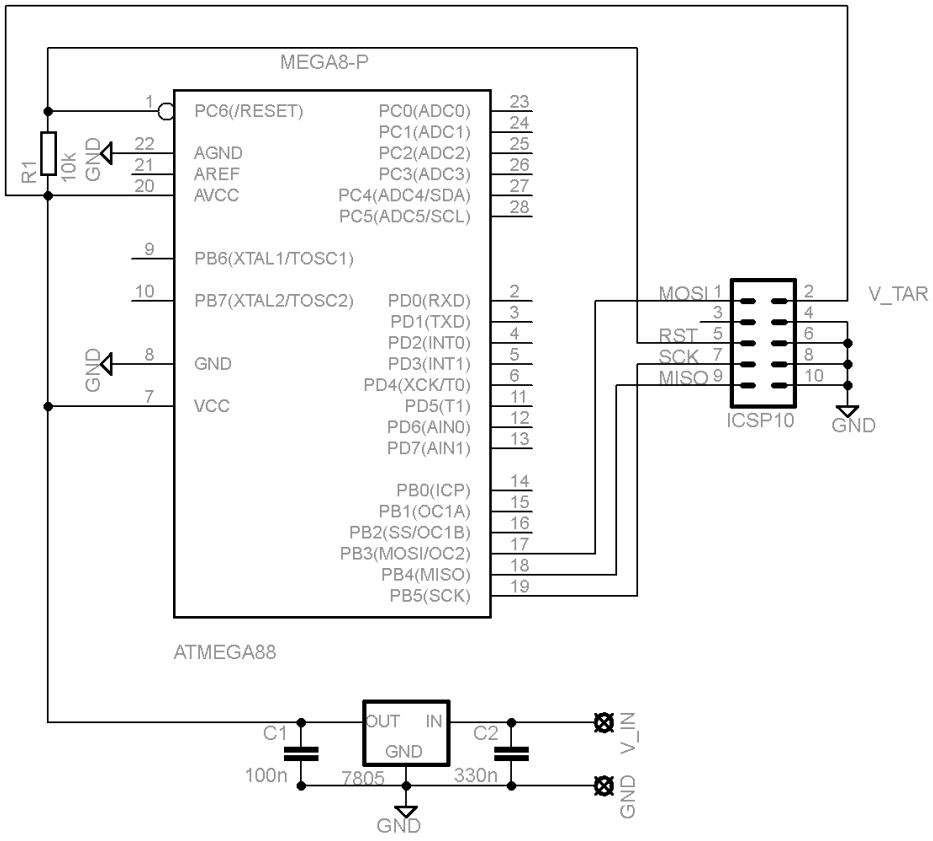
Tässä vaiheessa tarkistetaan, että kuva on ehjä, ja korjataan mahdolliset puutteet. Ylimääräinen kupari syövytetään piirilevyistä ferrikloridilla, ja tulostimen muste poistetaan lopuksi hiomalla tai liuottimella pyyhkimällä. Kuvassa 12 näkyvät piirilevyt ennen syövytystä ja sen jälkeen. [7]



Kuva 12. Piirilevyt ennen (vasemmalla) ja jälkeen (oikealla) syövytyksen.

3.3 Ohjelmointilaite

Ohjelmointilaitteeksi valittiin Olimex AVR-ISP500. AVR-ISP500 on USB-väylään liitettävä ohjelmointilaite, joka tukee STK500v2-protokollaa. Tällä laitteella voidaan ohjelmoida AVR-mikrokontrollereiden Flash- ja EEPROM-muistit sekä pystytään muokkaamaan sulakebittejä. Ohjelmointilaitteesta löytyy Atmelin standardit ICSP6- ja ICSP10-liittimet. ATmega88-mikrokontrollereiden ohjelmointia varten rakennettiin erillinen piirilevy, jonka kytkentäkaavio löytyy kuvasta 13. [8]



Kuva 13. ATmega88-ohjelmointiadapterin kytkentäkaavio.

