TAMPEREEN
AMMATTIKORKEAKOULU

SÄHKÖRULLALAUDAN KAUKO-OHJAIN

Mikael Jokinen

Opinnäytetyö
Huhtikuu 2017
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

JOKINEN, MIKAEL:
Sähkörullalaudan kauko-ohjain

Opinnäytetyö 54 sivua, joista liitteitä 10 sivua
Huhtikuu 2017

Tämän opinnäytetyön tarkoituksena oli suunnitella ja rakentaa prototyyppi sähkörullalaudan kauko-ohjaimesta. Työn tilasi Simo Sihvonen ELMEV-toiminimellään. Sihvonen oli rakentanut sähkörullalaudan prototyypin, jota ohjattiin radio-ohjattavan auton kauko-ohjaimella. Tässä työssä kehitettiin käyttökohteeseen sopivampi ohjain. Sähkörullalaudan kauko-ohjaimessa piti olla mahdollisuus ohjata laudan nopeutta sekä visualisoida laudan ja ohjaimen tilatietoja jollakin tavalla. Ohjaimen suunnitteluun kuului kytkennän suunnittelu, piirilevysuunnittelu, ohjelmakoodin kirjoittaminen sekä kotelon 3D-mallintaminen ja 3D-tulostaminen.

Työn lopputuloksena on prototyyppi, joka sisältää kaikki oleelliset ominaisuudet. Mikroprosessorina on ATmega328p ja radiopiirinä nRF24L01+. Nopeuden ohjaamiseen käytetään potentiometriä. Turvallisuus huomioitiin lisäämällä ohjaimeen kuolleen miehen kytkin, jonka vapautuessa lauta pysähtyy automaattisesti. Kaikki ohjaus- ja turvallisuuslogiikka toteutettiin laudan puolella, ohjain lähettää vain kytkimien ja potentiometrin tilatietoja. Kauko-ohjain sisältää LiPo-akun latausjärjestelmän, ja lataamiseen voi käyttää normaalia puhelimen laturia, jossa on Micro-USB-liitin. Kauko-ohjainta ei testattu varsinaisen laudan kanssa, koska työ tehtiin vuodenaikana, jona laudalla ei voinut ajaa ulkona. Oikean laudan sijaan kauko-ohjaimen toiminta testattiin Arduinolla tehdyn lautaa emuloivan järjestelmän kanssa. Lopputuloksena saatu prototyyppi ei ole hiottu tai optimoitu versio vaan lähtökohta, josta kehitystyö voi alkaa.

Opinnäytetyössä opittiin huomattavasti piirilevysuunnittelusta ja 3D-tulostamisesta. Työ oli myös harvinaislaatuinen tilaisuus työskennellä näin moniulotteisen projektin kanssa, jossa toteutettiin kaikki kotelosta piirilevyyn ja ohjelmakoodiin. Kokonaisuuden kanssa työskentely antoi hyvää näkemystä laajoista projektitöistä. Ohjainta tullaan jatkokehittämään vielä tämän opinnäytetyön jälkeen vapaa-ajalla. Piirilevyssä on vielä parantamisen varaa: Piirilevyä voi vielä pienentää, ja siihen voi samalla lisätä ominaisuuksia ja parantaa komponenttien sijoittelua. Eniten jatkokehitettävää on kuitenkin kotelossa, jota tulisi pienentää ja josta tulisi tehdä ergonomisempi.

Asiasanat: sähkörullalauta, sulautetut järjestelmät, kauko-ohjain

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree programme in ICT Engineering
Embedded Systems and Electronics

JOKINEN, MIKAEL:
Electric Skateboard Remote Controller

Bachelor's thesis 54 pages, appendices 10 pages
April 2017

The purpose of this thesis was to design and build a prototype of an electric skateboard wireless remote controller. The remote controller was ordered by Simo Sihvonen with his trade name ELMEV. Sihvonen had developed and built a prototype of an electric skateboard which used a R/C car remote. The R/C car remote was easy to implement at the time but it had some drawbacks: It was bulky and very limited on features. This project focused on designing a more appropriate remote controller. This project included electronics design, PCB design, 3D modeling and 3D printing.

The result of this project is a working prototype that includes all the required features. The controller uses an ATmega328p microcontroller, an nRF24L01+ radio module, a potentiometer and some switches as inputs. Safety concerns were addressed by using a dead man's switch to automatically stop the board if the switch is released. All the safety and control logic is done on the board's side; the controller only sends data about potentiometer and switch states. The remote controller includes a LiPo battery management system. The battery can be charged with a normal Micro-USB phone charger. The remote controller was not tested with the actual board, because the project was done in a season that prevents riding the board outside. The remote controller was tested with an Arduino based board emulator. The resulting remote prototype is neither polished nor optimized, but it does what a first prototype is supposed to do: It enables testing of the concept, and future improved prototypes can be based on it.

