TAMPEREEN TEKNILLINEN YLIOPISTO

ANTTI VARJONEN

KOTIAUTOMAATIOJÄRJESTELMÄ

Diplomityö

Tarkastaja: Karri Palovuori
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan
tiedekuntaneuvoston
kokouksessa 05.12.2012

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma
Varjonen, Antti: Kotiautomaatiojärjestelmä
Diplomityö, 42 sivua, 15 liitesivua
Marraskuu 2013
Pääaine: Elektroniikka: Elektroniikan laitesuunnittelu
Tarkastajat: Professori Karri Palovuori
Avainsanat: modulaarinen, kotiautomaatio, Zigbee

Työn tavoitteena oli rakentaa modulaarinen kotiautomaatiojärjestelmä ja tutustua langattomaan Zigbee -standardiin. Langattomaan väylän käyttöön päädyttiin, koska vanhoihin asuntoihin ei välttämättä ole mahdollista rakentaa langallista väylää.

Järjestelmä koostuu koordinaattorimoduulista ja neljästä apumoduulista. Koordinaattori hoitaa apumoduuleiden välisten viestien välityksen. Apumoduuleita ovat infrapuna-, rele-, näppäimistö- ja lämpömittarimoduuli. Infrapunamoduulin tarkoitus on ohjata kotitalouden kaukosäätimellä ohjattavia laitteita, kuten esimerkiksi televisio tai audiovahvistin. Tässä työssä moduulin ohjaus on rajattu Sonyn laitteille. Relemoduuli ohjaa verkkovirtaan kytkettyjä laitteita kytkemällä tai katkaisemalla laitteen käyttöjännitteet. Näppäimistömoduuli lähettää tiedot painetuista nappuloista ja välittää ne koordinaattorimoduulille. Lämpömittarimoduuli mittaa lämpötilaa ja lähettää lämpötila-arvon koordinaattorimoduulille havaittuaan muutoksen. Koordinaattori tulostaa lämpötila-arvot testauslohkon avulla tietokoneella käytettävälle terminaali ohjelmalle.

Järjestelmälle ohjelmoitiin kaksi kirjastoa. Yksi moduuleiden käyttämälle Zigbee-moduulille ja yksi Sony:n infrapunaprotokollalle. Zigbee-moduuleiden kirjasto hoitaa suurimman osan järjestelmän moduuleiden toiminnasta, eli tiedonsiirron. Käyttämämme Zigbee-moduuli tuottaa keskeytyssignaalin, kun se on saanut lähetettyä tai vastaanotettua dataa. Infrapunaprotokollan kirjasto on toteutettu käyttäen mikrokontrollerin ajastinkeskeytyksiä.

Tiedonsiirto on osoittautunut erittäin luotettavaksi. Testausaikana ei havaittu yhtään epäonnistunutta lähetystä. Lämpömittarimoduulin lämpötilat eroavat hieman testaukseen käytetyn kuluttajille suunnatun lämpömittarin arvoista, mutta lämpötila-arvot pysyvät samassa suuruusluokassa. Infrapunamoduulille ohjelmoitu kirjasto toimii hyvin ja vastaa protokollan vaatimuksia suurella tarkkuudella.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Signal Processing and Communications Engineering
Varjonen Antti: Home Automation System
Master of Science Thesis, 42 pages, 15 Appendix pages
November 2013
Major: Design of Electronic Circuits and Systems
Examiner: Professor Karri Palovuori
Keywords: modular, home, automation, Zigbee

Purpose of this thesis was to build a modular home automation system and learn more from Zigbee standard. Wireless communication bus was chosen since it might not be possible to build wired bus to old apartments.

System is made of a coordinator module and four peripheral modules. The coordinator handles message delivery between the peripheral modules. The peripheral modules are an infrared module, a relay module, a keyboard module and a temperature module. Purpose of the infrared module is to control household equipment that are normally controlled via a remote. For example a television or an audio amplifier. However in this project the module only provides a support for Sonys infrared protocol. The relay module controls a mains powered devices via connecting or disconnecting a phase conduit with relay and therefore cutting supply from the device. The keyboard module reads button status' and sends that information to the coordinator module. The temperature module measures ambient temperature and after detecting a change it sends a new temperature value to the coordinator module. After receiving new temperature value the coordinator prints these values to a terminal program running on a computer.

Two libraries were programmed for this system. First one being program interface with Zigbee module and second for Sony's infra-red protocol. Majority of the program inside modules is part of the Zigbee library. This library handles communication between modules. Zigbee module that has been chosen for this task provides an interrupt signal after successful data transaction. Infra-red library relies heavily on interrupts. Therefore it performs extremely well under the time constraints provided by the protocol.

Data transfer has been extremely reliable during testing period. During this time not a single failed data transfer was noticed. Temperature values from the temperature module are slightly different from commercial grade temperature sensor. However the module and commercial temperature sensor follows roughly the same pattern.

ALKUSANAT

Tämä on Tampereen teknillisen yliopiston diplomityö. Työ on tehty itsenäisesti omasta aiheesta.

Ohjaaja antoi neuvoja ja vinkkejä lähinnä diplomityön dokumentin rakenteeseen liittyen ja muihin yleisiin diplomityön suorittamiseen liittyviin asioihin. Kaverit auttoivat dokumentin oikoluvussa.

Kiitokset elektroniikan laitokselle, joka mahdollisti diplomityön teon tarjoamalla työtilat piirilevyjen valmistukseen sekä tuki komponenttihankinnoissa. Kiitokset ohjaajalle hyvistä neuvoista työn aikana. Lopuksi kiitokset Harri Manniselle, Niko Jokelalle ja Janne Virtaselle, jotka auttoivat dokumentin oikoluvussa.

Antti Varjonen
20.9.2013

SISÄLLYS

LYHENTEET

WPAN	Wireless Personal Area Networks
PHY	Fyysinen kerros (engl.: Physical Layer)
MAC	Medium Access Control
CCA	Clear Channel Assessment
MHR	MAC Header
ACK	Acknowledgement
IEEE	Institute of Electrical and Electronics Engineers
UWB	Ultra-Wideband
CRC	Cyclic Redundancy Check
FCS	Frame Check Sequency
LQI	Link Quality Indication
RSSI	Received Signal Strength Indicator
LDO	Low-Dropout
TVS	Transient Voltage Suppressor
ISP	In-System Programming
PDI	Programming and Debugging Interface
ESD	Electrostatic Discharge
EMI	Electromagnetic Iterference
USI	Universal Serial Interface

1. JOHDANTO

Työn tarkoituksena on tehdä modulaarinen kotiautomaatiojärjestelmä. Järjestelmä koostuu koordinaattorimoduulista ja neljästä apumoduulista. Koordinaattorimoduulin tärkein tehtävä on apumoduuleiden tietoliikenteen koordinointi. Apumoduulit toimivat järjestelmän rajapintoina ulkomaailmaan, eli toimivat input- ja output -moduuleina. Moduulit kommunikoivat keskenään Zigbee radiomoduuleiden avulla, jotka toteuttavat IEEE 802.15.4 standardin mukaista tiedonsiirtoa.

Input -moduuleina toimivat lämpötila-anturilla varustettu moduuli, sekä näppäimistömoduuli. Lämpötilamoduulin mittaamat lämpötila-arvot koordinaattorimoduuli lähettää testirajapinnan kautta PC:llä käytettävälle terminaaliohjelmalle. Näppäimistömoduulin viestit koordinaattori välittää järjestelmän output-moduuleille.

Output-moduuleina järjestelmässä toimivat infrapunamoduuli ja relemoduuli. Infrapunamoduulilla pyritään jäljittelemään normaaleiden kotitaloudesta löytyvien kaukosäätimien toiminnallisuutta. Relemoduulilla ohjataan verkkovirtaan kytkettyä laitetta vaihejohtimeen kytketyllä releellä.

Luvussa 2 käydään läpi järjestelmän rakenne yleisellä tasolla. Moduuleiden kommunikaatioon käytetty väylä on kuvattu luvussa 3. Luvussa 4 tarkastellaan moduuleiden rakennetta ja suunnitteluperusteita. Järjestelmän ohjelmisto ja kirjastot ovat kuvailtu luvussa 5. Testaus- ja mittaustulokset on koostettu lukuun 6. Lopuksi luvussa 7. on yhteenveto työstä.

2. RAKENTEELLINEN KUVAUS

Järjestelmän ytimessä toimii koordinaattorimoduuli, joka hoitaa tiedon välittämisen apumoduuleille. Lisäksi koordinaattorimoduuli tarjoaa rajapinnan, jota käytetään kehitysvaiheessa testaukseen. Koska tämän diplomityön puitteissa ei tehdä erillistä näyttöpäätteellistä moduulia, niin tätä samaista rajapintaa käytetään lämpötila-arvojen tarkasteluun.

Apumoduuleita järjestelmässä on neljä: lämpötila-, näppäimistö-, infrapuna- ja rele-moduuli. Lämpötilamoduuli koostuu yksinkertaisesta prosessorista, lämpötila-anturista ja Zigbee -moduulista. Tietyin väliajoin prosessori lähettää lämpötila-anturin arvon koordinaattorille. Jotta lämpötila-anturin sijoitus saadaan joustavaksi, niin lämpötilamoduuli on paristokäyttöinen.

Näppäimistömoduuli koostuu prosessorista, useasta nappulasta ja Zigbee -moduulista. Nappulan painallus tuottaa keskeytyksen. Keskeytyskäsittelijä tulkitsee mitä nappulaa on painettu ja välittää tiedon koordinaattorille. Tämän tiedon saatuaan koordinaattori välittää sen eteenpäin infrapuna- ja relemoduuleille, kuten kuvassa 2.1. Näppäimistömoduulin on tarkoitus olla kooltaan riittävän pieni kädessä pidettäväksi. Myös näppäimistömoduuli on paristokäyttöinen.

Kuva 2.1: *Järjestelmän lohkokaavio.*

Infrapunamoduuli tarjoaa rajapinnan kaupallisten tuotteiden, kuten audiovahvistimen ja television, hallintaan. Tässä diplomityössä infrapunamoduuli on toteutettu noudattamaan Sony:n infrapunaprotokollaa.

Relemoduuli on yksinkertaisesti etäohjattava kytkin. Moduuli katkoo tai yhdistää vaihejohdinta, kuten normaali kytkin. Ohjausinformaation relemoduuli saa tässä diplomityössä näppäimistömoduulilta.

3. VÄYLÄN KUVAUS

Järjestelmän käyttämäksi tiedonsiirtoväyläksi haluttiin langaton ratkaisu, sillä langallisen väylän rakentaminen asuntoon ei aina ole mahdollista ja osa apumoduuleista on paristokäyttöisiä ja helposti liikuteltavissa. Sähköverkkoa pystyisi käyttämään jossain määrin tiedonsiirtoon, mutta hylkäsimme tämän väylävaihtoehdon, koska käytössämme ei ole testiympäristön sähköverkon kytkentäkaavioita. Langattomia väyläratkaisuja tutkittaessa vastaan tuli kolme erilaista teknologiaa, jotka voisivat soveltua tähän projektiin: Bluetooth, WLAN ja Zigbee.

Bluetoothin suurimpana heikkoutena oli joustamattomuus väylätopologian suhteen, jossa laitteet kommunikoivat aina pareittain. Tämä pakottaisi väylän toimimaan pollaavasti ja vaatii jatkuvaa parien muodostamista. Lisäksi tämä aiheuttaa haasteita langattomien moduuleiden tehonkulutukselle, sillä näiden pitäisi odottaa, kunnes koordinaattori ottaa yhteyden. Lisäksi jatkuva parien muodostelu hidastaisi huomattavasti järjestelmän vasteaikoja.

WLAN puolestaan tukee luonnostaan väylätopologiaa, mutta on suunniteltu suuriin datamääriin ja tämä näkyy suurena virrankulutuksena. Toisaalta koska WLAN on hyvin yleinen kotitalouksien laajakaistamodeemeissa, tämä lähestymistapa tuottaisi suoraan rajapinnan tietokoneelle. Suurin ongelma WLAN -pohjaisessa väylässä on kuitenkin moduulien suuri hinta.

Zigbee -moduulit tarjoavat hyvin vapaan ympäristön toteuttaa väylää. Moduulit tarjoavat mahdollisuuden kommunikoida kaikille laitteille samanaikaisesti tai muodostaa linkkejä kahden laitteen välille. Kommunikointi laitteiden välillä voidaan toteuttaa vapaana kilpailuna tai aikajakoisena kanavointina. Zigbee -moduulit ovat suhteellisen virtapihejä ja heräävät nopeasti uni-tilasta. Hinnaltaan Zigbee -moduulit ovat samassa suuruusluokassa Bluetooth -moduuleiden kanssa.

Näistä valittiin Zigbee, sillä kyseisellä teknologialla pystyy helposti muodostamaan tarkoitukseen sopivan väylän. Pieni virrankulutus ja nopea uni-tilasta herääminen sopivat hyvin langattomille moduuleille.

3.1 IEEE 802.15.4

IEEE 802.14.4 -standardi on suunniteltu langattomien henkilökohtaisten verkkojen (WPAN, Wireless Personal Area Networks) kommunikointiin. Tavoitteena oli luoda standardi, joka mahdollistaisi halvan ja vähätehoisen kommunikointitavan verkon

laitteiden välillä. Nämä laitteet käyttävät lisenssivapaita radiokaistoja kommunikointiin: 868—868,6MHz; 902—928 MHz ja 2400—2483,5MHz. Näistä taajuuskaistoista 902—928 MHz taajuusalue ei ole lisenssivapaa taajuuskaista Euroopassa. [1]

Standardi kattaa fyysisen (PHY, Physical Layer) kerroksen ja MAC-kerroksen (Medium Access Control). Fyysinen kerros hoitaa kaiken varsinaiseen datan lähettämiseen liittyvän toiminnan. Näihin kuuluu muun muassa radiolähettimen aktivointi ja deaktivointi, kanavan valinta ja kanavan tilan arviointi (CCA, Clear Channel Assessment). Kanavan tilan arvioinnille standardi asettaa vaatimukseksi, että vähintään yksi kuudesta listatusta menetelmästä on toteutettu. Menetelmät on koostettu taulukkoon 3.1. Fyysinen kerros tukee myös radiolinkkien etäisyyden arviointia. [1]

Taulukko 3.1: *Taulukko kanavantilan arviointi metodeista.*

	Nimi	Selitys
1	Kynnysarvon ylittävä energia (Energy above threshold)	Kanava varattu, kun lähetyksen energiataso ylittää määrätyn energiatason.
2	Kantoaallon havainnointi (Carrier sense only)	Kanava tulkitaan varatuksi kun kantoaalto on havaittu.
3	Kantoaallon havainnointi ja kynnysarvon ylittävän eneriga	Energiatason tunnistus ja kantoaallon tunnistus.
4	ALOHA	Kanava ilmoitetaan aina vapaaksi.
5	UWB esilähetyksen havainnoinnilla.	Kanava merkataan varatuksi kun esilähetys havaitaan. Mikäli uutta esilähetystä ei havaita maksimi viestin pituuden suuruisen odotuksen jälkeen, niin kanava merkataan vapaaksi.
6	UWB esilähetyksen havainnointi multipleksatuista paketeista.	Kanava merkataan varatuksi kun standardin määrittelemä esilähetys havaitaan.

MAC-kerros puolestaan pitää sisällään lähetettävän datan rakenteen, viestintäkanavan valinnan ja merkinantosignaalin lähetyksen ja tähän tahdistumisen. Lisäksi tarjolla on turvamekanismi, jolla voidaan parantaa datan luotettavuutta ja autenttisuutta. Kuvassa 3.1 on kuva normaalista MAC-kehyksen rakenteesta. Otsikkokenttä MHR (MAC Header) pitää sisällään tiedot lähettäjästä, vastaanottajasta ja turvamekanismeista. MHR-kehyksen alussa on kahden tavun mittainen bittikenttä, joka määrittelee mitkä osiot MHR:sta löytyy. Siinä voidaan valita onko kehyksessä lähettäjän ja/tai vastaanottajan osoite ja käytetäänkö turvamekanismeja. Kolmantena tavuna tuleva järjestysnumero on apuväline, jota voidaan käyttää hyväksi ACK (Acknowledgement) viesteillä kuittaamiseen ja vaikkapa osoittamaan usean kehyksen lähetyksissä missä järjestyksessä data kuuluu käsitellä. FCS (Frame Check Sequency) kenttä pitää sisällään 16 bittisen CRC (Cyclic Redundancy Check) tarkistussumman. [1]

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	variable	2
Frame Control	Sequence Number	Destination PAN Identifier	Destination Address	Source PAN Identifier	Source Address	Auxiliary Security Header	Frame Payload	FCS
		Addressing fields						
MHR							MAC Payload	MFR

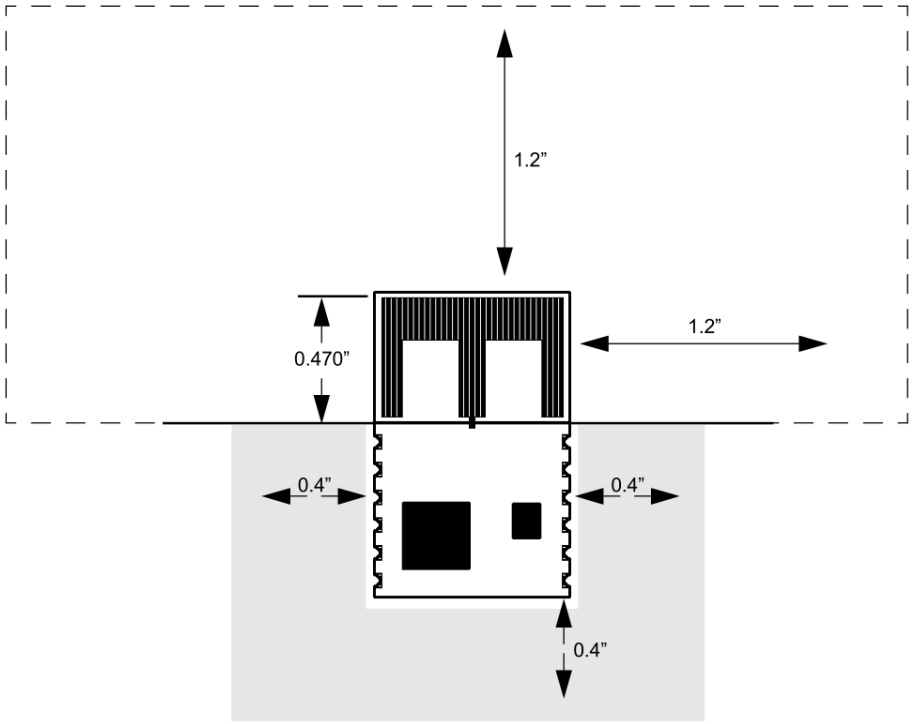
Kuva 3.1: *MAC -kehyksen rakenne.[1]*

3.2 MRF24J40

Projektin langatonta tiedonsiirtoa hoitamaan valittiin MRF24J40MA -moduuli, jonka ytimessä toimii MRF24J40 -mikropiiri. Moduulin sisältä löytyy oma sisäinen jännitteen regulointi kytkentä, sovitus piiri ja piirilevylle tehty antenni. Kuuluvuudelle datalehti lupaa jopa 120m etäisyyden. Moduuli käyttää hyväkseen 2,405—2,480GHz taajuuskaistaa. Tämä taajuuskaista on jaettu 16 kanavaksi, joista voidaan vapaasti valita mitä halutaan käyttää tiedonsiirtoon. MRF24J40MA -moduuli pystyy siirtämään tietoa 250kbps nopeudella. [2]

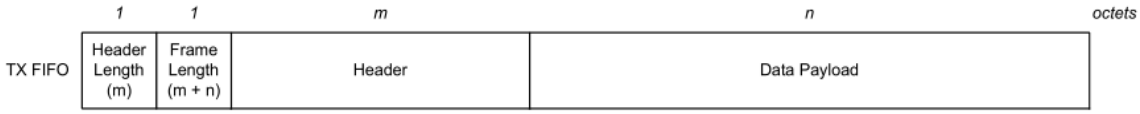
MRF24J40 -mikropiirin kanssa kommunikointiin käytetään SPI -väylää. Normaaleiden SPI -väylän vaatimien pinnien lisäksi mikropiiri tarjoaa wake, reset ja interrupt -pinnit. Wake -pinniä voidaan käyttää MRF24J40 -mikropiirin herättämiseen uni-tilasta. Koska uni-tilassa myös mikropiirin SPI -väylä on aktiivisena, niin herätys voidaan suorittaa myös mikropiirin REGWAKE -rekisterin arvoa muuttamalla. Reset -pinni asettaa mikropiirin alkuasetuksille. Tämän jälkeen on suotavaa odottaa 2ms, jotta mikropiirin radiopiirit käynnistyvät oikein [3]. Interrupt -pinnin tilan muutos ilmoittaa ulkopuoliselle laitteelle MRF24J40 -mikropiirin suorittaneen tiettyjä operaatioita tai suojausominaisuuksia käytettäessä pyytää isäntä kontrollerilta suojaukseen liittyvää lisätietoa. Operaatioita, joihin tämä pinni reagoi voidaan valita mikropiirin sisäisistä rekistereistä. Näistä operaatioista tärkeimpiä ovat lähetys- tai vastaanottotapahtuman suorittaminen valmiiksi.

Varsinaista laitetta suunnitellessa tulee ottaa huomioon, että MRF24J40MA -moduulissa on piirilevylle rakennettu antenni. Moduuli tarvitsee sijoittaa itse laitteen piirilevyn laidalle siten, että sen antenni osuus on piirilevyn ulkopuolella. Datalehti suosittelee antennin ja muiden mahdollisten metallien väliseksi etäisyydeksi vähintään 3cm parhaan suorituskyvyn takaamiseksi. Laitteen piirilevylle puolestaan datalehti suosittelee vähintään 1cm levyisen maa-alueen moduulin ympärille. Kuvassa 3.2 on havainnollistettu näitä etäisyyksiä paremmin. Harmaa alue kuvastaa suositeltua maa-aluetta. Katkoviivalla piirretty suorakulmio puolestaan kuvaa aluetta, jossa ei saisi olla ylimääräistä metallia.