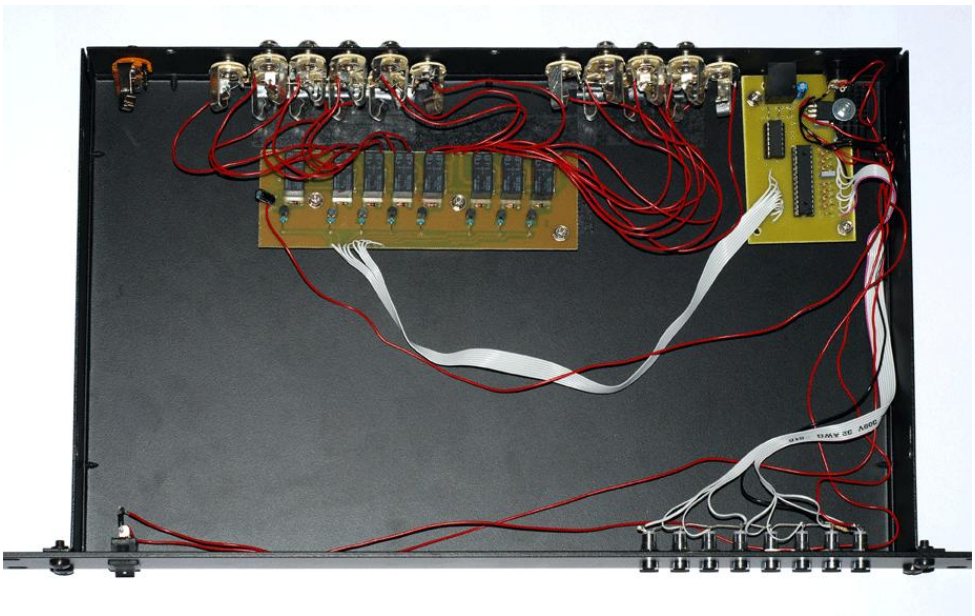
3.4 Laitteiden kotelointi

Molempiin laitteisiin valittiin metalliset kotelot. Ohjausyksikön kotelon täytyi olla kestävä, koska sitä käytetään jaloilla. Ohjausyksikössä on ainoastaan yksi RJ45-liitin sekä käyttöliittymän ledit ja kytkimet. Koteloon porattiin tarvittavat reiät, ja se maalattiin mustaksi. Kuvassa 14 ohjausyksikkö näkyy keskeneräisenä ja valmiina.

Kuva 14. Ohjausyksikkö.

Liitäntäyksikön koteloksi valittiin standardin mukainen 44,45 mm (1 U) korkea 19 ”:n räkkikotelo. Koteloon porattiin reiät liittimille ja ledeille. Kaikki liittimet ovat laitteen takalevyssä. Laitteen runko on maadoitettu äänisignaalin suojaamiseksi. Liitäntäyksikkö näkyy kuvassa 15 ulkoa ja kuvassa 16 sisäpuolelta.

Kuva 15. Liitäntäyksikkö edestä ja takaa.



Kuva 16. Liitäntäyksikkö sisältä.

4 Ohjelmisto

Molempien laitteiden mikrokontrollereita varten täytyi kirjoittaa ohjelmat. Mikrokontrollerit käyttävät sisäisiä kiteitä kahden jaolla, jolla saadaan aikaan 4 MHz:n kellotaajuus. Tiedonsiirtoon käytetään SPI-väylää 128:n jaolla, jolloin väylän nopeus on noin 31,25 kHz. Laitteiden välistä tiedonsiirtoa varten täytyi kehittää oma tiedonsiirtoprotokolla.

Ohjelmien kirjoittamiseen käytettiin AVR Studio 4:ää ja WinAVR-ympäristöä [9]. Ohjelmat kirjoitettiin C-kielellä. Työssä käytettiin iteratiivisen kehittämisen menetelmiä.

4.1 Tiedonsiirtoprotokolla

Tiedonsiirto tapahtuu viiden tavun paketeissa, koska vain isäntä voi aloittaa tiedonsiirron. Tieto siirretään half-duplex-tilassa. Ensimmäiset kolme tavua ovat isännältä orjalle, ja viimeiset kaksi ovat orjalta isännälle.

Ensimmäinen tavu sisältää aina komennon. Komentojen bittikuviot tehtiin tarkoituksella mahdollisimman yksinkertaisiksi tiedonsiirron helpottamiseksi (esim. 0xFF = 1111 1111 ja 0xFC = 1111 1100). Toinen tavu sisältää komentoon liittyvää tietoa, esim. muistipaikkaa vaihtaessa halutun muistipaikan numeron. Kolmas tavu sisältää tarkistussumman. Neljäs tavu on orjan vastaus isännälle. Jos komentoon ei ole informatiivista vastausta, tavu sisältää 0xF0. Viides tavu on orjan vastauksen tarkistussumma. Kaikki tiedonsiirtoprotokollan komennot näkyvät taulukossa 4.

Taulukko 4. Tiedonsiirtoprotokollan komennot.