A lot was learned about PCB design and 3D printing during this project. The project was also a unique opportunity to work at such a diverse project, which included everything from PCB design to 3D modeling the frame. Working with such a diverse project from start to finish will make planning of future projects easier. The remote controller will be improved in the future on free time. The PCB has several things that could be improved and some features could be added. The controller frame has the most things to improve on: it should be made smaller and more ergonomic.

Key words: electric skateboard, embedded systems, remote controller

SISÄLLYS

LYHENTEET JA TERMIT

DIP	Dual In-Line Package, suorakulmainen läpijuotettava elektroniikkakomponenttien kotelotyyppi
DRC	Design Rule Check, suunnittelusääntötarkastus
DSSS	Direct-Sequence Spread Spectrum, suorasekventointi
EIRP	Equivalent Isotropically Radiated Power, ideaalin isotrooppisesti säteilevän lähettimen säteilemä teho
FHSS	Frequency Hopping Spread Spectrum, taajuushyppely
haptinen	Tuntoaistia hyödyntävä teknologia
isotrooppinen	Suunnasta riippumaton
SPI	Serial Periphepal Interface Bus, synkroninen sarjaliikenneväylä lyhyille etäisyyksille
TQFP	Thin Quad Flat Package, pintaliitoskotelotyyppi jossa on ”gull wing” -tyyppiset jalat neljällä sivulla

1 JOHDANTO

Uudet urbaanit etenemismuodot, kuten tasapainoskootterit ja moottoroidut yksipyöräiset, ovat yleistyneet viime vuosina. Yksi näiden kanssa kilpaileva järjestelmä on sähkörullalaudat. Sähkörullalaudat ovat käytännössä rullalautoja, joihin on lisätty sähkömoottori ja sen ohjaamiseen tarvittavat komponentit. Sähkörullalautoja ohjataan yleensä langattomalla kaukosäätimellä. Sähkörullalaudat ovat kehityksen suhteen vielä melko alkuvaiheessa, minkä takia niiden parissa on vielä paljon innovoitavaa ja markkinoille pääsy on suhteellisen helppoa.

Tämän opinnäytetyön aiheena on sähkörullalaudan kauko-ohjaimen prototyypin suunnittelu ja valmistus. Työn tilasi Simo Sihvonen ELMEV-toiminimellään RoadRunner-sähkörullalautaprojektia varten. Prototyypin tarkoituksena ei ole olla viimeistelty, tuotantoon kelpaava ohjain, vaan konseptin testaamiseen riittävä järjestelmä. Kauko-ohjaimen ulkomuodolle ja sisällölle ei oltu asetettu tarkkoja ehtoja, mikä mahdollisti melko vapaat kädet konseptointivaiheessa. Aiheen teki poikkeuksellisen mielenkiintoiseksi sen laajuus; projekti sisälsi kaiken elektroniikkasuunnittelusta kotelon 3D-mallintamiseen. Aihevalintaan vaikutti myös mahdollisuus päästä harjoittelemaan kaksikerrospiirilevyjen suunnittelua, josta ei ollut aiempaa kokemusta.

Työssä käsitellään kaikkia suunnittelu- ja mallinnusprosessin vaiheita. Päähuomio työssä kohdistuu piirilevy- ja elektroniikkasuunnitteluun. Työssä käsitellään myös 3D-mallintamista ja ohjelmointia. Työssä käsitellään myös radiopiirin valintaan, radiotaajuuksien käyttöön ja litiumioniakkujen toimintaan liittyviä asioita. Radiotaajuuksien käyttö on vahvasti säänneltyä, minkä takia radiotaajuuksia hyödyntävää elektroniikkaa suunnitellessa täytyy perehtyä lainsäädäntöön. Litiumioniakkujen käyttöön liittyy useita turvallisuuteen ja akun käyttöikään vaikuttavia asioita, jotka täytyvät olla tiedossa suunnitteluvaiheessa.

2 SÄHKÖRULLALAUDAT

Sähkörullalauta on pitkä rullalauta, jossa on yksi tai useampi sähkömoottori työntövoiman tuottamiseksi. Sähkörullalautailu on uusi etenemistapa, joka kilpailee leijulautojen ja sähköpolkupyörien kanssa. Se on käytännöllinen etenemismuoto kaupunkialueilla, joilla on hyväkuntoiset tiet, sillä sähkörullalauta on keveytensä ja pienen kokonsa vuoksi helppo ottaa mukaan bussiin tai töihin sisälle. Suurin osa sähkörullalaudoista ei sovellu sorateille eikä maastoon pienten pyöriensä ja pienen maavaransa takia.