Kuva 3.2: *MRF24J40MA -moduulin suositeltu sijoittelu. [2]*

Moduulin lähetyskehys noudattaa kuvan 3.3 rakennetta. Header ja Data Payload osiot ovat käytännössä IEEE 802.15.4 standardin MAC -kehys. Ainoana erona on FCS -kentän puuttuminen, sillä MRF24J40 laskee tarkistussumman automaattisesti ja lisää sen kehyksen loppuun. Kehyksen *header length* ja *frame length* -kentät ovat lisätietoja, joita mikropiiri tarvitsee avuksi MAC -kehyksen luonnissa.



Kuva 3.3: *MRF24J40 -piirin vaatiman lähetyskehyksen rakenne. [3]*

Lähetyksen vastaanoton jälkeen MRF24J40 -piiri muokkaa vastaanottamansa datan kuvan 3.4 mukaiseen muotoon. Kontrolleri lisää vastaanottopakettiin LQI (Link Quality Indication) ja RSSI (Received Signal Strength Indicator) kentät, joiden avulla pystytään arvioimaan vastaanotetun signaalin laatua. RSSI arvioi signaalin tehoa. Tämän arvoasteikko on 0 (-100dBm)—255 (-20dBm). LQI -kenttä ilmoittaa radiolinkin laadun arvoasteikolla 0—255, jossa 0 on erittäin huono ja 255 vastaavasti erittäin hyvä. RSSI -kenttä on vapaaehtoinen ja se voidaan ottaa pois käytöstä moduulin asetuksista.

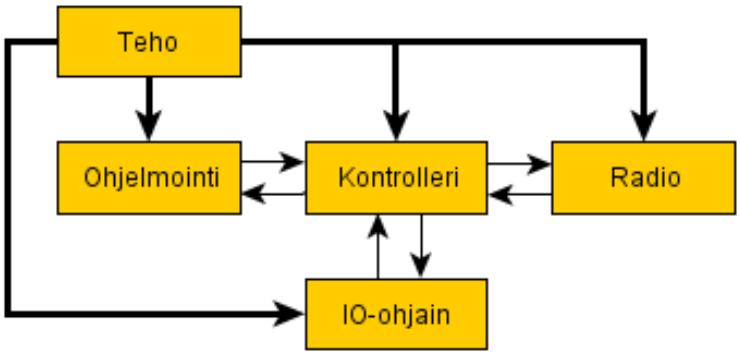
MRF24J40 tarjoaa mahdollisuuden automaattiselle lähetyksen kuittaukselle ACK -viestillä. Tämä on hyödyllinen ominaisuus, koska se vähentää ohjelmointityötä ja

Kuva 3.4: *MRF24J40 -piirin vaatiman vastaanottokehyksen rakenne. [3]*

ACK -viestin viive saadaan pysymään pienenä. MRF24J40MA -moduuleiden maksimi ACK -viestin odotusviive on noin 2ms. Tämän odotuksen jälkeen moduuli lähettää viestin uudelleen.

4. MODUULIEN KUVAUS

Moduulit koostuvat viidestä lohkosta; Teholähde-, radio-, mikrokontrolleri-, IO- ja ohjelmointilohkosta. Kuva 4.1 kuvaa moduulin lohkojen välisiä suhteita. Radio- ja ohjelmointilohko ovat kaikissa moduuleissa lähes samanlaiset. Kontrollerilohkoissa on hyvin paljon vaihtelua moduuleiden erilaisista tarpeista johtuvista syistä. Esimerkiksi koordinaattorimoduuli tarvitsee lähinnä muistia, kun taas näppäimistömoduuli tarvitsee enemmän IO -pinnejä.



Kuva 4.1: *Lohkokaavio moduuleiden rakenteesta.*

Teholähdelohko tuottaa moduuleiden logiikan tarvitsemat jännitetasot. Yksinkertaisimmillaan tämä tarkoittaa paristoja. Mutta sisääntulojännite voi olla 230 V_{ac} tai 5—20V_{dc}, jolloin tämä jännite reguloidaan logiikan käyttämäksi jännitetasoksi. Tyypillinen logiikan käyttämä jännitetaso on 3—3,3V_{dc}. Taulukkoon 4.1 on koostettu moduuleiden teholohkojen sisääntulojännitteet ja ohjattavan logiikan jännitteet.

Taulukko 4.1: *Moduuleiden teholähteiden sisääntulo- ja ulostulojännitteet.*

Moduulin nimi	Sisääntulojännite	Logiikanjännite
Koordinaattori	12Vdc	3,3Vdc
Näppäimistö	2xAAA Paristo	2,6—3,0Vdc
Lämpömittari	2xAAA Paristo	2,6—3,0Vdc
Rele	230Vac	3,3Vdc
IR	12Vdc	3,3Vdc

IO-ohjain on moduulin ainoa keino vaikuttaa ympäristöön ja saada aikaan muutoksia ympäristössä tai pistää ne aluilleen. Suurin osa moduuleista toimii pelkästään

input tai output -laitteena, mutta koordinaattorimoduuli tarjoaa rajapinnan molempiin suuntiin. Taulukossa 4.2 on listattu mitkä laitteet löytyvät minkäkin moduulin IO-ohjaimesta.

Taulukko 4.2: *Moduuleiden IO-ohjainten listaus.*

Moduulin nimi	IO-laite	Laitteen tyyppi
Koordinaattori	USB ja USART	Input ja Output
Näppäimistö	13 nappulaa	Input
Lämpömittari	Lämpötila-anturi	Input
Rele	Rele	Output
IR	IR-led	Output

4.1 Kontrollerilohko

Koordinaattorimoduuli

Tähän moduuliin on valittu kontrolleriksi ATxmega128B3. Tärkein valintaperuste kontrollerille oli vähintään neljän kilotavun SRAM, sillä arvion mukaan se riittää hyvin usean moduulin järjestelmään. Lisäksi kontrollerilta vaadittiin SPI -rajapinta ja mahdollisuus toteuttaa rajapinta tietokoneelle testaustarkoitukseen. Xmegassa on yksi SPI-, yksi USART- ja yksi USB-lohko. SPI-lohko tarvitaan radiomoduulin kanssa kommunikointiin. USB- ja USART-lohkoja käytetään moduulin testauksessa.

Kontrollerin kellolähteenä käytetään sisäistä kalibroitua 32MHz oskillaattoria. Tästä kellolähteestä on helppo luoda ylinäytteistämällä USB -lohkon tarvitsema 48MHz taajuus. Lisävaatimuksena kellolle on USB:n vaatima 0.25%:n kellotaajuuden tarkkuus. [4]

Näppäimistömoduuli

Kontrollerilta vaaditaan vähintään 14 IO-pinniä ja SPI-rajapinta. IO-pinneistä neljä tarvitaan radiomoduulille ja loput nappuloille. Toiminnallisuudeltaan näppäimistömoduuli on suhteellisen yksinkertainen, joten muistin suuruus ei ole kovin olennaista. Työhön päädyttiin valitsemaan ATmega88A kontrolleri, jossa on 23 IO-pinniä. Kaikilla näillä pinneillä on myös muuta toiminnallisuutta, kuten esimerkiksi SPI tai RESET. Täten lopullinen nappuloille käytettävien IO-pinnien määrä jää 13 kappaleeseen.

ATmega88A kontrollerissa on kahdeksan kilotavun ohjelmamuisti ja yhden kilotavun SRAM muisti. Kontrolleri tukee myös porttikohtaisia keskeytyksiä. Tämä on erittäin hyödyllinen ominaisuus näppäimistömoduulissa, sillä näin jokaisen nappulan tila pystytään lukemaan keskeytyspalvelussa. Lisäksi porttikeskeytykset pystyvät

herättämään kontrollerin uni-tilasta. Nämä kaksi ominaisuutta parantavat näppäimistömoduulin vasteaikaa ja pienentävät tehokulutusta.

Kellolähteenä käytetään kontrollerin sisäistä 8MHz kelloa. Sisäiseen kelloon päädyttiin siitä syystä, että näin saadaan vapautettua lisää IO-pinnejä ja lisäksi sisäisen kellon tarkkuus ja nopeus riittävät hyvin tähän moduuliin.

Rele-, IR- ja Lämpömittarimoduuli

Näiden moduuleiden kontrollerin vaatimuksena oli vähintään kahdeksan IO-pinniä. Näistä pinneistä seitsemän tarvitaan radiomoduulin ohjaukseen ja viimeinen jää IO-laitteen ohjaukseen.

Vaikka näiden moduuleiden tarpeet eroavat hieman toisistaan, niin valitsemamme ATtiny44V kontrolleri kykenee helposti täyttämään nämä tarpeet. Eniten vaatimuksia kontrollerille aiheuttaa IR-moduuli, jossa prosessorin tarvitsee luoda noin 40kHz taajuudella moduloitu signaali ohjaamaan infrapunalähetintä.

ATtiny44V kontrollerissa on neljän kilotavun ohjelmamuisti ja 256 tavun SRAM muisti. Kontrollerista löytyy 8- ja 16-bittiset laskurit, joilla pystytään tuottamaan aikakriittistä signaalia tarpeen vaatiessa. Moduulit käyttävät kontrolleria 8MHz kellotaajuudella.

4.2 Teholähdelohko

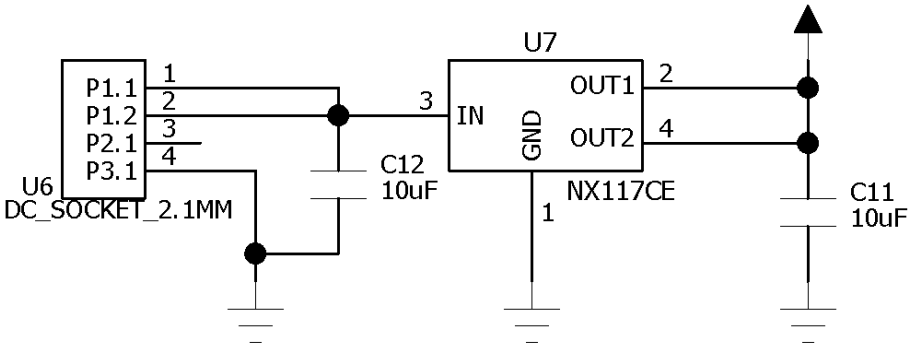
Koordinaattori- ja IR-moduuli

Teholähdelohko tuottaa koordinaattorimoduulin tarvitseman 3,3V jännitteen. Tämä tuotetaan NX117CE-sarjan LDO-regulaattorilla, jonka sisääntulojännitteeksi kelpaa korkeintaan 20V jännite. Regulaattori on kytketty kuten kuvassa 4.2, eli se tuottaa kiinteän 3,3V:n jänniteen. Suodatuskondensaattorit C11 ja C12 on valittu regulaattorin datalehden suositusten mukaisesti 10uF kondensaattoreiksi [5].

Lohkon käyttöjännitteet tulevat ulkoisesta muuntajasta, joka kytketään piirilevylle 2,1mm tasajänniteliittimellä. Muuntajina käytetään käytöstä poistuneiden laitteiden muuntajia, joiden nimellisjännite on 12V.

Näppäimistö- ja lämpömittarimoduuli

Moduulit saavat käyttöjännitteensä suoraan kahdesta sarjaan kytketystä AAA-luokan paristosta. Kaksi paristoa vaaditaan, koska radiomoduuli vaatii 2,6V käyttöjännitteet toimiakseen [3]. Paristoille ei ole tehty suojauskytkentää, sillä tarvittavien komponenttien yli jäävä jännitehäviö on turhan suuri. Arvion mukaan suojauksen toteuttamiseen vaadittavat komponentit vaativat noin 250—350mV jännitteen, joka aiheuttaisi noin 50% pariston käyttöiän vähenemisen [6].



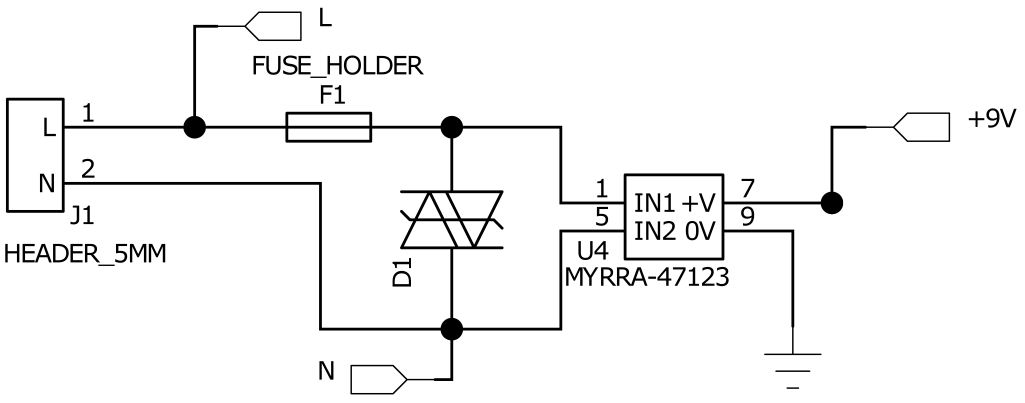
Kuva 4.2: *Teholähteen kytkentäkaavio.*

Rele

Relemoduuli on suunniteltu toimimaan suoraan verkkovirralla. Teholähdelohkosta löytyy ensin AC-DC -muunnin, jolla verkkovirta muutetaan 9V tasajännitteeksi. Tämän jälkeen tulee DC-DC -muunnin, jolla pudotetaan 9V jännite 3,3V:ksi.

Käytetty AC-DC -muunnin on Myrran valmistama valmiskomponentti. Tämä pitää sisällään tasasuuntaussillan, muuntajan ja suojauspiirejä. Myrran muuntajalle kelpaa syöttöjännitteeksi vähintään 85V tasa- tai vaihtojännitettä. Maksimijännite puolestaan on 265V vaihtojännitettä ja 370V tasajännitettä. Vaihtojännitteen taajuuden tulee olla 47 ja 440 Hz välillä. Hyötysuhteeksi muuntajalle on ilmoitettu 70%. Muuntajan maksimi syöttövirta on valitulla mallilla 270mA. [7] Logiikan käyttämä 3,3V jännite tuotetaan NX117CE -sarjan LDO-regulaattorilla.

Moduuliin on lisätty kuvan 4.3 mukaisesti suojauskytkentä sulakkeen ja kaksisuuntaisen TVS:n avulla. TVS:n tarkoituksena on suojata logiikkaa ylijännitettä vastaan ja sulakkeella suojataan liian suurilta virroilta.



Kuva 4.3: *AC-DC -muuntimen suojauskytkentä.*

4.3 Radiomoduuli

Radiomoduulin datalinja on kytketty suoraan kontrollerin MISO-, MOSI- ja SCK-pinneihin. Loput radiomoduulin IO-pinneistä on kytketty kontrollerin IO-pinneihin taulukon 4.3 mukaisesti. Lohkon käyttöjännitteisiin on lisätty ylimääräinen suodatuskondensaattori.

Lisäksi radiomoduulin CS-pinniin on lisätty ylösvetovastus. Tällä pyritään varmistamaan ettei radiomoduulin datalinjat ole aktiivisena ohjelmoinnin aikana.

Taulukko 4.3: *Radiomoduulin ohjauspinnien kytkentä.*

Moduulin pinni	Koordinaattori	Näppäimistö	IR	Lämpötila	Rele
Chip Select (CS)	PortC 4	PortC 0	PortA 3	PortA 3	PortA 3
Interrupt (INT)	PortB 6	PortB 2	PortA 2	PortA 2	PortA 2
WAKE	PortB 5	PortB 1	PortA 1	PortA 1	PortA 1
RESET	PortB 4	PortB 0	PortA 0	PortA 0	PortA 0

4.4 Ohjelmointilohko

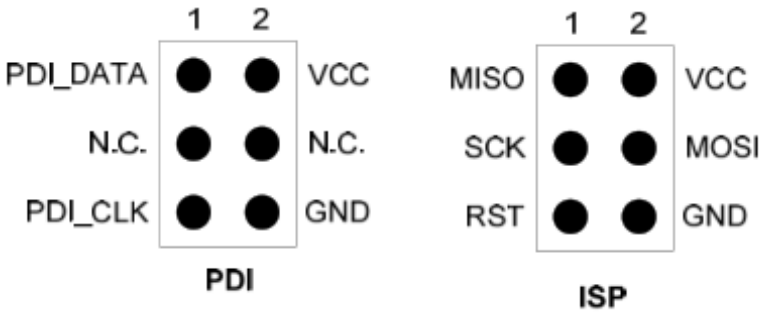
Käytössämme on Atmel:n AVR ISPmkII -ohjelmointilaite, mutta Xmega-sarjan kontrollerit eivät tue ISP (In-System Programming) -rajapintaa. Nämä laitteet käyttävät uudempaa PDI (Programming and Debugging Interface) -ohjelmointirajapintaa, joka on yhteensopiva käyttämämme ohjelmointilaitteen kanssa. [8] Muut moduulit voidaan ohjelmoida ISP -rajapinnan kautta.

ISP -ohjelmointirajapintaa käytettäessä ohjelmointiliitin kytketään kuvan 4.4 mukaisesti kontrollerin SPI-lohkoon. Lisäksi ohjelmointiliittimen ja kontrollerin RESET-pinnit pitää liittää yhteen. Ohjelmointilaite mittaa kontrollerin käyttöjännitetasoa ohjelmointiliittimen VCC-pinnin avulla. Näin voidaan varmistua, että kontrollerilla on riittävän suuri käyttöjännite ohjelmointia varten.

Koordinaattorimoduulin käyttämässä PDI -ohjelmointirajapinnassa PDI_DATA -pinni kytketään PDI-pinniin ja PDI_CLK kytketään RESET-pinniin. Koska PDI -rajapinnan kellosignaali kulkee kontrollerin RESET -linjassa, niin tähän linjaan ei saa kytkeä kondensaattoreita. PDI- ja SPI-ohjelmointiliittimen pinnijärjestykset on havainnollistettu kuvassa 4.4.

4.5 Testauslohko

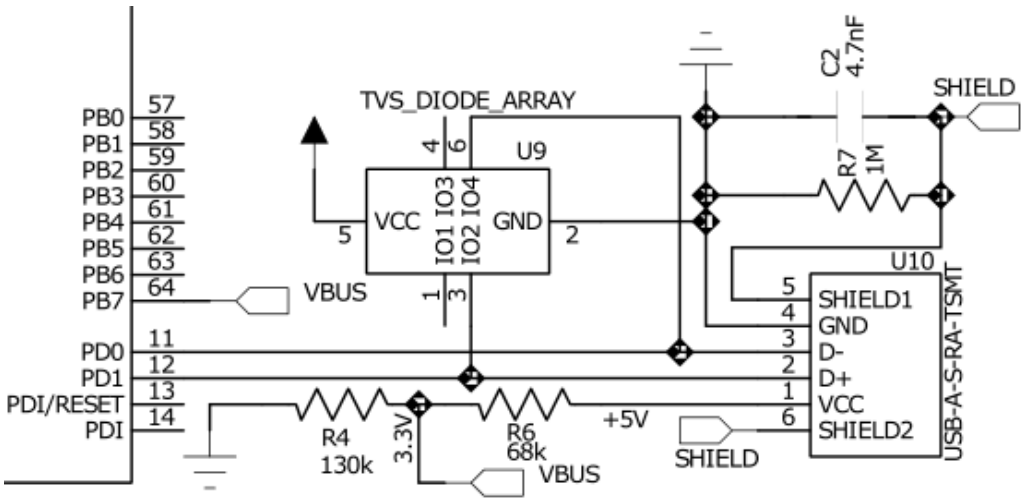
Testauslohkossa käytetään hyväksi mikrokontrollerin sisäistä USB-lohkoa. Kontrollerin sisäinen rakenne sisältää kaikki tarpeelliset komponentit, mutta testauslohkoon on lisätty ylimääräistä suojausta sähkömagneettisia häiriöitä vastaan. R7 ja C2 erottavat USB-kaapelin suojavaipan ja kontrollerin maatason, jotta suojavaippa ei toimisi antennina [4]. TVS-diodi on lisätty suojaamaan moduulia USB:stä aiheutuneilta



Kuva 4.4: *PDI- ja ISP-ohjelmointiliittimien pinnijärjestys.*

ylijännitepulsseilta.

USB:n käyttöjännite 5V on tiputettu vastusjaolla 3,3V:iin, jotta kontrollerilla pystytään havaitsemaan onko USB-liitin kytketty vai ei. Tämä lisäys tehtiin siksi, että USB:n datalinjat eivät saa olla aktiivisia, kun liitintä ei olla kytketty toiseen laitteeseen. [4; 9]



Kuva 4.5: *Testauslohkon kytkentäkaavio.*

Lisäksi USAT-lohkon lähetys- ja vastaanottopinnit ovat kytketty piikkirimaan, jotta tarpeen vaatiessa voidaan käyttää huomattavasti yksinkertaisempaa testausrajapintaa. Koordinaattorimoduuliin ei lisätty RS232-USB -muunninpiiriä, sillä käytössämme on kehitysalusta, joka mahdollistaa kyseisen toiminnallisuuden. Kokonaisuudessaan koordinaattorimoduulin kytkentäkaavio löytyy liitteestä 1.

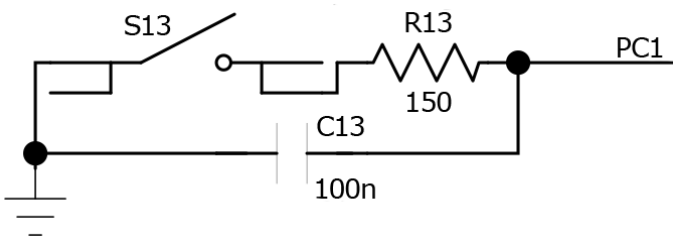
4.6 Näppäimistölohko

Näppäimistölohko koostuu 13:sta erillisestä näppäimestä, jotka ovat kytketty suoraan kontrollerin IO-pinneihin. Vaihtoehtoinen kytkentätapa olisi ollut matriisito-

teutus. Tämä vaihtoehto kuitenkin hylättiin turhan kompleksisuuden vuoksi. Lisäksi usean näppäimen yhtäaikainen painallus on huomattavasti vaikeampaa havaita.