	Isäntä => Orja			Orja => Isäntä	
Komento	Tavu 1	Tavu 2	Tavu 3	Tavu 4	Tavu 5
Vaihda muistipaikkaa	0xFF	Muistipaikan numero	**TARKISTUSSUMMA (tavu1 + tavu2) ^ 0xFF**	0xF0	**TARKISTUSSUMMA tavu4 ^ 0xFF**
Tämänhetkinen muistipaikka?	0xFC	0x00		Muistipaikan numero	
Muistipaikan sisältö?	0xF3	Muistipaikan numero		Muistipaikan sisältö	
Tallenna muistipaikkaan	0xCF	Muistipaikan sisältö		0xF0	
Tallenna EEPROM:iin	0xCC	Muistipaikan numero		0xF0	

Tarkistussummista tehtiin mahdollisimman yksinkertaiset siirrettävän tiedon pienen määrän takia. Isännän lähettämien kahden tavun tarkistussumma lasketaan seuraavalla tavalla: (tavu1 + tavu2) ^ 0xFF. Orjan on helppo tarkistaa vastaanotetun tiedon eheys laskemalla kaikki kolme vastaanotettua tavua yhteen, jolloin tuloksen tulee olla 0xFF. Isännän tiedonsiirtoon käyttämät funktiot näkyvät ohjelmassa 1.

Orjan lähettämästä tavusta otetaan tarkistussummaksi suoraan tavun komplementti. Isäntä voi tällöin laskea vastaanotetut tavut suoraan yhteen, ja tuloksen ollessa 0xFF tieto on siirtynyt ehjänä. Orjan tiedonsiirtoon käyttämät funktiot näkyvät ohjelmassa 2.

Tiedonsiirtoprotokollassa ei ole erillistä kuittausta, vaan se hoidetaan tarkistussummien avulla. Jos orjan laskema tarkistussumma ei täsmää, se vastaa lähettämällä tahallaan virheellisen tarkistussumman (molemmissa tavuissa 0x00). Isännän laskiessa vastaanottamaansa tarkistussummaa se huomaa virheen ja lähettää tiedon uudestaan. Isäntä yrittää lähettää paketin viisi kertaa, ennen kuin se ilmoittaa käyttäjälle virheestä.

Jos jostain syystä yksi tavu jää vastaanottamatta ja tavujen numerointi menee sekaisin, tarkistussummat eivät enää täsmää. Jos näin pääsee tapahtumaan, tiedonsiirto ei toimi enää ollenkaan. Tämän korjaamiseksi orja nollaa vastaanotettujen tavujen laskurin joka viidennen virheellisen paketin jälkeen.

Ohjelma 1. Isännän tiedonsiirtoon käyttämät funktiot.

```
uint8_t Transmit_bytes(uint8_t byte1, uint8_t byte2)
 {
   uint8_t retry = 0;
   uint8_t checksum = (byte1 + byte2) ^ 0xFF;

   while(1){
       SPI_MasterTransmit(byte1);
       SPI_MasterTransmit(byte2);
       SPI_MasterTransmit(checksum);
       WAIT(1);
       SPI_MasterTransmit(0x00);
       inc_byte = SPDR;
       SPI_MasterTransmit(0x00);
       inc_checksum = SPDR;

       if ( (inc_byte + inc_checksum) == 0xFF) { //if checksum ok return 0xFF
              return 0xFF;
              }
       retry++;
       checksum_failed++;
       if ( retry > 4) { //if checksum failed 5 times led flash and return 0x00
              Error();
              return 0x00;
              }
       WAIT(1);
  }
}

void SPI_MasterTransmit(uint8_t data)
 {
  PORTB &= ~(1<<2); // SS Down
  SPDR = data;
  while (!(SPSR & (1<<SPIF))); // wait for transmit to complete
  PORTB |= (1<<2);  // SS Up
}
```