Nappuloille on lisätty RC -kytkentä suojaukseksi. Yksittäisen nappulan kytkentä on esitetty kuvassa 4.6. Kondensaattori C13 suodattaa pois kytkinvärähtelyn ja vastus R13 on lisätty suojaamaan kytkintä virroilta.

Vastus on mitoitettu siten, ettei kytkimen kestämä 50mA virta ylittyisi kondensaattorin purkautuessa [10]. Käyttämällä 150Ω vastusta kytkimen läpi kulkeva virta voi olla korkeintaan 20mA. Käyttöjännitteen nappulat saavat kontrollerin sisäisiä ylösvetovastuksia käyttäen, jotka ovat arvoiltaan noin 22kΩ luokkaa [11]. Näppäimistömoduulin lopullinen kytkentäkaavio löytyy liitteestä 4.



Kuva 4.6: *Yksittäisen näppäimen kytkentä.*

4.7 Lämpömittarilohko

Lämmön mittaus on toteutettu TC72 -lämpötila-anturilla. Piirillä pystyy mittaamaan -50°C — +125°C lämpötiloja. Lämpötilojen muunnostarkkuus on 10-bittiä, eli 0,25°C/bitti. TC72 -piirille luvataan ±2°C tarkkuus. Piiri kuluttaa 250μA virran tehdessään lämpötilamuunnosta. Kun piiri lopettaa tai käsketään lopettamaan lämpötilamuunnosten tekeminen, niin virran kulutus putoaa alle 1μA. Lämpötilamuunnoksen tekeminen kestää tyypillisesti noin 150ms. Arvojen lukeminen piiriltä tapahtuu SPI-väylän avulla. [12]

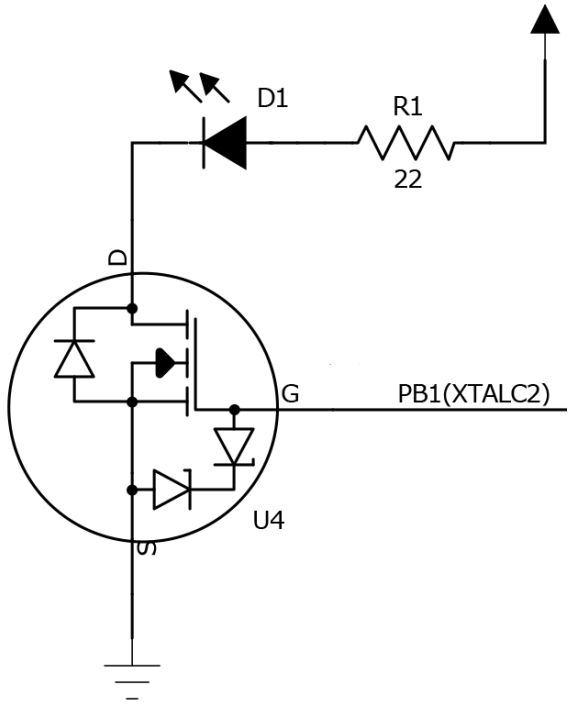
Koska lämpömittarimoduulin ohjelmointi käyttää SPI-väylää, niin TC72-piirin CE-pinniin on kytketty alasvetovastus. Tällä pyritään estämään ettei TC72 ole aktiivisena ohjelmoinnin aikana. Lämpömittarimoduulin kytkentäkaavio löytyy kokonaisuudessaan liitteestä 5.

4.8 Infrapunalohko

Infrapunalohko koostuu IR-LED:stä ja MOSFET:stä. Infrapunalähettimeksi valittiin Vishay:n TSAL4400 IR-LED. Tämä IR-LED tuottaa valoa 940 nm aallonpituudella, joka on samaa suuruusluokkaa kuin ohjattavan laitteen alkuperäisellä kaukosäätimellä. Valitsemamme infrapunalähetin kestää 100mA jatkuvan myötävirran ja tarvittaessa jopa 1.5A suuruisia lyhyitä purskevirtoja. [13]

Moduulin käyttämä IR-LED on kytketty MOSFET:n ja vastuksen kautta maahan (kuva 4.7), koska kontrolleri ei pysty ajamaan riittävän suuria virtoja IR-LED:n läpi. IR-LED:lle on lisätty 22Ω vastus rajoittamaan maksimivirtaa. Infrapunamoduulin kytkentäkaavio löytyy liitteestä 2.

MOSFET:ä pyritään käyttämään kytkimen tavoin, eli joko avataan täysin tai pidetään kokonaan kiinni. Transistorin ohjaus tulee suoraan kontrollerin IO-pinnistä.



Kuva 4.7: *IR-LED:n kytkentä.*

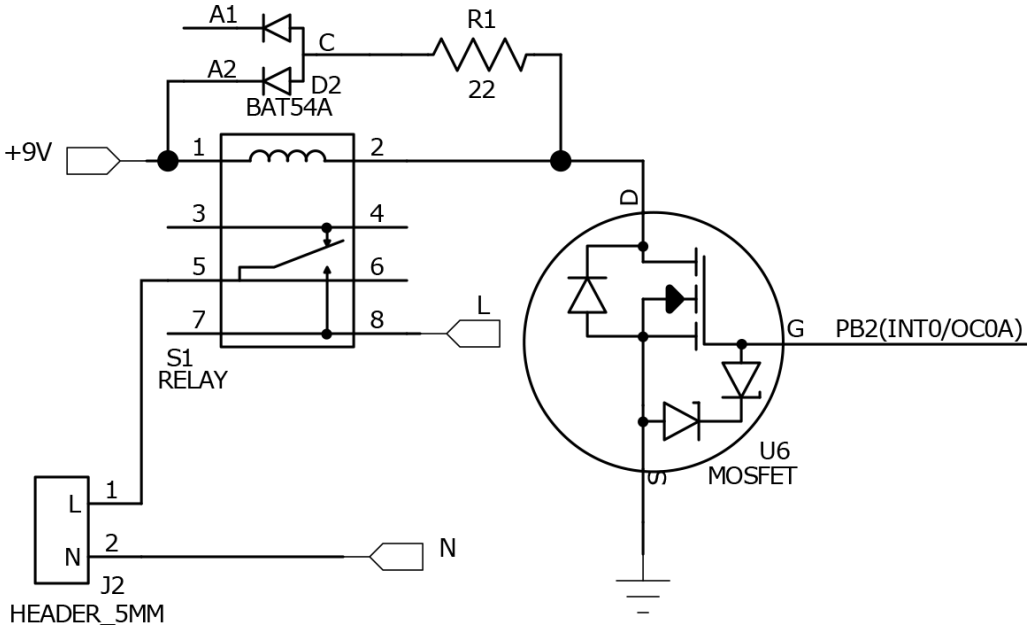
4.9 Relelohko

Rele on valittu käyttämään $9V_{dc}$ jännitettä. Koska kontrolleri ei pysty syöttämään riittävästi virtaa ohjatakseen $3V_{dc}$ relettä, niin joudumme käyttämään transistoria releen ohjaukseen. Relemoduulilta löytyy $9V_{dc}$ ja $3{,}3V_{dc}$ jännitetasot, joten käyttöön voidaan valita halvempi $9V_{dc}$ jännitetasoa hyödyntävä rele. Lisäksi nämä pystyvät kytkemään suurempia virtoja.

Releeksi valittiin multicomp:n HRM2-S DC9V rele. Tämän käämin resistanssi on noin 112,5Ω. Toimiakseen rele vaatii väihtään 6,75V jännitteen. Releen kontakti puoli kestää $250V_{dc}$ jännitteen ja jopa 16A virran. [14] Eli rele soveltuu jopa isojen kuormien ohjaamiseen. Tässä työssä releellä on tarkoitus ohjata tavallista 60W hehkulamppua.

Rele on kytketty siten, että ilman ohjausta vaihejohdin muodostaa avoimen piirin. Tällä pyritään varmistamaan, että jos jostain syystä relemoduulin kontrollerin

käyttöjännitteet katkeavat, niin kuormana oleva laite ei ole aktiivisena. Releen ohjauksen apuna kontrolleri käyttää MOSFET transistoria kytkimen tavoin. Releelle on lisätty kuvan 4.8 mukaisesti suojadiodi, jolla mahdollistetaan releen käämin magneettikentän hallittu purkautuminen. Vastus on lisätty rajoittamaan suojadiodin läpi kulkevaa virtaa. Koko moduulin kytkentäkaavio on liitteessä 3.



Kuva 4.8: *Releen kytkentä.*

5. OHJELMISTO

5.1 MRF24J40 -kirjasto

Kirjasto ohjelmoitiin hoitamaan radiomoduuleiden välistä kommunikaatiota ja alustamaan radiomoduuli. Lisäksi kirjasto hoitaa tarvittaessa moduulin asettamisen unitilaan ja siitä heräämisen. Apuna tässä työssä kirjasto käyttää makroja, jotka hoitavat IO-pinnien tilanvaihdokset. Makroilla on myös määritetty mihin kontrollerin pinneihin radiomoduulin WAKE-, RESET-, CS- ja INT-pinnit ovat kytketty. Esimerkiksi näppäimistö moduulin käyttämät IO-pinnien tilanmuutos makrot ovat seuraavanlaiset:

Listaus 5.1: *Näppäimistömoduulin makrot radiomoduulin ohjaukselle.*

```
/* Cs macros */
#define MRF_CS_SET() MRF_CS_port &= ~MRF_CS_bm
#define MRF_CS_CLR() MRF_CS_port |= MRF_CS_bm
/* Wake macros */
#define MRF_WAKE_SET() MRF_WAKE_port |= MRF_WAKE_bm
#define MRF_WAKE_CLR() MRF_WAKE_port &= ~MRF_WAKE_bm
/* Reset macros */
#define MRF_RESET_SET() MRF_RESET_port &= ~
    MRF_RESET_bm
#define MRF_RESET_CLR() MRF_RESET_port |=
    MRF_RESET_bm
```

Tässä esimerkissä WAKE ja RESET makrojen portit ovat PORTB ja CS-pinnin portti on PORTC. Lisäksi CS- ja RESET-pinnit ovat alhaalla aktiivisia. Makroissa on otettu tämä huomioon käyttämällä JA -operaatiota, jolloin SET-makrossa bittimaskeja *MRF_CS_bm* ja *MRF_RESET_bm* vastaavat bitit nollataan.

Kirjasto käyttää kolmea funktiokutsua toteuttaakseen SPI -muotoisen kommunikaation kontrollerin ja radiomoduulin välillä. Nämä funktiokutsut ovat listauksessa 5.2. Xmega:lle ja ATmega:lle ne on toteutettu kirjastossa itsessään. Kyseiset toteutukset käyttävät kontrollerin omaa SPI -lohkoa.

Listaus 5.2: *MRF24J40 -kirjaston tarvitsemat funktiot.*

```
void spi_send_byte( uint8_t byte );
uint8_t spi_receive_byte( void );
void spi_config( void );
```

ATtiny -kontrollereiden tapauksessa tapahtui suunnitteluvirhe, joten kontrollerin USI (Universal Serial Interface) -lohkoa ei voitu käyttää näiden funktioiden toteutukseen. Tästä syystä SPI -toiminnallisuus ohjelmoitiin assemblyllä käyttäen normaaleja IO-pinnejä.

5.1.1 Tietueet

Tietueita kirjastosta löytyy kolme. Yksi jokaiseen viestiin liitettävälle otsikolle ja kaksi tietuetta lähetettäville viesteille. Otsikko noudattaa IEEE 802.15 standardin vaatimuksia, kuten luvussa 3.1 on kuvailtu. Koska kirjasto käyttää vain dataviestejä ja salaamatonta dataa, niin tietue on voitu yksinkertaistaa listauksen 5.3 mukaiseksi. Antamalle tietueelle attribuutiksi **packed** varmistetaan, että kääntäjä sijoittaa tietueen muuttujat peräkkäisiin osoitteisiin kontrollerin osoiteavaruuteen. Käytettäessä tämän kaltaisia tietueita tulee kiinnittää erityistä huomiota miten data tallennetaan kontrollerin muistiin. Erityisesti 16-bittisten sanojen tavujärjestys saattaa aiheuttaa ongelmia, sillä SPI-lohko pystyy käsittelemään vain kahdeksaa bittiä kerrallaan.

Listaus 5.3: *MRF24J40 moduulin otsikon rakenne.*

```
typedef struct MRF_Header {
    uint16_t frameControl;
    uint8_t sequenceNumber;
    uint16_t destPanID;
    uint16_t destAddr;
    uint16_t sourceAddr;
} __attribute__ ((packed)) MRF_Header_t;
```

Datanlähetykselle käytetty tietue noudattaa radiomoduulin tarvitseman datan rakennetta. IEEE 802.15 standardin vaatima tarkistussumma puuttuu tästä tietueesta, sillä MRF24J40MA -moduuli suorittaa tarkistussumman laskemisen ja lisää sen automaattisesti lähetettävään pakettiin. Tietueen *data*-taulukon koko on määritelty makroilla mrf.h tiedostossa. Tässä työssä sen kooksi on asetettu kahdeksan alkiota. Tietue on listauksessa 5.4.

Listaus 5.4: *Kirjaston käyttämän lähetyspaketin rakenne.*

```
typedef struct MRF_Tx_packet {
    uint8_t headerLength;
    uint8_t frameLength;
    MRF_Header_t header;
    uint8_t data[MRF_DATA_PAYLOAD_SIZE];
} __attribute__ ((packed)) MRF_Tx_packet_t;
```

Kolmas tietue on datan vastaanotolle. Tämä rakenne noudattaa radiomoduulin datalehden vaatimuksia. Koska tässä työssä *data* -taulukon pituus on vakio ja *header* -tietue on myös kooltaan vakio, niin vastaanotetut bitit voidaan tallentaa suoraan

tähän tietueeseen. Tästä syystä vastaanottavan laitteen datan käsittely helpottuu huomattavasti ja voidaan käsitellä suoraan esimerkiksi vastaanotetun *data* -taulukon sisältöä ilman, että tarvitsee laskea missä kohtaa viestiä kyseinen data löytyy. Tietueen tarkempi kuvaus on listauksessa 5.5.

Listaus 5.5: *Kirjaston käyttämän vastaanottopaketin rakenne.*

```
typedef struct MRF_Rx_packet {
    /* Headerl length + data length + 2 */
    uint8_t frameLength;
    MRF_Header_t header;
    uint8_t data[MRF_DATA_PAYLOAD_SIZE];
    uint16_t FCS;
    uint8_t LQI;
    uint8_t RSSI;
} __attribute__ ((packed)) MRF_Rx_packet_t;
```

5.1.2 Funktiot

void mrf_config(void)

Funktio tekee kontrollerin tarvitsemat alustukset kontrollerin ja MRF24J40 -moduulin välille. Tämän jälkeen aloitetaan moduulin alustus. MRF24J40 -moduulin alustuksessa käytetään hyväksi laitekohtaisesta config.h tiedostosta löytyvää osoitetta ja mrf.h tiedoston alusta löytyviä tietoja. Nämä tiedot pitävät sisällään mihin kontrollerin pinniin MRF24J40 -moduulin IO -pinnit ovat kytketty ja mitä kanavaa kommunikointiin käytetään.

Tämä funktio käyttää apunaan *spi_ config()* -funktiota, joka on XMEGA ja ATMEGA kontrollereiden tapauksessa sisällytetty kirjastoon. ATTINY kontrollerit käyttävät omaa kirjastoa SPI muotoisen kommunikaation toteuttamiseen, joten näiden kontrollerien tapauksessa tuo funktiokutsu käyttää toisen kirjaston funktioita.

Tässä työssä MRF24J40MA -moduuli alustetaan seuraavanlaisin asetuksin:

Radiomoduulin kanava:	16
PanID:	0x0001
Lähetysteho:	0dBm
uni-tilan kellolähde:	100kHz (sisäinen)
Kanavan tilanarviointi:	Vain energiatason mittaus
Keskeytys tapahtumat:	Vastaanotto ja lähetys valmis

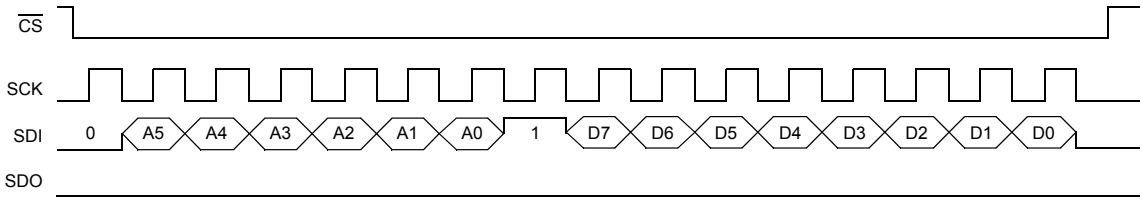
Näppäimistö ja lämpömittarimoduuleiden tapauksessa on lisäksi sallittu WAKE -pinnin toiminta ja MRF24J40MA -moduulin uni-tila. Muut asetukset on pidetty

tehdasasetuksilla. Lopuksi funktio suorittaa RF-tilakoneen nollauksen, jotta kanavan valinta ja lähetysteho asetukset tulisivat voimaan. Jos uni-tila on sallittu niin MRF24J40MA -moduuli asetetaan uni-tilaan ennen funktiosta poistumista.

void mrf_short_address_write(uint8_t address, uint8_t data)

Funktio toteuttaa MRF24J40 -moduulin lyhyeen kuusibittiseen muistiavaruuteen kohdistuvat kirjoitusoperaatiot. Apuna tässä työssä funktio käyttää *spi_send_byte()* -funktiota. Parametri *address* pitää sisällään mihin osoitteeseen kirjoitusoperaatio kohdistuu ja *data* pitää sisällään tähän osoitteeseen kirjoitettavan tiedon.

Funktio manipuloi osoitteen ja datan noudattamaan kuvan 5.1 mukaista muotoa. Tässä rakenteessa ensimmäinen lähetetty bitti kertoo onko kyseessä lyhyeen vai pitkään osoiteavaruuteen kohdistuva operaatio ja kahdeksas bitti kertoo onko kyseessä luku- vai kirjoitusoperaatio. [3]

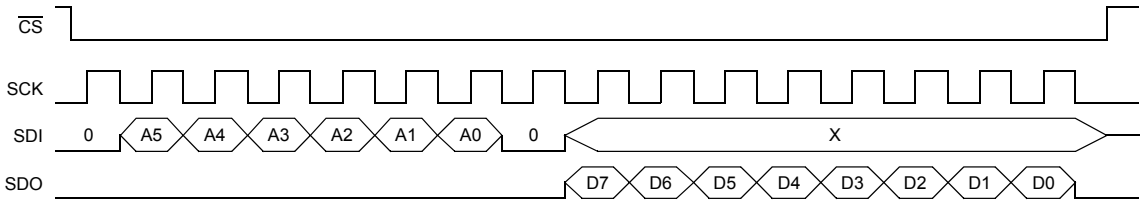


Kuva 5.1: *Lyhyen osoiteavaruuden kirjoituksen ajoituskaavio. [3]*

uint8_t mrf_short_address_read(uint8_t address)

Funktio toteuttaa MRF24J40 -moduulin lyhyeen kuusibittiseen muistiavaruuteen kohdistuvat lukuoperaatiot. Apuna tässä työssä funktio käyttää *spi_send_byte()* ja *spi_receive_byte()* -funktioita. Parametri *address* pitää sisällään mihin osoitteeseen lukuoperaatio kohdistuu. Paluuarvona on *address*:n osoittaman muistipaikan tieto.

Funktio manipuloi osoitteen ja datan noudattamaan kuvan 5.2 mukaista muotoa. Tässä rakenteessa ensimmäinen lähetetty bitti kertoo onko kyseessä lyhyeen vai pitkään osoiteavaruuteen kohdistuva operaatio ja kahdeksas bitti kertoo onko kyseessä luku- vai kirjoitusoperaatio. [3]

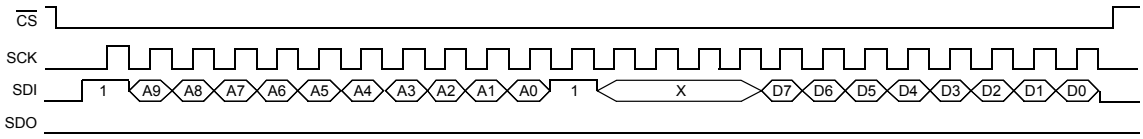


Kuva 5.2: *Lyhyen osoiteavaruuden lukemisen ajoituskaavio. [3]*

void mrf_long_address_write(uint16_t address, uint8_t data)

Funktio toteuttaa MRF24J40 -moduulin pitkään 12-bittiseen muistiavaruuteen kohdistuvat lukuoperaatiot. Apuna tässä työssä funktio käyttää *spi_send_byte()* -funktioita. Parametri *address* pitää sisällään mihin osoitteeseen kirjoitusoperaatio kohdistuu ja *data* sinne kirjoitettavan tiedon.

Funktio manipuloi osoitteen ja datan noudattamaan kuvan 5.3 mukaista muotoa. Tässä rakenteessa ensimmmäisenä lähetetty bitti kertoo onko kyseessä lyhyeen vai pitkään osoiteavaruuteen kohdistuva operaatio ja 14:sta bitti kertoo onko kyseessä luku- vai kirjoitusoperaatio. [3]

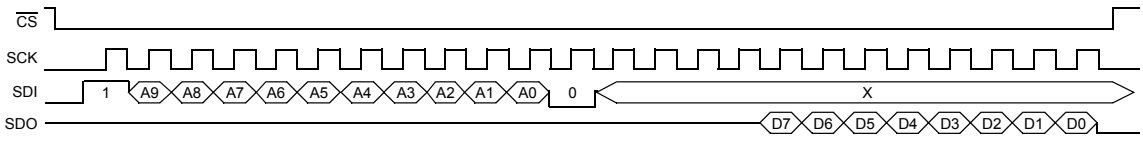


Kuva 5.3: *Pitkän osoiteavaruuden kirjoituksen ajoituskaavio. [3]*

uint8_t mrf_long_address_read(uint16_t address)

Funktio toteuttaa MRF24J40 -moduulin pitkään 12-bittiseen muistiavaruuteen kohdistuvat kirjoitusoperaatiot. Apuna tässä työssä funktio käyttää *spi_send_byte()* ja *spi_receive_byte()* -funktioita. Parametri *address* pitää sisällään mihin osoitteeseen lukuoperaatio kohdistuu. Paluuarvo on kyseisestä osoitteesta luettu tieto.

Funktio manipuloi osoitteen ja datan noudattamaan kuvan 5.4 mukaista muotoa. Tässä rakenteessa ensimmmäinen lähetetty bitti kertoo onko kyseessä lyhyeen vai pitkään osoiteavaruuteen kohdistuva operaatio ja 14:sta bitti kertoo onko kyseessä luku- vai kirjoitusoperaatio. [3]



Kuva 5.4: *Pitkän osoiteavaruuden lukemisen ajoituskaavio. [3]*

void mrf_sleep()

Funktio asettaa MRF24J40 -moduulin virransäästötilaan, jotta tehonkulutus saadaan minimoitua. Kirjasto käyttää tätä funktiota vain jos #MRF_SLEEP_EN on määritetty config.h -tiedostossa.

Jotta MRF24J40MA -moduuli saadaan asetettua uni-tilaan, niin ensin varmistetaan että moduulin herätykseen käytettävä WAKE -pinni on nollassa. Tämän jälkeen

moduulille lähetetään tehonhallinta yksikön nollaus komento. Välittömästi tämän jälkeen lähetetään varsinainen käsky uni-tilaan siirtymiselle. [3]

void mrf_wake()

Funktio herättää MRF24J40 -moduulin WAKE-pinnin avulla. Tämän jälkeen suoritetaan moduulin RF-tilakoneen resetointi datalehden suositusten mukaisesti. [3] Kirjasto käyttää tätä funktiota vain jos #MRF_SLEEP_EN on määritetty. Loppuun on lisätty 2ms viive, jotta tilakone stabiloituisi.

void mrf_send_data(uint16_t address, uint16_t PanId, uint8_t size, char* data)

Funktio hoitaa MRF24J40 -moduulille lähetettävän datan kehystämisen, datan lähetyksen moduulille ja antaa moduulille käskyn aloittaa radiolähetys. Otsikko -kehyksestä löytyvä *framecontrol* sana on kiinteästi koodattu arvo. Tässä työssä kyseinen sana määrittelee, että viesti sisältää dataa, osoitteet ovat 16-bittisiä ja vain vastaanottajan PANID -tunnus löytyy otsikosta. Lähettäjän osoitteeksi funktio asettaa config.h -tiedostosta löytyvän oman osoitteen. Lisäksi kun moduulin uni-tila on sallittu, niin funktio hoitaa moduulin herätyksen *mrf_wake()* -funktiokutsun avulla.

Parametri *address* pitää sisällään vastaanottajan osoitteen. *PanId* -parametrissa kerrotaan minkä PAN -verkon laitteelle viesti on tarkoitettu. Muuttuja *size* kertoo kuinka monta tavua pitkä on viimeisenä parametrina annettu *data* -taulukko.

Lähetyspaketti lähetetään MRF24J40MA -moduulille osoittimien avulla. Tämä on mahdollista sillä MRF24J40 -mikropiiri ja Atmel:n mikroprosessorit noudattavat **little endian** tavujärjestystä. Näin ollen sanan (16 bittiä) mittaiset muuttujat siirtyvät oikein näiden piirien välillä. Kun kaikki data on kirjoitettu MRF24J40MA -moduulin muistiin täytyy erikseen antaa käsky lähetyksen aloitukselle kirjoittamalla TXCON rekisteriin 0x01.

uint8_t mrf_receive_data(uint8_t* data)

Funktio lukee MRF24J40 -moduulin vastaanottaman datan ja tallentaa sen kirjaston sisäiseen lukupuskuriin. Tämä tehdään lukemalla MRF24J40 -moduulin vastaanottopuskurin osoitteet ja kirjoitetaan ne mikrokontrollerin muistin vastaanottopuskuriin järjestyksessä osoittimien avulla. Koska molempien tavujärjestys on **little endian**, niin myös yli yhden tavun mittaiset muuttujat tallentuvat oikein. Tämän lukusuorituksen ajaksi radiolähetysten vastaanotto on poistettu käytöstä, jottei vastaanottopuskurin sisältö vaihdu kesken lukuoperaation.

Tämän jälkeen kirjaston sisäisen tietueen *data* osio kopioidaan viiteparametrina annettuun *data* -taulukkoon. Paluuarvona annetaan kopioidun datan määrä.

void mrf_clear_data()

Funktio tyhjentää kirjaston sisäisen vastaanottopuskurin täyttämällä datapuskurin nollilla ja asettamalla datan pituudeksi nollan.

uint16_t mrf_get_sender_addr(void)

Funktio hakee kirjaston sisäisestä tietorakenteesta lähettäjän osoitteen ja antaa sen paluuarvona.

5.2 SIRC -kirjasto

Kirjasto tuottaa Sonyn laitteiden kanssa yhteensopivaa kommunikaatiota. Infrapunavastaanottimen vaatima 40kHz taajuinen kantoaalto luodaan **Timer0** keskeytyksen avulla. Sonyn laitteet tulkitsevat lähetetyn kantoaallon pituudesta onko lähetetty ykkönen vai nolla. Lähetetyn kantoaallon pituutta säädetään **Timer1** keskeytyksen avulla.

5.2.1 Tietorakenteet

Kirjasto käyttää yhtä tietuetta, johon tallennetaan laitteen osoite ja komento (listaus 5.6). Tietuetta käytetään paketoimaan osoite ja komento yhdeksi parametriksi, jota käytetään infrapunaviestin lähetykseen.

Listaus 5.6: *SIRC -kirjaston tietue.*

```
1 typedef struct SIRC_packet {
      uint8_t Command;
3     uint8_t Address;
  } SIRC_packet_t;
```

Ohjelma on tehty noudattamaan 12-bittistä standardia. Tässä standardisssa on viisi osoitebittiä, jotka ilmoittavat minkä tyypin laitteelle viesti menee, ja loput seitsemän bittiä ovat databittejä. [15; 16]

5.2.2 Funktiot

void init_sirc()

Funktio alustaa ajastimet infrapunaprotokollaa varten. **Timer0** -ajastin alustetaan tuottamaan keskeytys 12,5μs välein, jotta saadaan luotua 40kHz kantoaalto. **Timer1** -lohkosta alustetaan ajastimen vertailukeskeytykset, jotta vertailu arvoa muuttamalla pystytään määrittelemään kuinka kauan kantoaaltoa lähetetään.

void sirc_send(SIRC_packet_t data, uint8_t send_count)

Funktio hoitaa datan lähetyksen apufunktioiden avulla. Parametri *data* pitää sisällään lähetettävän tiedon ja osoitteen. Parametrillä *send_count* ilmoitetaan kuinka monta kertaa ensimmäinen parametri *data* lähetetään.

Sonyn kaukosäätimet lähettävät datan vähintään kolme kertaa [16]. Lähetykset ovat erotettu 24ms hiljaisuudella, jotta lähetysten jaksonajaksi tulisi Sonyn kaukosäätimiä vastaava 45ms aika.

5.2.3 Apufunktiot

static void sirc_send_carrier(uint16_t duration)

Funktio lähettää *duration* -parametrissa määritetyn ajan kantoaaltoa. Parametrin arvo laitetaan sellaisenaan ajastimen vertailuarvoksi. Listaukseen 5.7 on koostettu oleellisimmille lähetysajan kestoille määritellyt makrot, joiden tarkoitus on helpottaa lähdekoodin luettavuutta. Lukemat vastaavat ajastimen vertailuarvoa, kun käytössä on 8MHz kellolähde.

Listaus 5.7: *Kantoaallon lähetyksen apuna käytettävät makrot.*

```
#define SIRC_600us  4800
#define SIRC_1200us  9600
#define SIRC_2400us  19200
```

static void sirc_delay_600us()

Funktio asettaa ajastimen tuottamaan keskeytyksen 600μs kuluttua. Tämän mittaista hiljaisuutta käytetään SIRC -standardissa erottamaan viestin bitit toisistaan. Samaa funktiota käytetään myös erottamaan aloituskehys muusta viestistä.

static void sirc_send_start()

Funktio lähettää aloituskehyksen, eli 2,4ms ajan kantoaaltoa. Tämän jälkeen tulee 600μs hiljaisuus.

static void sirc_send_bits(uint8_t bits, uint8_t bit_count)

Funktio lähettää yhden tavun bitit. Lähetettävän bitin ollessa yksi, niin funktio lähettää 1,2ms ajan kantoaaltoa. Muuten lähetetään 600μs ajan kantoaaltoa. Lähetetyt bitit erotetaan standardin mukaisesti 600μs hiljaisuudella.

Kaikki lähetettävät bitit käydään läpi silmukassa aloittaen vähiten merkitsevästä bitistä. *bit_count* ilmoittaa kuinka monta bittiä silmukassa lähetetään.

5.3 Koordinaattori

5.3.1 Tietueet

Koordinaattorimoduuli pitää sisällään vain yhden tietueen (listaus 5.8). Tämän tietueen tarkoituksena on tallentaa yksittäisen moduulin tiedot yhdeksi kokonaisuudeksi. *LinkedModules[]* -taulukkoon tallennetaan kaikkien moduuleiden osoitteet, joille kyseistä tietuetta vastaavan moduulin viestit kuuluu lähettää. *LinkedModuleCount* -muuttuja pitää kirjaa kuinkan monta osoitetta kyseinen taulukko pitää sisällään.

Toinen puolisko tietuetta kertoo onko kyseistä tietutetta vastaavalle moduulille lähetettäviä viestejä. *SendBuffer[]* -taulukkoon tallennetaan mahdolliset viestit joita moduulille välitetään ja *SendDataCount* pitää kirjaa lähetettävien databittien määrästä.

Listaus 5.8: *Koordinaattorin käyttämä tietue.*

```
1 typedef struct Module {
      uint8_t LinkedModules[MAX_LINKED_MODULES];
3     uint8_t LinkedModuleCount;
      uint8_t SendBuffer[MRF_DATA_PAYLOAD_SIZE];
5     uint8_t SendDataCount;
  } Module_t;
```

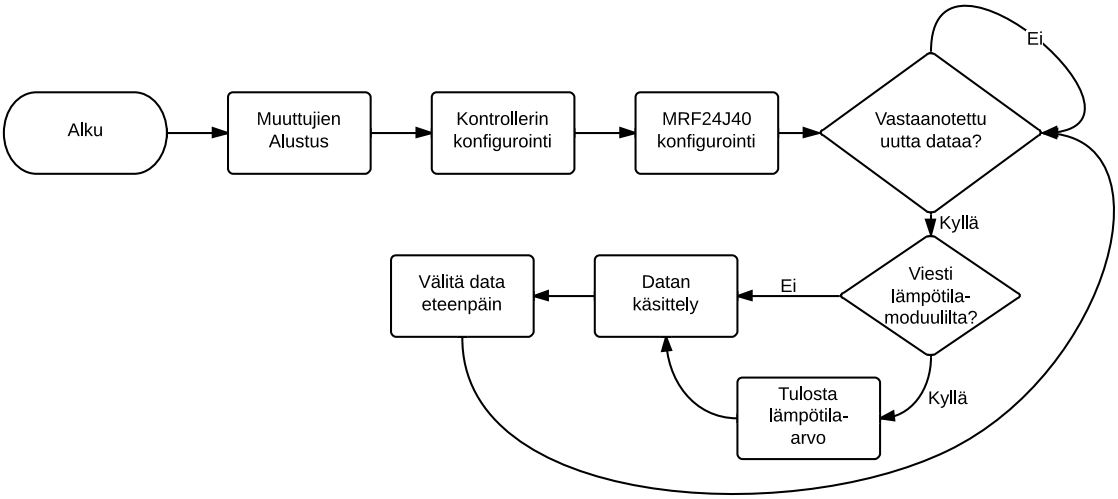
Tästä tietueesta luodaan tietorakenne tekemällä taulukko, jonka alkiot ovat listauksen 5.8, eli *Module*, tietotyyppiä. Tietorakennetta käytetään siten, että taulukon alkio vastaa tarkasteltavan moduulin osoitetta. Esimerkiksi jos näppäimistömoduulin osoite on 0x0004, niin tietorakenteena käytetyn taulukon indeksistä 4 löytyy näppäimistömoduulin tiedot. Tässä tapauksessa *LinkedModules[]* -taulukosta löytyisi rele- ja infrapunamoduuleiden osoitteet. Kun taas *SendBuffer[]* -taulukosta löytyy viesti, jonka koordinaattorimoduuli lähettää näppäimistömoduulille.

5.3.2 Pääohjelma

Koordinaattori moduulin ohjelmisto on toteutettu pollaavana rakenteena. Ohjelma tarkastelee jatkuvasti kuvan 5.5 lohkokaavion mukaisesti onko uutta dataa vastaanotettu. Mikäli dataa on vastaanotettu tarkastetaan lähettäneen moduulin osoite. Tämä tehdään siitä syystä, että eri moduuleiden viestit käsitellään eri tavalla. Lämpötilamoduulilta tulleet lämpötila-arvot käsitellään siten, että ne voidaan tulostaa PC:llä pyörivällä terminaaliohjelmalla. Käytännössä tämä lämpötila-arvojen käsittely muuntaa kahden komplementti muodossa olevat lämpötila-arvot yhden komplementti muotoon. Seuraavaksi luvun numerot muunnetaan yksittäisiksi tulostettaviksi merkeiksi. Lopuksi mahdollinen etumerkki ja luku lähetetään terminaaliohjelmalle.

Lämpötila-arvojen tulostuksen jälkeen, tai jos viesti tulee joltain muulta moduulilta, siirrytään datan käsittelyvaiheeseen. Tässä vaiheessa vastaanotetusta datasta luetaan vastaanottajan osoite. Tämän osoitteen perusteella tarkistetaan koordinaattorimoduulin sisäisestä tietorakenteesta mille moduuleille kyseinen viesti kuuluu välittää *LinkedModules[]* taulukosta. Lopuksi vastaanotetun viestin data tallennetaan asianomaisten moduuleiden *SendBuffer[]* taulukkoon.

Viimeisenä vaiheena tarkistetaan onko sisäisessä tietorakenteessa lähettämätöntä dataa. Kaikki koordinaattorimoduulin sisäisten tietorakenteiden *SendBuffer[]* taulukot lähetetään kyseistä taulukon lokeroa vastaavalle moduulille.



Kuva 5.5: *Koordinaattorin lohkokaavio.*

5.3.3 Keskeytyspalvelut

Apumoduuleilta tulevat viestit luetaan listauksen 5.9 keskeytyspalvelun avulla. Tässä MRF24J40 -moduulilta pyydetään aluksi keskeytystilarekisterin sisältöä. Tämän jälkeen tarkistetaan onko keskeytyksen aiheuttaja lähetys- vai vastaanottokeskeytys. Mikäli kyseessä on vastaanottopuolen keskeytys, niin pyydetään MRF24J40 -moduulilta vastaanottopuskurin sisältö ja tallennetaan paluuarvona tullut datan määrä globaaliin *data_count* -muuttujaan. Tämän jälkeen otetaan vielä talteen viestin lähettäjän osoite globaaliin *data_source* -muuttujaan. Lähetyskeskeytyksen tapauksessa merkataan *mrf_ready* -muuttujan arvoksi tosi, jolloin pääohjelma tietää radiomoduulin olevan valmis uuteen lähetykseen.

Listaus 5.9: *Pääohjelman keskeytyspalvelu.*

```
ISR(PORTB_INT1_vect) {
    uint8_t intstat = 0;
    intstat = mrf_short_address_read(MRF_INTSTAT_ADDR
        );
    if ( intstat&0x08) { // RX interrupt
        Rx_interrupt = true;
        data_count = mrf_receive_data(data);
        data_source = mrf_get_sender_addr();
    }
    else if (intstat&0x01) {
        mrf_ready = true;
    }
}
```

5.3.4 Apufunktiot

void config()

Tämä funktio hoitaa kontrollerin alustukset. Aluksi funktio alustaa kontrollerin kelloksi sisäisen 32MHz kellolähteen. Tämän jälkeen se kutsuu *IO_config()* funktiota hoitamaan IO-porttien alustukset. Lopuksi kutsutaan funktiot alustamaan kontrollerin tietorakenteet.

static void IO_config()

Funktio hoitaa IO -pinnien alustukset ja näihin liittyvien keskeytyspalveluiden alustukset. Tämä funktio ei kuitenkaan koske SPI:n toimintaan liittyviin IO -pinneihin.

static void initDatabase(Module_t Database[MAX_MODULE_COUNT])

Tämä funktion tarkoitus on varmistaa, ettei koordinaattorimoduulin tietorakenteessa ole mitään epämääräisiä arvoja. Tämä varmistetaan kirjoittamalla kaikkiin tietorakenteen alkioihin alkuarvoksi nolla. Parametrina funktion koordinaattorimoduulin tietorakenne.

static void buildDatabase(Module_t Database[MAX_MODULE_COUNT])

Tämän funktion avulla tietorakenteeseen lisätään moduuleiden väliset suhteet. Tämä funktio luotiin, sillä tähän työhön ei tehty käyttöliittymää, jolla pystyttäisiin helposti ja järkevästi asettamaan mitkä viestit kuuluu välittää millekkin apumoduulille. Parametrina funktiolle annetaan käsiteltävä tietorakenne.

static void dataHandler(Module_t Database[MAX_MODULE_COUNT])

Funktio käy koordinaattorimoduulin sisäisestä tietorakenteesta tarkistamassa kenelle vastaanotettu viesti kuuluu lähettää. Tämän jälkeen vastaanotetun viestin data kopioidaan kyseisten apumoduuleiden lähetyspuskuriin odottamaan viestin lähetystä. Parametrina funktiolle annetaan käsiteltävä tietorakenne.

static void dataSendTask(Module_t Database[MAX_MODULE_COUNT])

Funktio käy läpi sisäisen tietorakenteen ja tarkistaa kaikkien moduuleiden vastaanottopuskurin tilan. Datan löytäessään se siirtää datan MRF24J40 -moduulille ja jatkaa tarkistusta. Jos radiomoduuli ei ole saanut edellisen viestin lähetystä valmiiksi, niin odotetaan lähetyksen valmistumista ennen uuden datan lähetystä radiomoduulille. Parametrina funktion koordinaattorimoduulin tietorakenne.

void num2str(char array[6], int8_t integer, int8_t decimal)

Tämä funktio käsittelee liukuluvun tulostettavaan muotoon. Ensin tarkastetaan onko kyseessä negatiivinen vai positiivinen luku. Negatiivisten lukujen tapauksessa merkataan ylös lukujen olevan negatiivisia, jonka jälkeen luvusta tehdään positiivinen vähentämällä se nollasta. Tämän jälkeen jakolaskun ja jakojäännöksien avulla jaetaan luku ykkösiksi, kymmeniksi ja sadoiksi. Vastaava operaatio tehdään luvun desimaalipuoliskolle.

Parametri *array[]* -taulukko pitää sisällään valmiin merkkijonon. Parametrit *integer* ja *decimal* pitävät sisällään muunnettavan luvun. Näistä ensimmäisenä mainittu pitää sisällään liukuluvun kokonaislukuosuuden ja jälkimmäinen desimaaliosuuden.

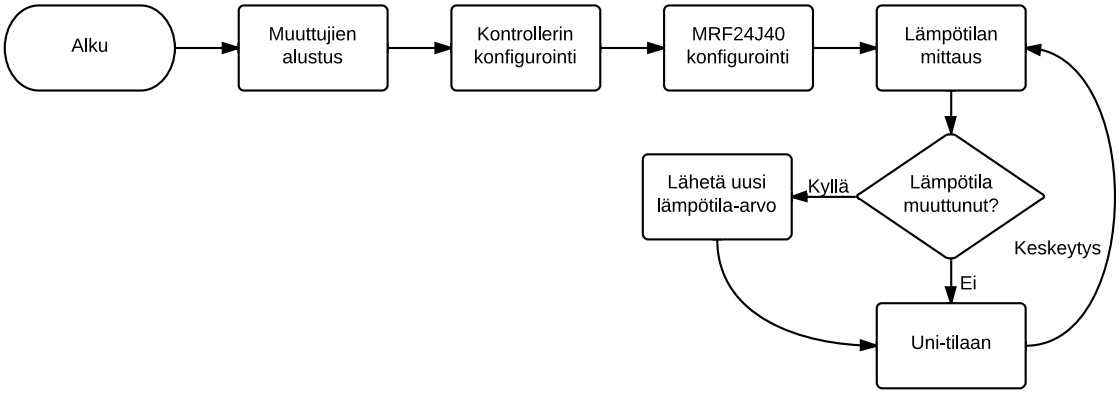
5.4 Lämpömittari

5.4.1 Pääohjelma

Lämpömittarimoduulin ohjelma alkaa muuttujien alustuksilla, jonka jälkeen siirrytään muuttamaan kontrollerin asetuksia. Kontrolleri asetetaan toimimaan sisäisen 8MHz kellolähteen tahdilla. Lisäksi koska kyseessä on paristokäyttöinen moduuli, niin alustetaan kontrollerin uni-tilan ja vahtikoiran asetukset. Lopuksi asetetaan IO-porttien suunnat oikeiksi ja sallitaan keskeytykset, jotta radiomoduuli pystyy ilmoittamaan valmiista tapahtumista.