Ohjelma 2. Orjan tiedonsiirtoon käyttämät funktiot.

```
ISR(SPI_STC_vect) // interrupt - byte received
 {
  inc_byte++;

    if(inc_byte == 1){
       inc_byte1 = SPDR;
              }
    if(inc_byte == 2){
       inc_byte2 = SPDR;
              }
    if(inc_byte == 3){
       inc_checksum = SPDR;
       Data_inc(inc_byte1, inc_byte2, inc_checksum);
              }
       if(inc_byte == 4){
       SPDR = out_checksum;
              }
    if(inc_byte == 5){
       inc_byte = 0;
              }

}
```

```
void Data_inc(uint8_t byte1, uint8_t byte2, uint8_t checksum)
  {
    uint8_t temp = byte1 + byte2 + checksum;
    if ( temp == 0xFF ){ // checksum ok
       switch( byte1 )
        {
       case 0xFF: // change memoryslot
              Data_out(0xF0);
              current_state = byte2;
              Output(memoryslot[current_state]);
              break;
       case 0xFC: // current_state?
              Data_out(current_state);
              break;
       case 0xF3: // memoryslot data?
              Data_out(memoryslot[byte2]);
              break;
       case 0xCF: // edit memoryslot
              Data_out(0xF0);
              memoryslot[current_state] = byte2;
              Output(memoryslot[current_state]);
              break;
       case 0xCC: // save data to eeprom
              Data_out(0xF0);
              eeprom_write_byte((uint8_t*)current_state,
              memoryslot[current_state]);
              break;
       default: // unknown command - return faulty checksum
              SPDR = 0x00;
              out_checksum = 0x00;
              Error();
        }
       }
       else { // checksum failed - return faulty checksum
       SPDR = 0x00;
       out_checksum = 0x00;
       Error();
       }
 }

void Error(void)
 {
       checksum_failed++;
       if(checksum_failed > 4){
       checksum_failed = 0;
       cli();
       reset = 0xFF;
       UI_delay();
       Output(memoryslot[current_state]);
       sei();
       }
}

void Data_out(uint8_t byte)
  {
       SPDR = byte;
       out_checksum = byte ^ 0xFF;
 }
```

4.2 Ohjausyksikkö (isäntä)

Ohjausyksikön käyttöliittymä koostuu yhdeksästä kytkimestä ja ledistä. Ensimmäiset kahdeksan kytkintä sisältävät kukin yhden muistipaikan. Kytkintä painaessa laite lähettää liitäntäyksikölle komennon vaihtaa muistipaikkaa. Tiedonsiirron onnistuessa kyseisen kytkimen päällä oleva ledi syttyy näyttäen käytössä olevan muistipaikan. Yhdeksäs kytkin vaihtaa muistipankin sivua, eli yhdeksännen ledin palaessa kytkimet sisältävät toiset kahdeksan muistipaikkaa. Kytkimien sijainnit näkyvät kuvassa 17.

Kuva 17. Ohjausyksikön kytkimien sijainnit.

Käytössä olevaa muistipaikkaa voidaan muokata painamalla kyseisen muistipaikan kytkintä uudelleen noin viiden sekunnin ajan. Ohjausyksikkö kysyy liitäntäyksiköltä muistipaikan sisällön, jonka jälkeen sitä voidaan muokata. Muokkaustilassa ensimmäiset kahdeksan kytkintä muuttavat jokainen yhden releen tilaa, jos efekti on kytketty, kytkimen päällä oleva ledi palaa. Yhdeksännen kytkimen ledi vilkkuu, jotta käyttäjä tietää olevansa muokkaustilassa. Yhdeksäs kytkin tallentaa muutokset liitäntäyksikön EEPROM-muistiin

ja vie ohjelman takaisin perustilaan. Tehdyt muutokset tapahtuvat reaaliajassa, joten käyttäjä kuulee tekemänsä muutokset välittömästi.

Tiedonsiirron toiminnan testaamista varten laitteeseen lisättiin muutama toiminto. Kun yhdeksättä kytkintä pitää pohjassa, kytkin 1 lähettää yksittäisiä tavuja tiedonsiirron sekoittamiseksi. Kytkin 5 näyttää tiedonsiirtovirheiden määrän binaari-muodossa laitteen ledeillä.

Laitteiden välisen kaapelin irtoaminen ei vaikuta releiden tilaan. Kun kaapeli kytketään takaisin, ohjausyksikkö kysyy käynnistyksen yhteydessä käytössä olevan muistipaikan ja toiminta jatkuu normaalisti.

4.3 Liitäntäyksikkö (orja)

Liitäntäyksikön tehtävä on ohjata äänisignaalia releiden avulla ohjausyksikön käskyjen mukaisesti. Muistipaikkojen sisältö (16 tavua) on tallennettu laitteen EEPROM-muistiin, josta ne luetaan käynnistyksen yhteydessä taulukkoon. Perustilassa laite ainoastaan virkistää releiden ohjauksia ja odottaa käskyjä ohjausyksiköltä. Laitteessa on kahdeksan lediä, jotka näyttävät releiden tilat (kuva 18). Ensimmäiset seitsemän lediä näyttävät äänisignaalin ohjaukseen käytettävien releiden tilat. Kahdeksas ledi on kaksivärinen ja toimii myös laitteen on/off-ledinä. Se palaa vihreänä ohjausreleen ollessa pois päältä ja punaisena sen ollessa päällä.