Radiomoduulin konfiguroinnin jälkeen siirrytään suorittamaan pääohjelman silmukkarakennetta. Tässä rakenteessa suoritetaan ensin lämpötilamittaus. Tämän jälkeen tarkastetaan onko lämpötila-arvo muuttunut edellisen ja uuden mittauksen välillä. Lämpötila-arvon muutoksen havaittuaan kontrolleri lähettää uuden lämpötila-arvon koordinaattorimoduulille. Kun uudet lämpötila-arvot on saatu onnistuneesti

lähetettyä, tai jos lämpötila-arvoissa ei ollut muutosta, niin siirrytään uni-tilaan. Uni-tilasta herätään vahtikoirakeskeytyksen tai radiomoduulin tuottaman keskeytyksen avulla. Tämän jälkeen siirrytään silmukan alkuun ja aloitetaan uusi lämpötilamittaus. Tätä prosessia on havainnollistettu kuvan 5.6 lohkokaaviossa.



Kuva 5.6: *Lämpömittarimoduulin lohkokaavio.*

5.4.2 Keskeytyspalvelut

Lämpömittarimoduuli käyttää hyväkseen kahta kesketyspalvelua. Ensimmäinen on listauksen 5.10 mukainen keskeytyspalvelu, jolla on kaksi toiminnallisuutta: lukea vastaanotettu data ja kun datan lähetys on valmis, käskeä radiomoduulin siirtyä uni-tilaan. Näistä toiminnallisuuksista lämpömittarimoduulin tapauksessa jälkimmäinen on tärkeämpi. Valmiin lähetyksen kohdalle on lisätty makrot, joilla uni-tila pystytään helposti conf.h tiedoston muutoksella poistaa käytöstä.

Listaus 5.10: *Keskeytyspalvelu radiomoduulin datan lukemiselle.*

```
ISR(PCINT0_vect) {
    // RX complete interrupt
    if ( mrf_short_address_read(MRF_INTSTAT_ADDR)&0
        x08 ) {
        data_count = mrf_receive_data(data);
    }
    // TX complete interrupt
    else {
        #ifdef MRF_SLEEP_EN
        mrf_sleep();
        #endif
    }
}
```

Toinen on vahtikoiran keskeytyspalvelu, jonka ainoa tarkoitus on herättää kontrolleri. Tämän vuoksi se on tyhjä keskeytyspalvelu, mutta se täytyy kuitenkin olla luotuna, jotta kontrollerin toiminallisuus pystytään takaamaan

5.4.3 Apufunktiot

static void sleep()

Funktio asettaa kontrollerin uni-tilaan. Ennen uni-tilaan siirtymistä kirjoitetaan vahtikoiran hallintarekisterin keskeytyslipun kohdalle ykkönen. Tällöin saadaan vahtikoira tuottamaan keskeytys nollauksen sijaan, kun laskurin lukualue ylittyy.

static void sleep_250ms()

Funktio asettaa kontrollerin noin 250ms ajaksi uni-tilaan. Vanha vahtikoiran kellolähteen esijakajan arvo luetaan talteen muistiin. Tämän jälkeen asetetaan uusi kellolähteen esijakajan arvo, jolla vahtikoiran laskurin lukualue ylittyy noin 250ms kuluttua. Lisäksi merkataan vahtikoiran hallintarekisterin keskeytyksen sallintaa vastaava bitti ykköseksi. Kun kontrolleri on herännyt uni-tilasta, niin asetetaan vanhat arvot takaisin. Tätä funktiota käytetään korvaamaan viivesilmukoita ja pienentämään näin kontrollerin tehonkulutusta.

void config(void)

Funktio asettaa kontrollerin kellolähteeksi sisäisen kellolähteen. Tämän kellolähteen esijakajaksi asetetaan 1, jolloin saadaan 8MHz kellolähde käyttöön. Tämän jälkeen sallitaan keskeytykset radiomoduulin keskeytyspinniin yhteydessä olevalle pinnille. Lopuksi asetetaan uni-tilan ja vahtikoiran asetukset oikein.

void temperature_read(int8_t temp[2])

Funktio lukee SPI -lohkon avustuksella lämpötilamittauspiirin sisäisestä rekisteristä uuden lämpötila-arvon. Koska lämpötilamittauspiirin sisäinen osoiteavaruus on kahdeksan bittinen, niin tämä 10-bittinen lukuarvo tarvitsee lukea kahdessa osassa.

Luettu lämpötila-arvo palautetaan viiteparametrina annettuun *temp[]* taulukkoon. Arvot tallennetaan siten, että eniten merkitsevä tavu tallennetaan taulukon ensimmäiseen indeksiin ja vähiten merkitsevä jälkimmäiseen indeksiin.

void temperature_start_conversion()

Funktio lähettää SPI -lohkon avustuksella lämpötilamittauspiirille käskyn aloittaa uusi lämpötila-arvon mittaus. Tämä tapahtuu kirjoittamalla heksaluku 0x15 lämpötilamittauspiirin kontrollirekisterin osoitteeseen.

5.5 Näppäimistö

5.5.1 Tietueet

Näppäimistömoduuli käyttää yhtä tietuetta. Molempiin tallennetaan yksittäisten näppäinten tiloja. Tietue on listauksen 5.11 mukainen bittikenttä. Tällä tietueella luodaan tavusta bittikenttä muistin säästämiseksi.

Listaus 5.11: *Näppäimistömoduulin tietue.*

```
typedef struct{
    bool f0:1;
    bool f1:1;
    bool f2:1;
    bool f3:1;
    bool f4:1;
    bool f5:1;
    bool f6:1;
    bool f7:1;
} Flags;
```

Kun tietuetta käytetään yleiskäyttöisten rekisterien **GPIOR0** ja **GPIOR1** kanssa pystytään myös optimoimaan suoritusaikoja. Kyseiset rekisterit toimivat kuten kontrollerin IO -porttien rekisterit, eli niistä pystytään yksittäisiä bittejä asettamaan ja nollaamaan yhdellä kellojaksolla. Tavalliset rekisterit kuten **R30** ja **R31** eivät tue vastaavia komentoja, joten joudutaan käyttämään maskeja. Tämä on hitaampi tapa käsitellä rekisteriä. Makroja käyttämällä tietuetta pystytään käsittelemään kuten mitä tahansa **bool** tyyppistä muuttujaa. Listauksessa 5.12 on esimerkki makroista, jotka toteuttavat tämän toiminnallisuuden. Kääntäjä osaa optimoida esimerkin mukaisen makron käyttämään bittien asetus- ja nollauskomentoja.

Listaus 5.12: *Esimerkki tietueen käytöstä.*

```
#define button1_status ((volatile Flags*) (&GPIOR0))
    -> f0
#define button2_status ((volatile Flags*) (&GPIOR0))
    -> f1
#define button3_status ((volatile Flags*) (&GPIOR0))
    -> f2
```

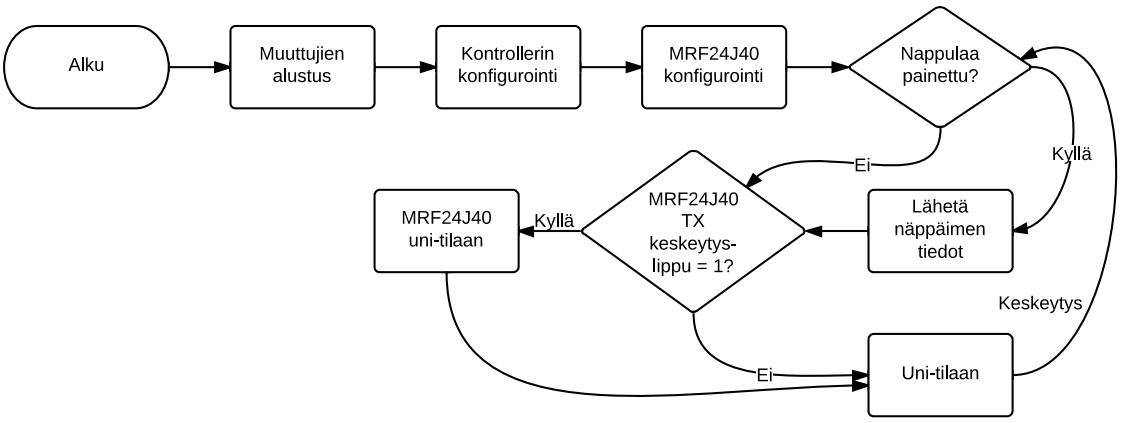
5.5.2 Pääohjelma

Näppäimistömoduulin pääohjelmaa on havainnollistettu kuvan 5.7 lohkokaaviolla. Muuttujien alustuksen jälkeen asetetaan nappuloiden pinnit sisääntuloiksi ja laitetaan sisäiset ylösvedot käyttöön. Näitä näppäimiä vastaavat porttikeskeytykset otetaan myös käyttöön. Tämän jälkeen siirrytään muiden kontrollerin asetusten konfi-

gurointiin. Sisäisen kellolähteen esijakajaksi asetetaan yksi, jotta saadaan se toimimaan 8MHz taajuudella. Lisäksi alustetaan uni-tila ja vahtikoira käyttöön.

Seuraava vaihe on lähettää radiomoduulille sen tarvitsemat asetukset. Pääohjelman silmukassa tarkistetaan ensin onko nappuloita painettu. Jos jotain nappulaa on painettu siirrytään lähettämään tämä tieto koordinaattorille. Muussa tapauksessa siirrytään suoraan tarkastamaan radiomoduulin keskeytysten tilarekisteriä. Tämä tarvitsee tehdä, koska suunnitteluvirheen takia MRF24J40 -moduulin keskeytyspinni on yhteydessä kontrollerin SS-pinniin. Kun radiomoduuli lähettää keskeytyssignaalin, kontrolleri reagoi siihen asettamalla oman SPI -lohkon toimimaan orjana. Tämän jälkeen molemmat laitteet ovat orjia ja joudutaan pattitilanteeseen, jolloin SPI -väylä ei enää toimi.

Havaittaessa radiomoduulin keskeytystilarekisterin lähetys valmis -tilalippu, käsketään sen siirtyä uni-tilaan. Lopuksi asetetaan kontrolleri hetkeksi uni-tilaan ennen uutta pollausta.



Kuva 5.7: *Näppäimistömoduulin lohkokaavio.*

5.5.3 Keskeytyspalvelut

Näppäimistömoduuli käyttää neljää keskeytyspalvelua, joista yksi on tyhjä vahtikoiran keskeytyspalvelu. Tämä keskeytyspalvelu on jätetty tyhjäksi, koska vahtikoira automaattisesti nollaa keskeytysbitin keskeytyksen jälkeen. Tämä keskeytysbitti käskee vahtikoiraa tekemään keskeytyksen laitteen nollauksen sijaan.

Muut keskeytykset ovat listauksen 5.13 tyylisiä keskeytyspalveluita. Koska nämä ovat porttikeskeytyksiä, niin näissä tarkistetaan mitä nappulaa on painettu. Kun havaitaan että nappulaa on painettu, niin asetetaan sitä nappulaa vastaava bitti moduulin tietueesta arvoon tosi.

Listaus 5.13: *Esimerkki näppäimistömoduulin keskeytyspalvelusta.*

```
ISR(PCINT0_vect) {
    if ( !(PINB&BUTTON10_bm) ) {
        button10_status = true;
    }
    if ( !(PINB&BUTTON13_bm) ) {
        button13_status = true;
    }
}
```

5.5.4 Apufunktiot

static void config()

Funktio ottaa käyttöön kontrollerin sisäisen kellolähteen ja asettaa tälle esijakajaksi ykkösen. Näin saadaan kontrollerille käyttöön 8MHz sisäinen kellolähde. Tämän jälkeen asetetaan uni-tilaksi **power-down** -tila ja sallitaan kontrollerin meno unitilaan. Suunnitteluvirheen takia kontrollerin tarvitsee pollata radiomoduulin keskeytystilarekisteria, joten myös vahtikoira on alustettu käyttöön.

static void io_config()

Tässä funktiossa asetetaan nappuloiden käyttämät pinnit sisääntulopinneiksi ja asetetaan niiden ylösvetovastukset käyttöön. Tämän jälkeen sallitaan kontrollerin porttien tilamuutoskeskeytykset kyseisille pinneille.

static void sleep()

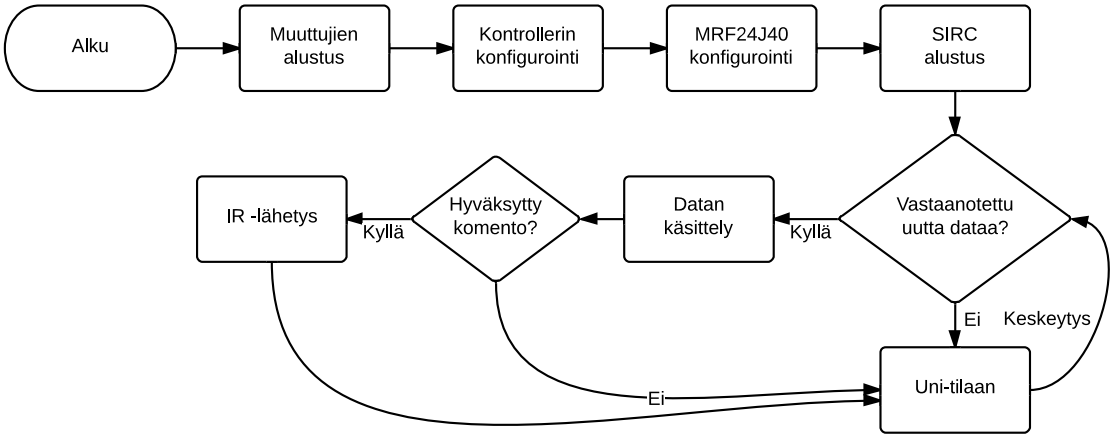
Aluksi funktio käskee vahtikoiraa tuottamaan kontrollerin nollauksen sijaan keskeytyksen, kun tulee laskurin ylivuoto. Tätä bittiä ei tarvitse nollata, koska kontrolleri hoitaa sen itse vahtikoirakeskeytyksen tapahtuessa. Lopuksi käsketään kontrollerin mennä uni-tilaan.

5.6 Infrapuna

5.6.1 Pääohjelma

Aluksi infrapunamoduuli alustaa tarvittavat muuttujat, jonka jälkeen se siirtyy kontrollerin konfigurointiin. Tässä vaiheessa asetetaan sisäisen kellon esijakajaksi yksi, jolloin saadaan kontrollerin kellotaajuus nostettua 1MHz kellotaajuudesta 8MHz taajuuteen. Seuraavaksi kontrolleri sallii uni-tilan ja konfiguroi vahtikoiran toimimaan.

Radiomoduulin alustuksien jälkeen kutsutaan SIRC -kirjaston alustusfunktioita. Tämän jälkeen siirrytään suorittamaan pääohjelman silmukkaa. Kuvan 5.8 lohkokaaviossa on pyritty havainnollistamaan tämän silmukan toiminnallisuutta. Silmukan alussa tarkastetaan onko radiomoduuli vastaanottanut uutta dataa. Jos uutta dataa on vastaanotettu, siirrytään sitä käsittelemään infrapunamoduulin tietorakenteeseen sopivaksi. Muuten siirretään kontrolleri uni-tilaan. Datan käsittelyvaiheessa otetaan viestin datasta vain ensimmäiset kaksi tavua talteen.



Kuva 5.8: *Infrapunamoduulin lohkokaavio.*

Seuraava vaihe on tarkistaa onko komento hyväksytty. Taulukkoon 5.1 on koottu kaikki infrapunamoduulin ymmärtämät tavuyhdistelmät ja mitä perinteisen kaukosäätimen näppäimen toiminnallisuutta simuloidaan. Muut tavuyhdistelmät tulkitaan virheellisiksi ja hylätään.

Taulukko 5.1: *IR-moduulin hyväksytyt komennot.*

1. Tavu	2. Tavu	Komento
0000 0001	0000 0000	Numero 1
0000 0010	0000 0000	Numero 2
0000 0100	0000 0000	Numero 3
0000 1000	0000 0000	Numero 4
0001 0000	0000 0000	Numero 5
0010 0000	0000 0000	Numero 6
0100 0000	0000 0000	Mute
1000 0000	0000 0000	Volumen kasvatus
0000 0000	0000 0001	Kanavan kasvatus
0000 0000	0000 0010	Virtanappi
0000 0000	0000 0100	Volumen pienennys
0000 0000	0000 1000	Kanavan pienennys

Lopuksi infrapunamoduuli lähettää hyväksytyn komenno infrapunaledin avulla

televisiolle. Tämän jälkeen kontrolleri siirtyy uni-tilaan, josta herätään keskeytyksen seurauksena ja aloitetaan silmukan suorittaminen uudelleen.

5.6.2 Keskeytykset

Relemoduulilla on käytössä kaksi keskeytyspalvelua. Ensimmäinen on listauksen 5.14 mukainen portin tilamuutoskeskeytyksen keskeytyspalvelu. Tämä tarkoittaa, että radiomoduuli on vastaanottanut uutta dataa. Keskeytyspalvelu hoitaa uuden datan lukemisen radiomoduulilta.

Listaus 5.14: *Keskeytyspalvelu radiomoduulin datan lukemiselle.*

```
ISR(PCINT0_vect) {
    // RX complete interrupt
    if ( mrf_short_address_read(MRF_INTSTAT_ADDR)&0
        x08 ) {
        data_count = mrf_receive_data(data);
    }
}
```

Toinen keskeytyspalvelu on tyhjä vahtikoiran ylivuotokesketyspalvelu. Tämä on jouduttu lisäämään, että vahtikoiraa voidaan käyttää kontrollerin herätykseen.

5.6.3 Apufunktiot

void config(void)

Tässä funktiossa asetetaan kontrollerin sisäisen kellolähteen esijakajaksi yksi, jotta saadaan se toimimaan 8MHz kellotaajuudella. Lisäksi sallitaan kontrollerin uni-tila ja asetetaan vahtikoira toimintaan. Vahtikoira tuottaa ylivuodon noin yhden sekunnin välein.

Lisäksi funktio sallii portin tilamuutoskeskeytykset radiomoduulin keskeytyspinnille. Lopuksi asetetaan infrapunalediä ohjaava pinni ulostuloksi ja varmistetaan, että se ei ole päällä.

static void sleep()

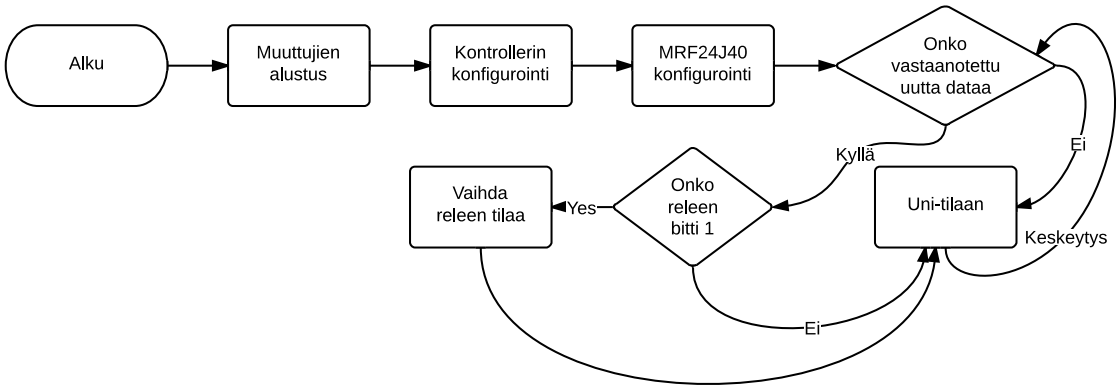
Funktio käskee aluksi vahtikoiraa tuottamaan kontrollerin nollauksen sijaan keskeytyksen. Tämän jälkeen asetetaan kontrolleri uni-tilaan.

5.7 Rele

Relemoduuli toimii kuvan 5.9 lohkokaavion mukaisesti. Muuttujien alustusten jälkeen konfiguroidaan kontrolleri toimimaan 8MHz taajuudella. Lisäksi sallitaan kont-

rollerin uni-tila ja laitetaan vahtikoira päälle. Tämän jälkeen asetetaan relettä ohjaava pinni ulostuloksi.

Radiomoduulin konfiguroinnin jälkeen siirrytään pääohjelman silmukkaan. Tässä tarkistetaan aluksi onko uutta dataa vastanotettu. Vastaanotetusta viestistä tarkistetaan kolmannentoista bitin tila. Tämän bitin ollessa yksi vaihdetaan releen tilaa. Silmukan lopuksi asetetaan kontrolleri uni-tilaan. Keskeytyksen seurauksena aloitetaan silmukan läpikäynti uudestaan.



Kuva 5.9: *Relemoduulin lohkokaavio.*

5.7.1 Keskeytyspalvelut

Tässä moduulissa käytetään kahta keskeytyspalvelua, joista toinen on listauksen 5.14 mukainen portin tilamuutoskeskeytys. Tämä hoitaa radiomoduulin vastaanottamien viestien lukemisen.

Toinen keskeytyspalvelu on tyhjä vahtikoiran ylivuotokeskeytys. Tällä taataan kontrollerin oikea toiminnallisuus, kun halutaan vahtikoiran herättävän kontrolleri.

5.7.2 Apufunktiot

void config(void)

Tässä funktiossa asetetaan kontrollerin sisäisen kellolähteen esijakajaksi yksi, jotta saadaan se toimimaan 8MHz kellotaajuudella. Lisäksi sallitaan kontrollerin uni-tila ja asetetaan vahtikoira toimintaan. Vahtikoira tuottaa ylivuodon noin yhden sekunnin välein.

Lisäksi funktio sallii portin tilamuutoskeskeytykset radiomoduulin keskeytyspinnille. Lopuksi asetetaan infrapunalediä ohjaava pinni ulostuloksi ja varmistetaan että se ei ole päällä.

static void sleep()

Funktio käskee aluksi vahtikoiraa tuottamaan kontrollerin nollauksen sijaan keskeytyksen. Tämän jälkeen asetetaan kontrolleri uni-tilaan.

static void RELAY_TGL()

Funktiossa luetaan portin tila. Tämän jälkeen **XOR** -operaation avulla muutetaan releen ohjauspinniä vastaavan bitin arvo. Lopuksi tämä arvo asetetaan kyseisen portin ulostulon tilaksi.

6. TESTAUS- JA MITTAUSTULOKSET

Moduuleiden valmistuksen jälkeisissä mittauksissa osoittautui, että jokaisessa piirilevyssä oli pieni oikosulku signaalilinjojen ja maa- tai käyttöjännitetason kanssa. Kyseiset oikosulut olivat kooltaan niin pienet, että 10 kertaa suurentavalla suurennuslasilla niitä ei pystytty havaitsemaan. Pieni pyyhkäisy kääntöteräveitsellä poisti kyseisen ongelman useimmissa tapauksissa. Lämpötilamoduulissa oikosulku paikannettiin radiomoduulin alle, joten siihen ei päästy kääntöteräveitsellä käsiksi. Tämä ongelma korjattiin pakottamalla signaalilinjan käyttöjännitteet loogista ykköstä vastaavaan tilaan, eli 3,0V tasoon.

Laadullisesti moduuleiden piirilevyjen valmistus onnistui hyvin. Moduuleissa oli hieman ylivalotuksesta johtuvia jälkiä, eli pieniä reikiä kuparissa. Vain kahdessa moduulissa nämä jäljet osuivat signaalilinjoihin. Tästä syystä näppäimistömoduulissa ja infrapunamoduulissa on juotostinalla päällystetty hieman signaalilinjoja. Kuvat valmiista piirilevyistä löytyvät liitteinä: koordinaattorimoduulin liitteissä 6 ja 7, infrapunamoduulin liitteissä 8 ja 9, relemoduulin liitteissä 10 ja 11, näppäimistömoduulin liitteissä 12 ja 13, ja lämpömittarimoduulin liitteissä 14 ja 15.

Laitteet ja ohjelmisto ovat testauksen aikana toimineet hyvin. Näppäimistömoduulin nappia painettaessa rele- ja infrapunamoduuli reagoivat hyvin nopeasti. Silmämääräisellä testauksella ei pysty havaitsemaan onko tekemämme järjestelmän vai television kaukosäätimen vasteaika pienempi. Tarkemman vasteajan määrittelyyn tarvittavaa mittausjärjestelmää ei ole käytettävissä.

Testausjakson aikana näppäimistömoduuli on jättänyt reagoimatta muutamaan näppäimen painallukseen. Tätä virhetoiminnallisuutta ei ole pystytty toistamaan mitenkään systemaattisesti. Todennäköisimmäksi syyksi on arvioitu, että tämä virhe johtuu uni-tilasta heräämisestä aiheutuneesta viiveestä, huonosta nopeasta näppäimen painalluksesta tai näiden yhdistelmästä. Koska virhe on näin satunnainen, niin lopullista syytä ei ole onnistuttu selvittämään.

6.1 Radiomoduulin virrankulutus

Näppäimistömoduulilta mitattiin radiomoduulin lähetysaikoja. Mittausten mukaan näppäimistömoduulin näppäimen painalluksesta aiheutuva lähetys kestää noin 12,8 ms. Kuvasta 6.1 nähdään myös, että radiomoduulin virrankulutus on jonkinverran pienempi, kun sen lähetysrekisteriin kirjoitetaan viesti.

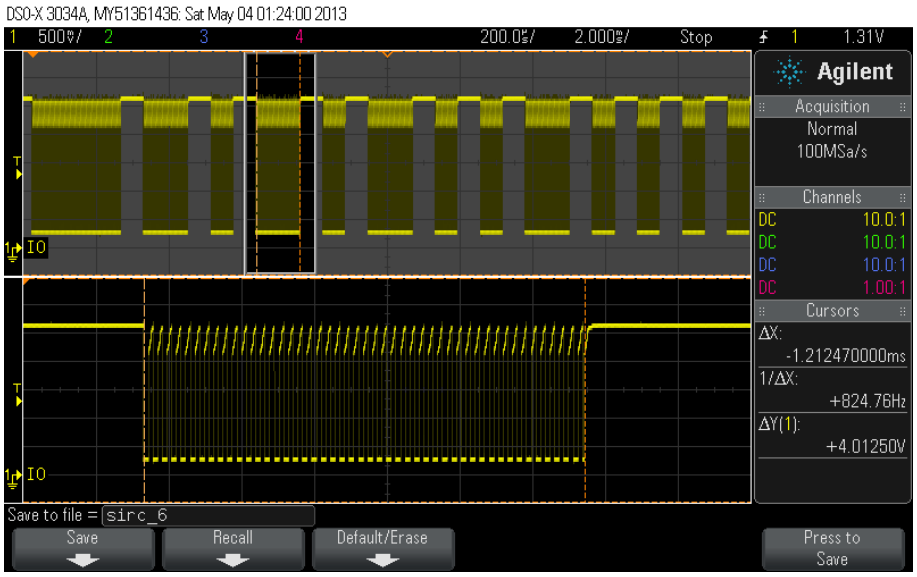
Kuva 6.1: *Näppäimistömoduulin jännitehäviö 1Ω virranmittausvastuksen yli.*

Vastuksen tarkaksi arvoksi mitattiin 0,9824 Ω. Tästä sadaan laskettua, että lähetystilassa näppämistömoduulin virrankulutus lähetystilassa on noin 47,59mA. Tämä on huomattavasti enemmän, kuin datalehtien ilmoittamat 23mA radiomoduulille ja 2mA kontrollerille [11; 2]. Datalehtien mittausarvot ovat ilmoitettu 3,3V jännitteillä ja mittaus tilanteessamme oli kontrollerin käyttöjännitteet noin 3,0V, joka vastaa paristoilla tuotettua käyttöjännitettä.

6.2 SIRC

Tekemämme lähetysalgoritmi SIRC-protokollalle suoriutuu määritelmistä erittäin hyvin. Käyttämällä kontrollerin sisäisiä ajastinlohkoja aikakriittisten toimintojen suorittamiseen, päästiin alle 1%:n tarkkuuteen. Esimerkiksi loogista ykköstä vastaava 1,2ms mittainen kantoaaltopurske on mittausten perusteella 1,21ms pituinen. Kuvassa 6.2 on esitetty tämä mittaustulos. Kuvan ylemmällä puoliskolla on koko lähetetty viesti ja alemmassa puoliskossa on yhden bitin lähetyksen aaltomuoto.

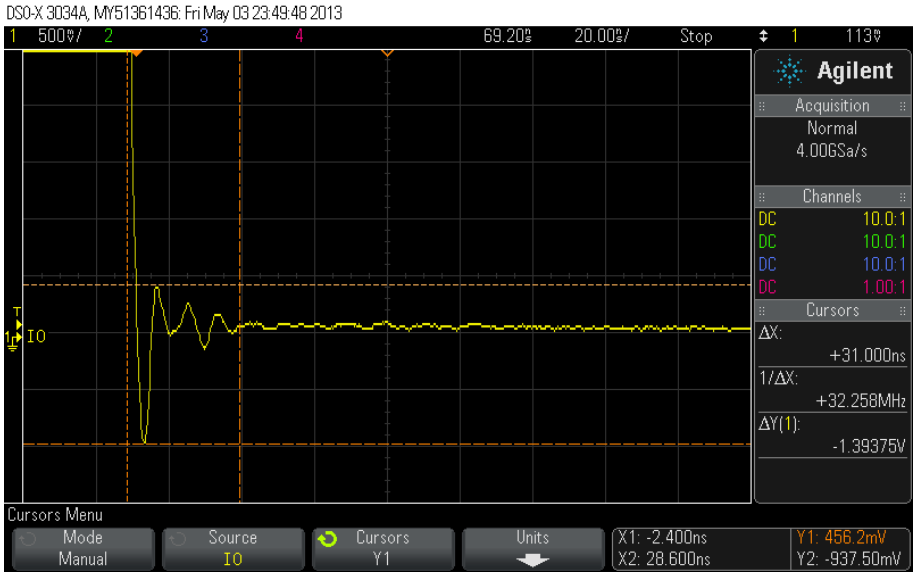
Vastaava tarkkuus löytyy myös loogista nollatilaa vastaavasta ja aloituskehystä vastaavasta kantoaaltopurskeesta. Aloituskehyksen ja ensimmäisen lähetetyn bitin purskeiden välissä oleva hiljaisuus on venynyt yli 640 μs mittaiseksi. Tämä aiheutuu bittien lähetysfunktion alussa olevista muuttujien alustuksista ja funktion parametrien välityksestä aiheutuvista viiveistä.



Kuva 6.2: *Loogisen ykkösen mittaustulos.*

6.3 Näppäimistö

Nappuloille rakennettu kytkinvärähtelyn poistopiiri suoriutuu tehtävästään hyvin. Kuvan 6.3 mittaustuloksesta nähdään, että värähtelyn positiivinen huippuarvo jää alle 500mV tasolle. Tämä jännitetaso jää kauas kontrollerin vaatimasta 1,6V jännitetasosta, joka vaaditaan loogisen ykköstason tulkintaan [11]



Kuva 6.3: *Kytkinvärähtelyn mittaustulos.*

Testauksen alkuvaiheessa huomattiin että moduuli on liian leveä. Tästä syystä sen käytettävyys on heikko yhdellä kädellä käytettäessä. Mikäli laitetta käytetään

esimerkiksi pöydän päällä perinteisen tietokoneen näppäimistön tyylisesti, niin moduulin pituussuunnassa on paljon turhaa tilaa. Lisäksi ohjelmointiliittimen sijainti on huono oikeakätisille käyttäjille. Käytettäessä liittimen pinnit painavat helposti kämmeneen ja häiritsevät käytettävyyttä. Kotelointi poistaisi tämän ongelman, mutta koska laite on jo valmiiksi liian suuri, niin yksinkertainen kotelo ei välttämättä parantaisi käytettävyyttä.

6.4 Lämpömittari

Lämpötila-arvojen mittaustulosten tarkasteluun apuna käytettiin kahta kuluttajalle suunnattua sisä- ja ulkoilman lämpömittaria. Mittaukset suoritettiin ottamalla kaikkien lämpömittareiden lämpötila-arvot muistiin sillä hetkellä, kun lämpömittarimoduuli suorittaa lämpötila-arvon päivityksen. Lämpömittarimoduulin lämpötila-arvojen mittaustulokset ovat koostettu taulukkoon 6.1.

Taulukko 6.1: *Lämpötila-arvojen mittaustulokset.*

Lämpömittari 1	Lämpömittari 2.	Lämpömittarimoduuli
24,0°C	24,3°C	23,75°C
24,3°C	24,3°C	23,50°C
24,3°C	24,4°C	23,75°C
24,4°C	24,4°C	23,50°C
24,0°C	24,4°C	23,75°C
24,1°C	24,4°C	23,75°C
24,2°C	24,7°C	23,50°C
24,1°C	24,8°C	23,50°C
24,1°C	24,9°C	23,75°C
24,1°C	24,9°C	23,50°C
24,3°C	24,9°C	23,75°C

Mittausarvoista huomataan, että kaupallisten lämpömittareiden esittämät arvot esitetään tarkemmassa muodossa, kuin lämpötilamoduulin mittaustulokset. Toisaalta kaupallisten mittareiden lämpötila-arvot vaihtelevat enemmän. Lämpömittarimoduulin pienempi tarkkuus lämpötila-arvoissa saa mittaustuloksen näyttämään vakaammalta.

7. YHTEENVETO

Työ onnistui kohtalaisen hyvin ensimmäiseksi versioksi. Parhaiten onnistunut moduuli on lämpömittarimoduuli. Tässä moduulissa ei ole muita suuria virheitä, kuin USI-lohkon virheelliset kytkennät. Koska kyseessä on vain lämmönmittausmoduuli, niin tämän fyysinen koko voisi olla huomattavasti pienempi. Lämpötila-arvot pysyivät johdonmukaisesti samassa tasossa käytössä olevan lämpömittarin kanssa, joten lämpötila-arvoihin ollaan hyvin tyytyväisiä.

Näppäimistömoduulista löytyy eniten parannuskohteita. Tämän moduulin käytettävyyttä haittaa sen suuri leveys. Leveyden takia laitteesta ei saa hyvää otetta siten, että moduulin kaikkia näppäimiä pystyisi helposti käyttämään. Lisäksi suunnitteluvirheen takia radiomoduulin keskeytyssignaalia ei pystytä käyttämään hyväksi.

Muut moduulit toimivat odotusten mukaisesti. Näissä moduuleissa on myös parannettavaa koon suhteen. Lisäksi kunnollinen kotelointi saisi laitteet näyttämään siistimmältä.

Työssä opittiin paljon radiomoduulin, etenkin IEEE 802.15 standardin, ja SIRC-protokollan toiminnasta. Lisäksi tuli perehdyttyä Xmega-sarjan kontrolleriin ja opittua paljon uutta tämän ohjelmoinnista. Näissä kontrollereissa on huomattavasti enemmän rekisterejä tekemään erilaisia asioita kuin ATmega- tai ATtiny -sarjoissa.

LÄHTEET

[1] Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)," Saatavissa: http://standards.ieee.org/about/get/802/802.15.html, 2011.

[2] Microchip, "MRF24J40MA data sheet," Saatavissa: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en535967, Rev. C, 11/2008, Viitattu: 19.11.2012.

[3] ——, "MRF24J40 data sheet," Saatavissa: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en535967, Rev. C, 08/2010, Viitattu: 19.11.2012.

[4] Atmel, "XMEGA - USB hardware design recommendations," Saatavissa: http://www.atmel.com/devices/atxmega128b3.aspx?tab=documents, Rev. A, 07/2012, Viitattu: 28.11.2012.

[5] N. Semiconductors, "NX1117C; NX1117CE series Low-dropout linear regulators," Saatavissa: http://www.nxp.com/products/power_management/linear_voltage_regulators/ldo_medium_voltage/NX1117CE50Z.html#documentation, Rev. 1, 06/2011, Viitattu: 06.12.2012.

[6] Duracell, "Mx2400 - data sheet," Saatavissa: http://www.duracell.com/en-US/Global-Technical-Content-Library/Product-Data-Sheets.jspx, 06/2008, Viitattu: 19.11.2012.

[7] Myrra, "47000 Series - Electronic Transformers - Datasheet," Saatavissa: http://www.myrra.com/en/productsmyrra, viitattu: 18.12.2012.

[8] Atmel, "Application Note - AVR1005: Getting started with XMEGA," Saatavissa: http://www.atmel.com/products/microcontrollers/avr/avr_xmega.aspx?tab=documents, Rev. B, 11/2009, Viitattu: 06.12.2012.

[9] ——, "Xmega B manual," Saatavissa: http://www.atmel.com/devices/atxmega128b3.aspx?tab=documents, Rev. A, 10/2011, Viitattu: 28.11.2012.

[10] D. Sarkisian, "Micro miniature pushbutton switch fsm jsm," Saatavissa: http://www.te.com/catalog/pn/en/4-1437565-1, Rev. S, 01/2008, Viitattu: 06.12.2012.

[11] Atmel, "Atmel 8-bit Microcontroller with 4/8/16/32KBytes In-System Programmable Flash," Saatavissa: http://www.atmel.com/devices/ATMEGA88A.aspx?tab=documents, Rev. E, 08/2012, Viitattu: 06.12.2012.

[12] Microchip, "Digital temperature sensor with SPI interface," Saatavissa: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010748, Rev. B, 07/2011, Viitattu: 19.11.2012.

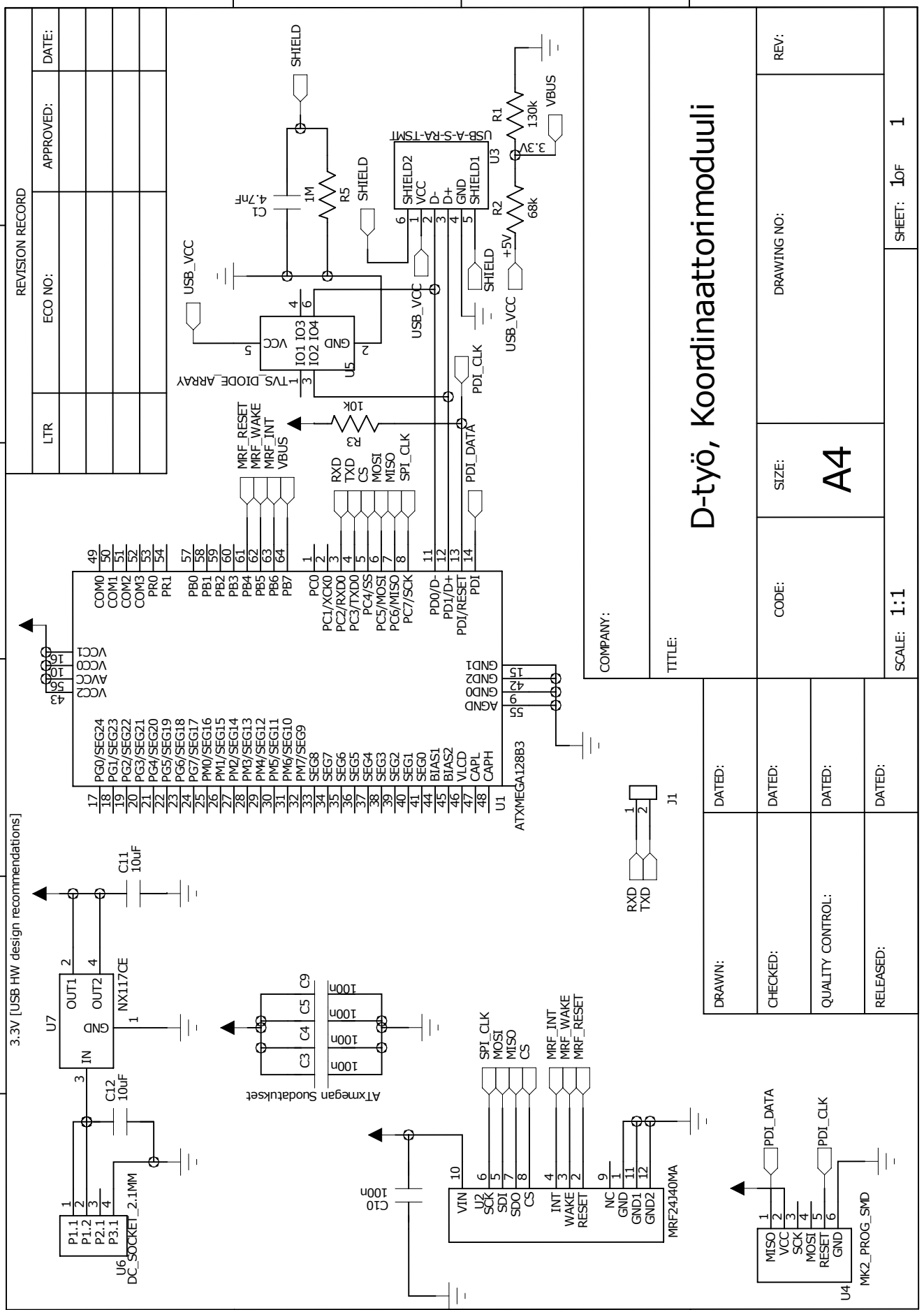
[13] VISHAY, "High power infrared emitting diode, 940nm, GaAlAs/GaAs," Saatavissa: www.vishay.com/docs/81006/tsal4400.pdf, Rev. 1.7, 08/2011, Viitattu: 18.12.2012.

[14] multicomp, "Power relays," Saatavissa: http://www.farnell.com/datasheets/1318173.pdf, Rev. 1.1, 04/2011, Viitattu: 18.12.2012.

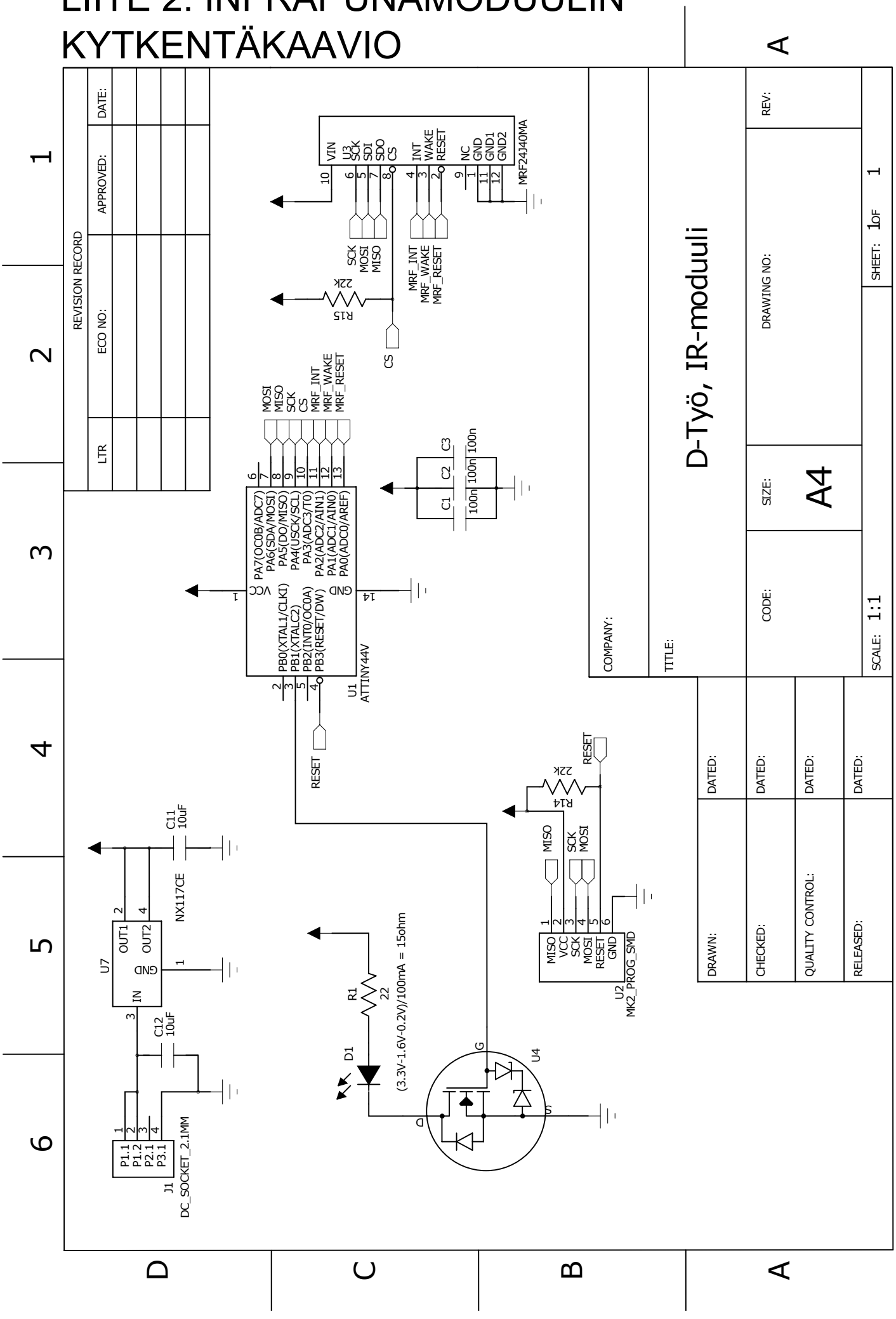
[15] , "Sony sirc protocol [www]," Saatavissa: http://www.sbprojects.com/knowledge/ir/sirc.php, 05/2011, Viitattu: 12.02.2013.