Kuva 18. Liitäntäyksikön ledit.

5 Testaus

Laitteiden toimintaa testattiin käytännössä kytkemällä liitäntäyksikköön maksimimäärä efektilaitteita ja suorittamalla testi laitteen tulevassa ympäristössä. Häiriönsietoa testattiin käärimällä kolvin virtajohto laitteiden välisen RJ45-kaapelin ympärille. Kun kolvin virta sekä kytketään että kytketään pois, aiheutuu kaapeliin häiriöpiikkejä.

5.1 Ohjelmisto ja tiedonsiirto

Laitteen elektroniikka otti tahallaan aiheutetuista häiriöistä välillä virheellisiä painalluksia. Tämä korjattiin ohjelmiston avulla, tarkistamalla onko kytkin edelleen painettuna noin 10 ms:n kuluttua ensimmäisestä painalluksesta.

Tiedonsiirtoa yritettiin hankaloittaa aiheuttamalla häiriöitä RJ45-kaapeliin, mutta toiminta oli niin varmaa, että alipäästösuotimet päätettiin jättää kokonaan kytkemättä. Virheen sattuessa tietoa yritetään siirtää viisi kertaa uudelleen, joten on erittäin epätodennäköistä, että kaikissa peräkkäisissä paketeissa olisi virhe.

5.2 Äänisignaalin ohjaus

Tiedonsiirto ja releiden toiminta on niin nopeaa, ettei muistipaikkaa vaihdettaessa ole huomattavissa minkäänlaista viivettä. Äänisignaaliin tulee pieni napsahdus aina releen vaihtaessa tilaa. Ääni on kuitenkin niin pieni signaalin tasoon verrattuna, että se on kuultavissa ainoastaan kovilla äänenvoimakkuuksilla, kun äänisignaali on olemattoman pieni.

Osa efektilaitteista kierrättää äänisignaalin laitteen elektroniikan läpi, vaikka efekti ei olisi käytössä. Tämä yleensä heikentää signaalia hieman ja ongelma alkaa kertaantua, kun tällaisia laitteita on kytketty monta peräkkäin. Äänisignaali pysyy huomattavasti parempana, kun efektilaitteet ohitetaan täysin releiden avulla.

6 Yhteenveto

Projekti onnistui hyvin. Kaikki asetetut tavoitteet saavutettiin, vaikka laitteisiin suunniteltu elektroniikka olikin hieman puutteellista. Elektroniikasta johtuvat ongelmat saatiin kuitenkin korjattua ohjelmiston avulla.

Laitteita on mahdollista kehittää vielä eteenpäin ohjelmiston avulla, kunhan ne saadaan pitempiaikaiseen käyttöön ja käyttäjiltä saadaan palautetta.

Lähteet

[1] Atmel AVR. [www-dokumentti]. Saatavilla
http://en.wikipedia.org/wiki/Atmel_AVR (luettu 27.4.2009)

[2] ATmega48/88/168 datasheet. [pdf-dokumentti]. Saatavilla
http://atmel.com/dyn/resources/prod_documents/doc2545.pdf (luettu 14.4.2009)

[3] Serial Peripheral Interface Bus. [www-dokumentti]. Saatavilla
http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus (luettu 15.4.2009)

[4] I^2C. [www-dokumentti]. Saatavilla
http://en.wikipedia.org/wiki/I%C2%B2C (luettu 17.5.2009)

[5] USART. [www-dokumentti]. Saatavilla
http://en.wikipedia.org/wiki/USART (luettu 16.8.2009)

[6] Cadsoft Eagle. [www-dokumentti]. Saatavilla
http://www.cadsoft.de (luettu 15.4.2009)

[7] Make PCBs at home. [www-dokumentti]. Saatavilla
http://www.riccibitti.com/pcb/pcb.htm (luettu 15.5.2009)

[8] Olimex AVR-ISP500. [pdf-dokumentti]. Saatavilla
http://www.olimex.com/dev/pdf/AVR/AVR-ISP500.pdf (luettu 10.5.2009)

[9] WinAVR. [www-dokumentti] Saatavilla
http://winavr.sourceforge.net (luettu 10.4.2009)