[16] , "Sony sirc infrared protocol [www]," Saatavissa: http://picprojects.org.uk/projects/sirc/, 02/2010, Viitattu: 12.02.2013.

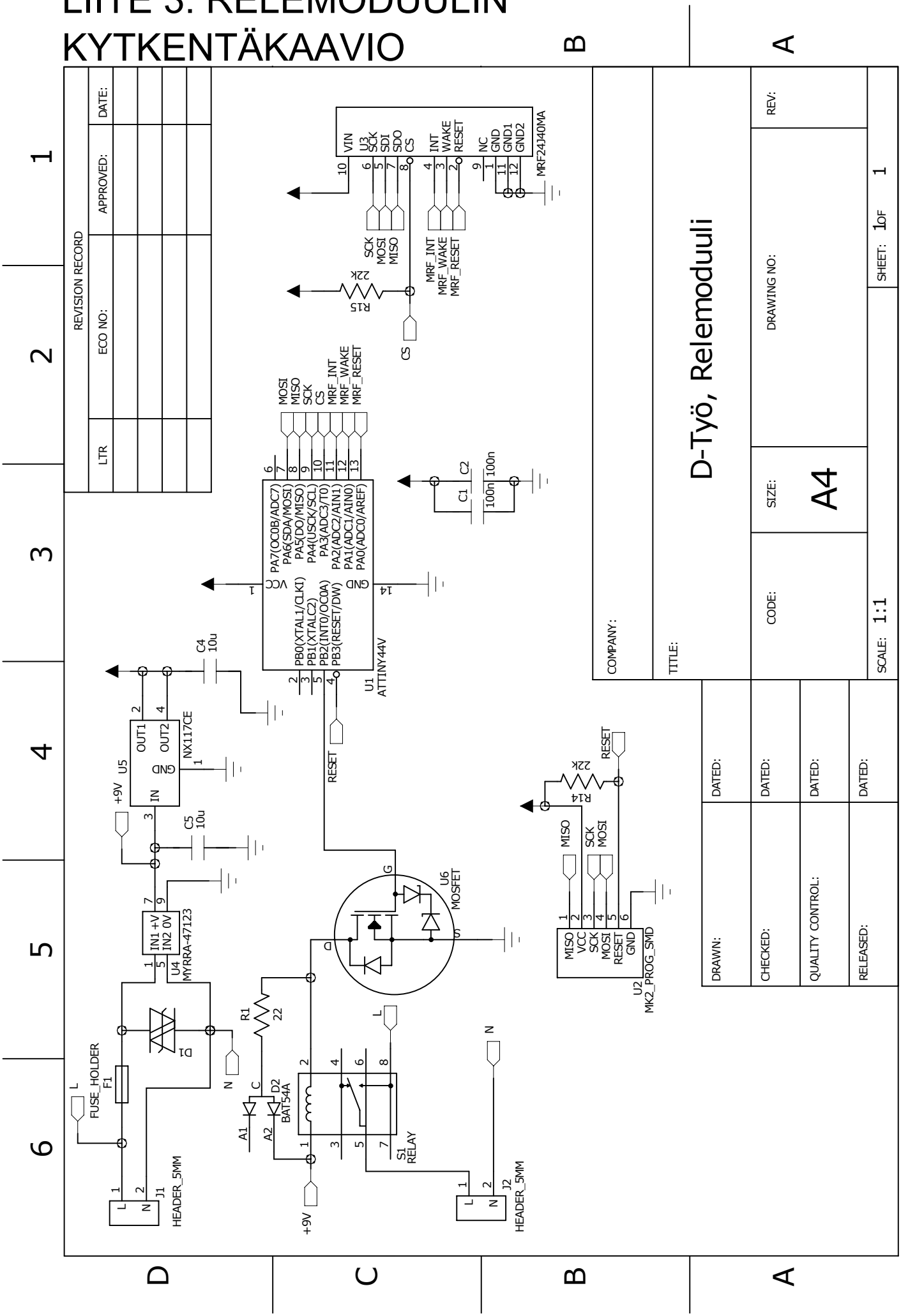
LIITE 1: KOORDINAATTORIMODUULIN KYTKENTÄKAAVIO



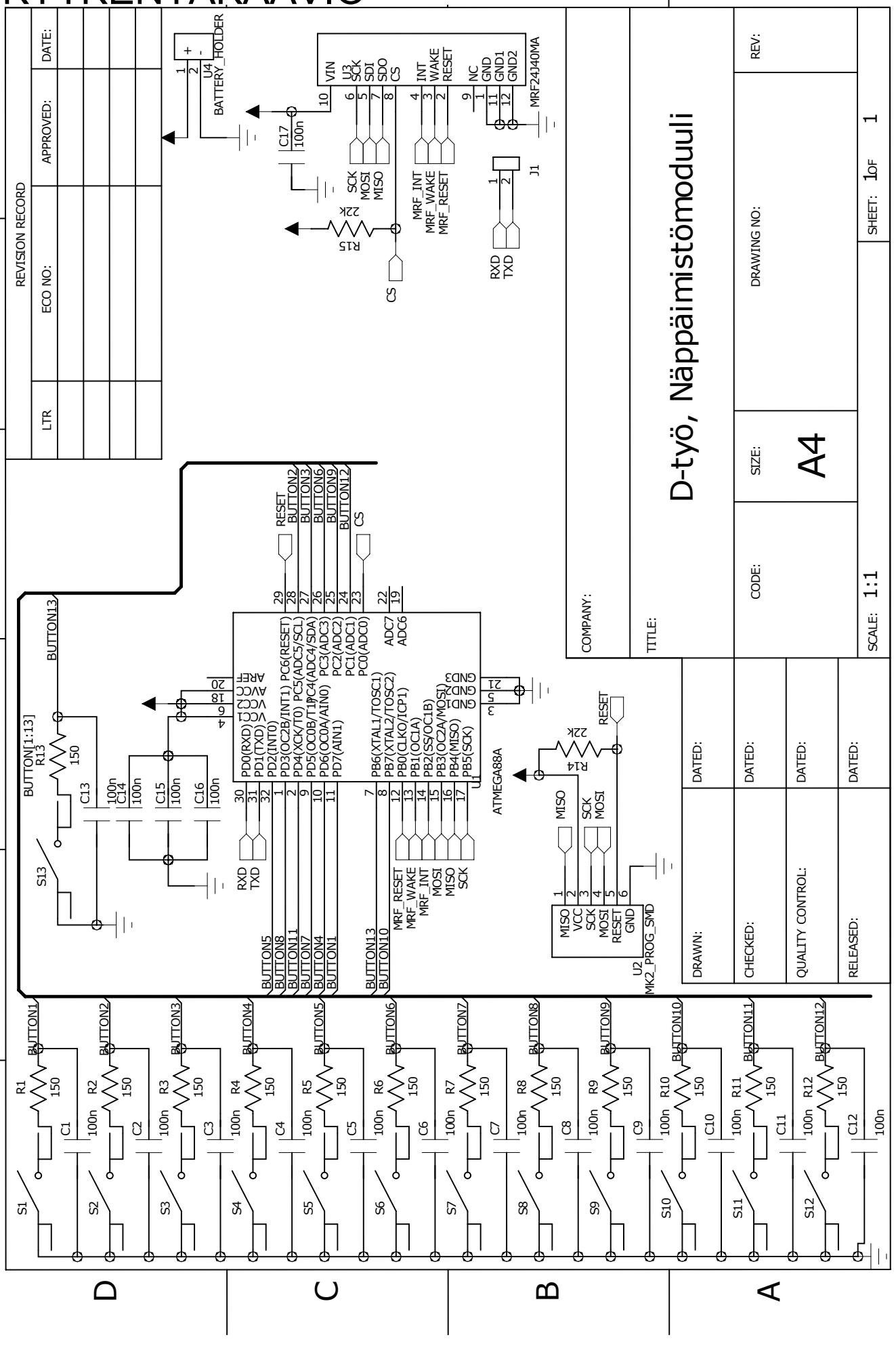
LIITE 2: INFRAPUNAMODUULIN KYTKENTÄKAAVIO



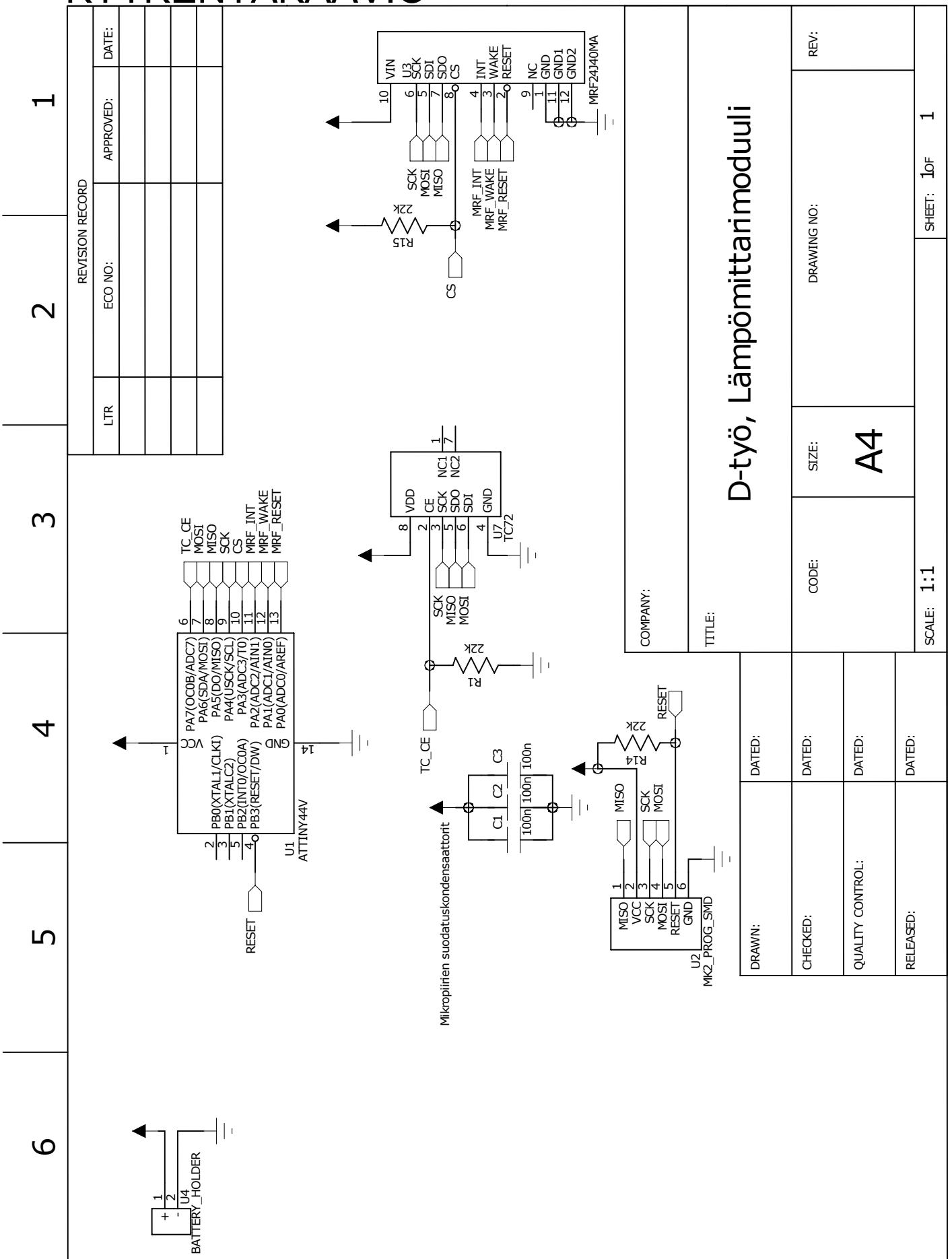
LIITE 3: RELEMODUULIN KYTKENTÄKAAVIO



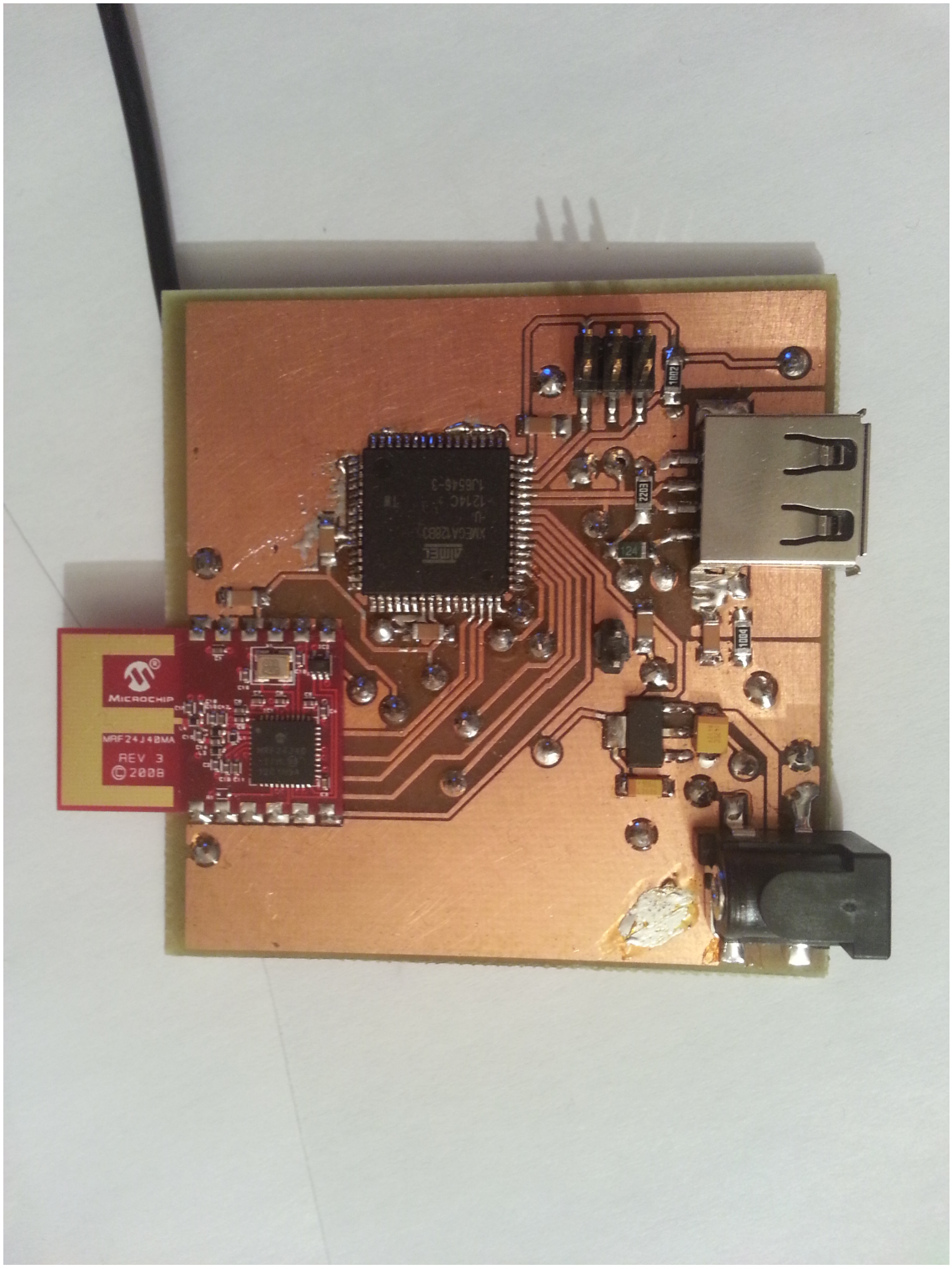
LIITE 4: NÄPPÄIMISTÖMODUULIN KYTKENTÄKAAVIO



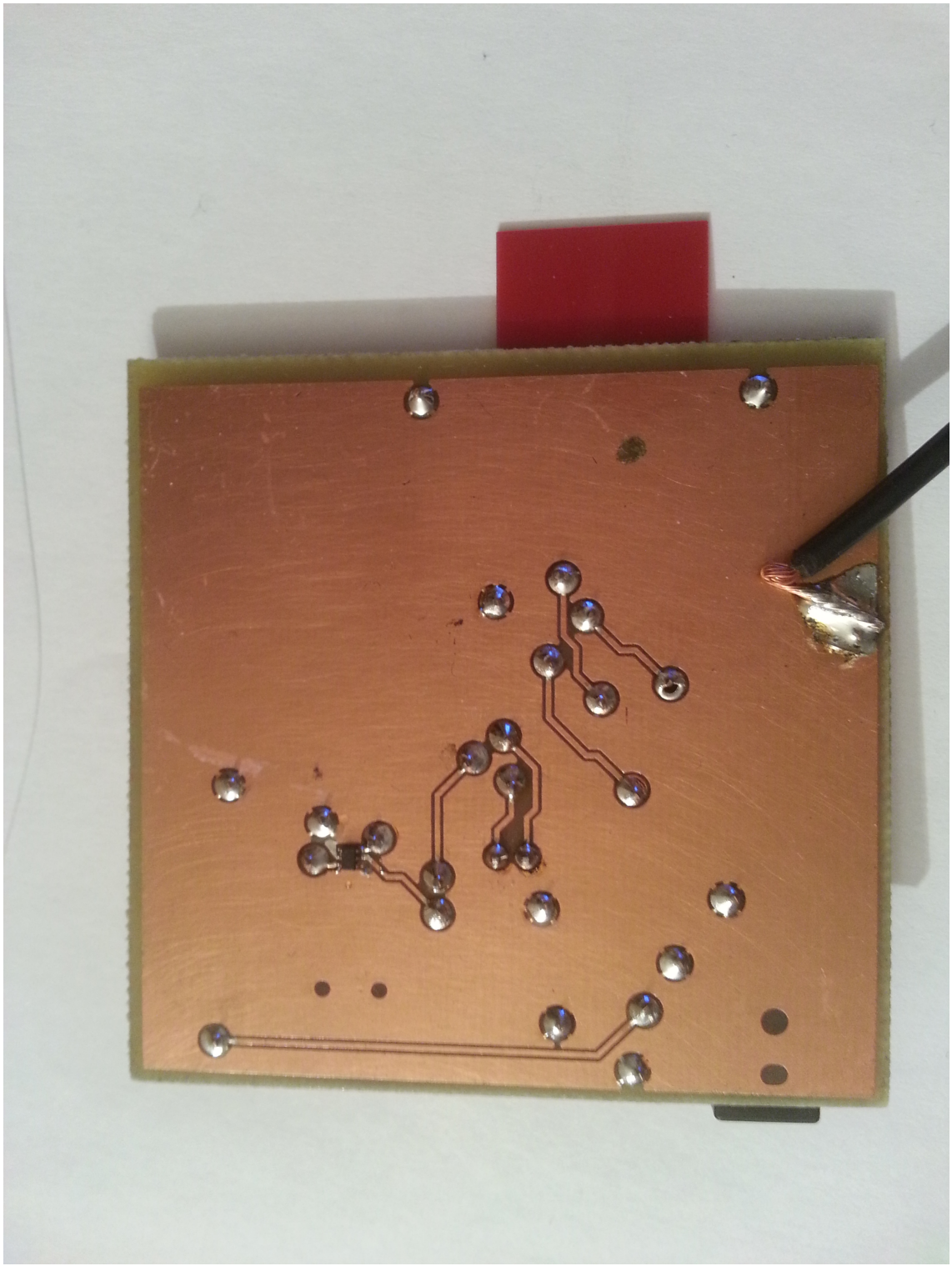
LIITE 5: LÄMPÖMITTARIMODUULIN KYTKENTÄKAAVIO



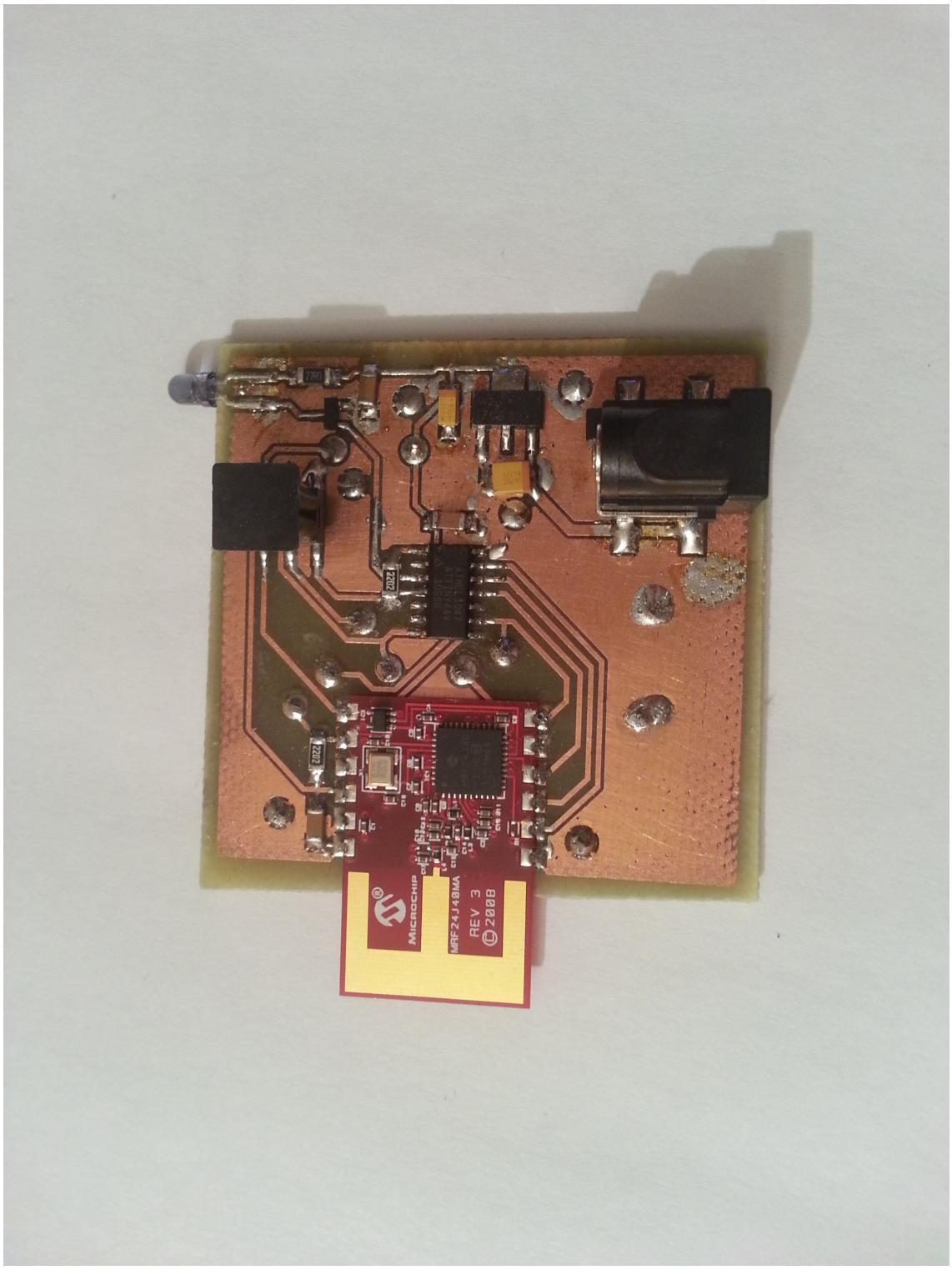
LIITE 6: KOORDINAATTORIMODUULIN YLÄPUOLI



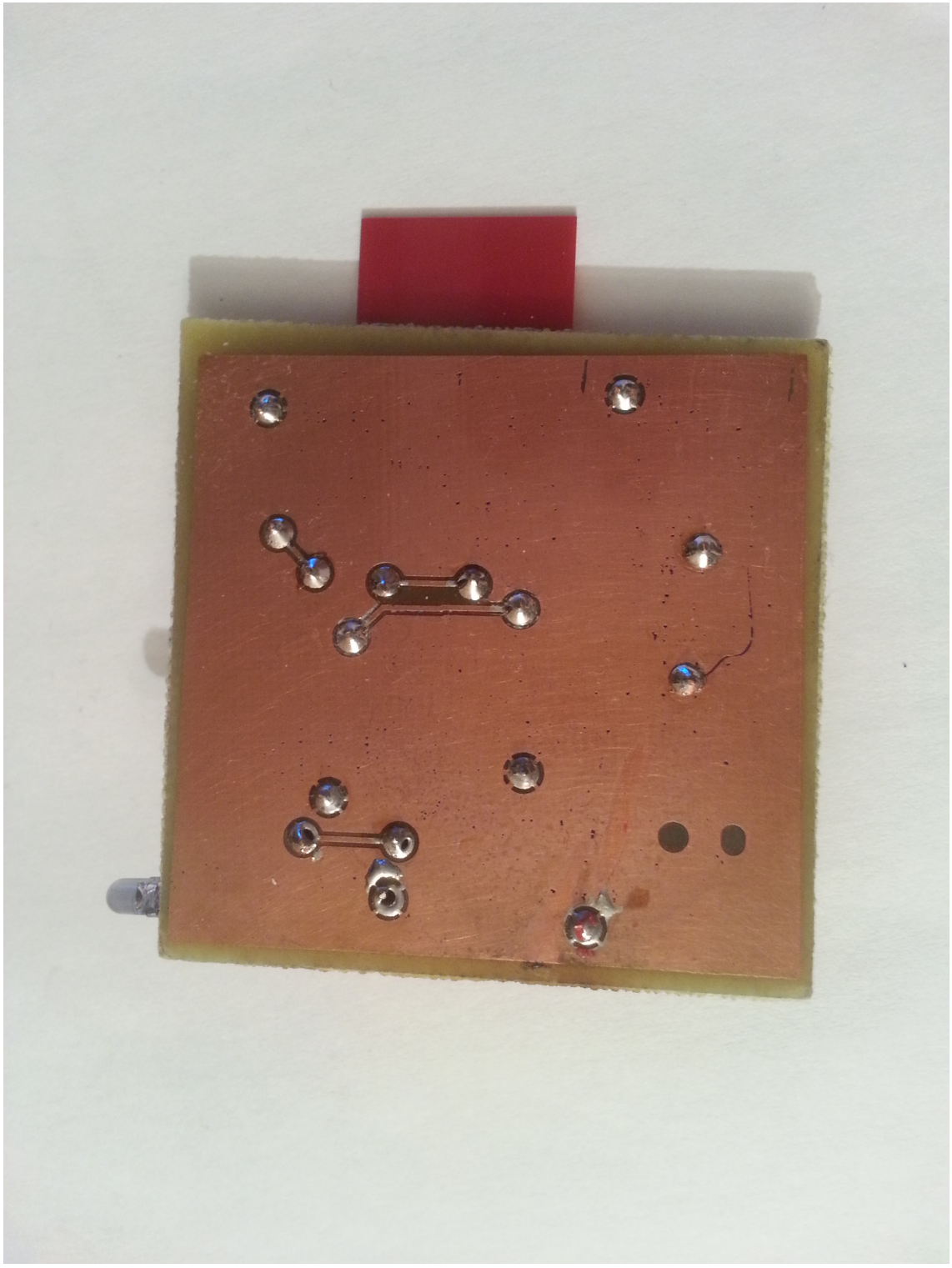
LIITE 7: KOORDINAATTORIMODUULIN ALAPUOLI



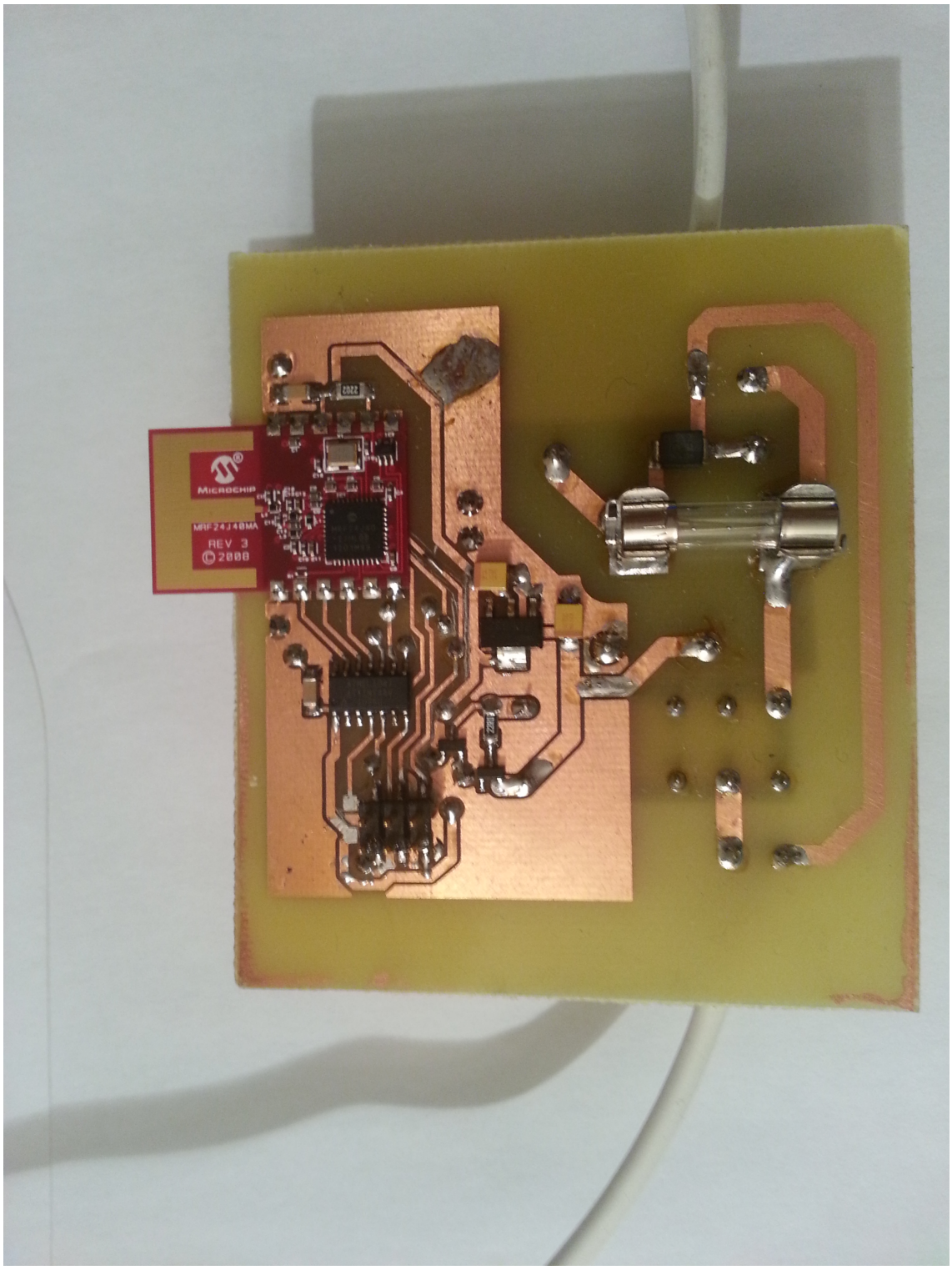
LIITE 8: INFRAPUNAMODUULIN YLÄPUOLI



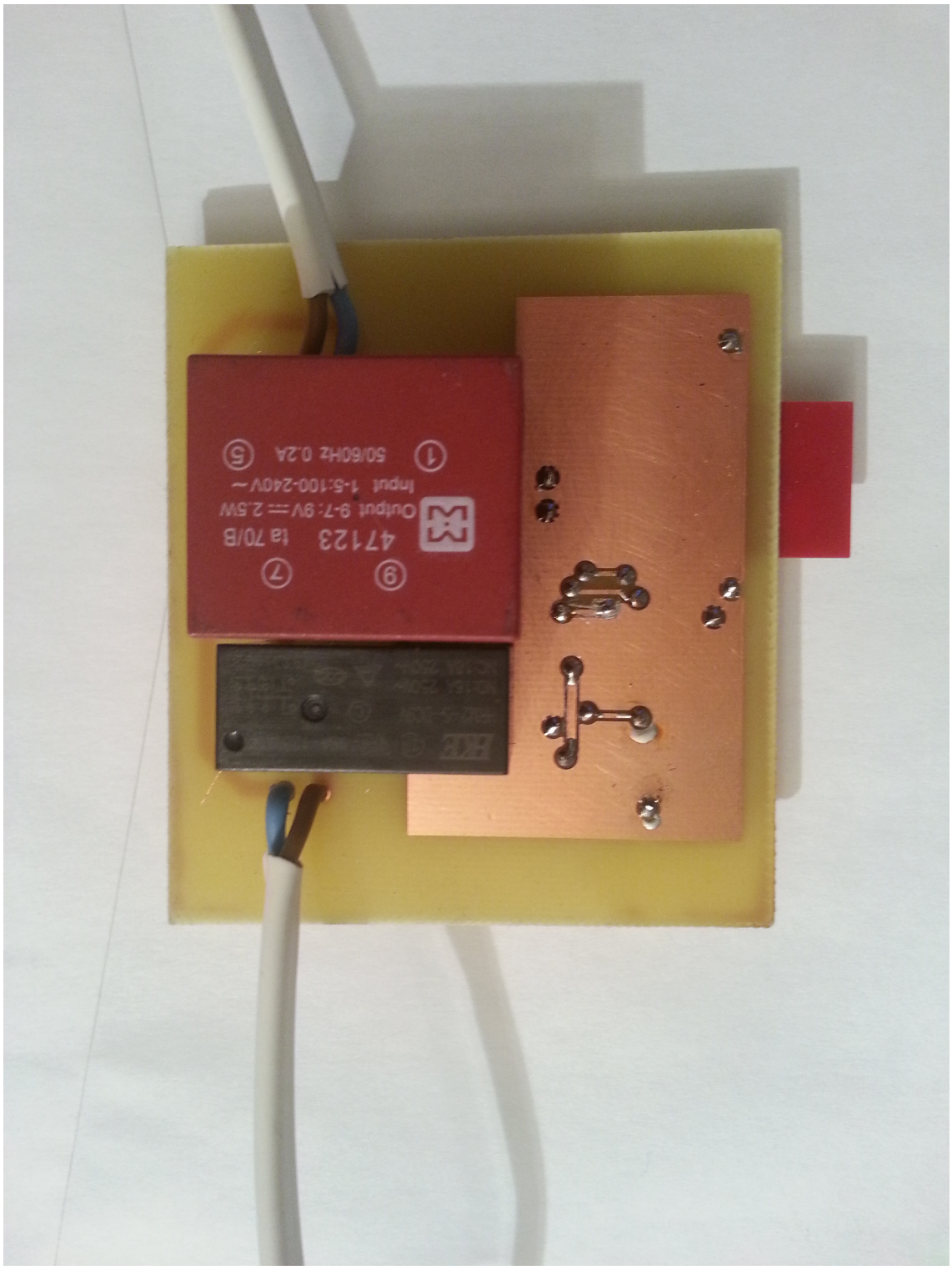
LIITE 9: INFRAPUNAMODUULIN ALAPUOLI



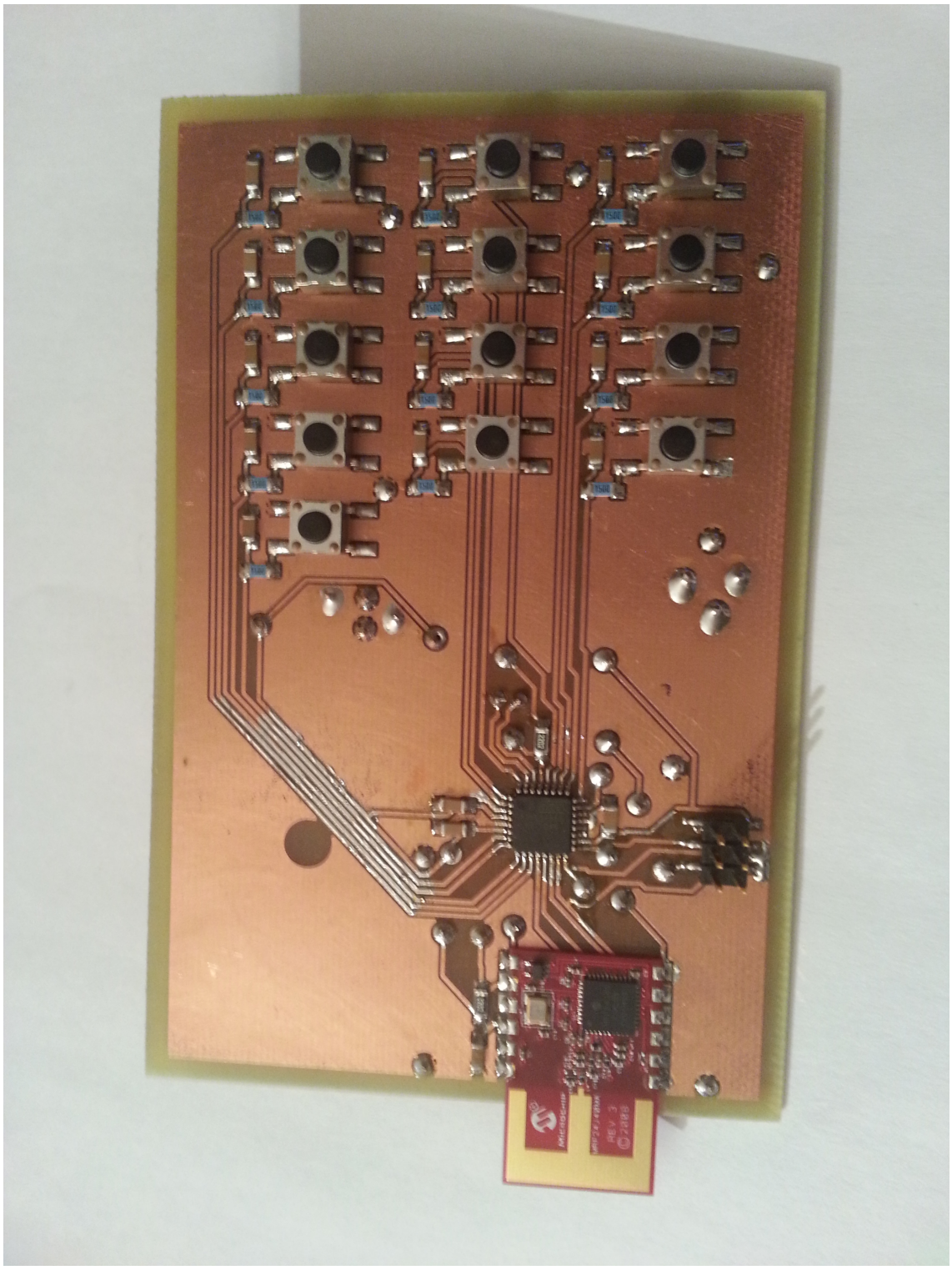
LIITE 10: RELEMODUULIN YLÄPUOLI



LIITE 11: RELEMODUULIN ALAPUOLI

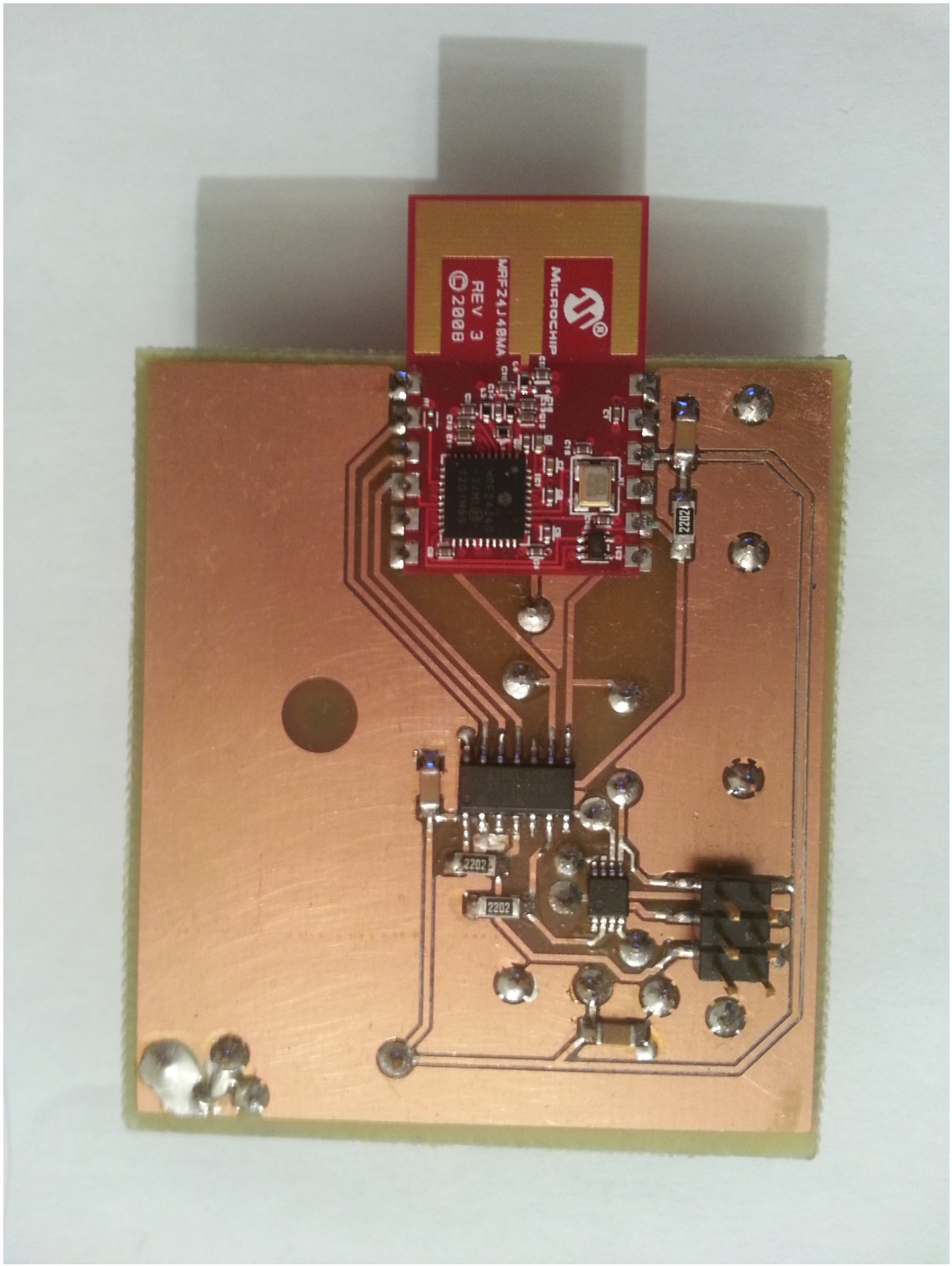


LIITE 12: NÄPPÄIMISTÖMODUULIN YLÄPUOLI



LIITE 13: NÄPPÄIMISTÖMODUULIN ALAPUOLI

LIITE 14: LÄMPÖMITTARIMODUULIN YLÄPUOLI



LIITE 15: LÄMPÖMITTARIMODUULIN ALAPUOLI