Markkinoilla olevien sähkörullalautojen akustojen varaus riittää yleensä 20–30 km matkalle ja niiden maksiminopeus vaihtelee välillä 15–30 km/h. Hintataso, jolla saa hyvälaatuisia sähkörullalautoja, on tällä hetkellä 1000–2000 euroa.

2.1 Rakenne ja toiminta

Sähkörullalauta koostuu käytännössä normaalista pitkästä rullalaudasta, johon on lisätty akusto, moottorinohjausjärjestelmä, radiovastaanotin, sähkömoottori ja voimansiirto. Voimansiirto voidaan toteuttaa yhdellä tai useammalla trukkiin kiinnitetyillä moottorilla, joiden kanssa käytetään hihnaa voimansiirtoon pyöriin. Vaihtoehtoinen tapa on käyttää pyörien sisässä olevia napamoottoreita, jolloin voimansiirtoon ei tarvita ylimääräisiä osia. Napamoottoreiden huono puoli on se, että silloin on yleensä sidoksissa napamoottorin valmistajan tekemiin renkaisiin. Napamoottorit myös kuluvat käytössä enemmän kuin hihnavedossa käytettävät moottorit.

Sähkörullalaudan työntövoimaa ohjataan yleensä yhdessä kädessä pidettävällä kauko-ohjaimella. Sähkörullalaudalla käännytään kallistamalla lautaa kuten normaalillakin rullalaudalla. Normaalilla rullalaudalla jarruttaminen on melko vaikeaa: suuremmissa nopeuksissa se tehdään yleensä joko painamalla toista jalkaa asfalttiin (kengät kuluvat nopeasti) tai liukumalla laudalla poikittain (vaikeaa). Sähkörullalaudalla voidaan käyttää sähkömoottoria apuna jarrutuksessa, mikä on erittäin aloittelijaystävällistä. Lisäksi sähkömoottorilla jarrutettaessa on mahdollista ottaa jarrutusenergia talteen akustoon, mikä parantaa toimintamatkaa. Jarrutusenergian talteenotto hyödyttää luonnollisesti eniten maastossa, jossa on paljon jarrutuksia.

2.2 Kauko-ohjain

Sähkörullalaudan kauko-ohjain on kokonaisuus, joka muodostuu rungosta kytkimineen ja mekaniikkoineen, akusta ja sen laturista sekä muusta elektroniikasta ja radiopiiristä. Kauko-ohjaimen tärkein tehtävä on luonnollisesti välittää kontrollidataa laudalle, mutta ohjaimella on myös hyvä pystyä indikoimaan esimerkiksi laudan ja ohjaimen akun tilaa sekä radiosignaalin tasoa.

Muita loppukäyttäjän kannalta oleellisia asioita ovat akunkesto, ergonomisuus ja käyttöliittymä. Ohjaimen akun pitäisi mielellään kestää useita kertoja pidempään kuin laudan akun. Ohjaimen pitää tuntua hyvältä kädessä pidemmilläkin lenkeillä ja sopia erikokoisiin käsiin. Laudan ohjaamisen kuuluu tuntua luonnolliselta ja onnistua myös hanskat kädessä. Tärinä ei saa vaikuttaa ohjaimen toimintaan, eikä ohjain saa lipsua tärähdyksissä helposti kädestä.

Kauko-ohjaimen suunnittelussa tulee ottaa huomioon myös turvallisuusnäkökulma. Ongelmatilanteissa tärkeintä on laudan hallittu pysähtyminen ja lautailijan informointi esimerkiksi äänimerkillä tai haptisella palautteella ennen kuin automaattinen jarrutus alkaa. Vähintään taulukossa 1 listatut ongelmatilanteet pitää pystyä hallitsemaan.

TAULUKKO 1. Turvallisuuden kannalta huomioitavia tilanteita

Tilanne	Vaikutus
Ohjain putoaa ajossa lautailijan kädestä alamäessä, lautailija jää laudalle	Jos lauta ei ala jarruttaa automaattisesti ja lautailija ei ole tottunut jarruttamaan ilman sähkömoottoria, vauhti kiihtyy vaarallisen suureksi.
Lautailija putoaa laudalta, ohjain putoaa samalla kädestä	Laudan pitää alkaa jarruttaa automaattisesti, jotta lauta ei karkaa tai törmää johonkin.
Yhteys laudan ja ohjaimen välillä katkeaa	Yhteys voi katketa esimerkiksi kaupungin keskustassa, kun lautailija aikoo jarruttaa risteykseen. Törmäysriskin minimoimiseksi on jarrutettava automaattisesti mutta ei niin äkillisesti, että lautailija kaatuu.

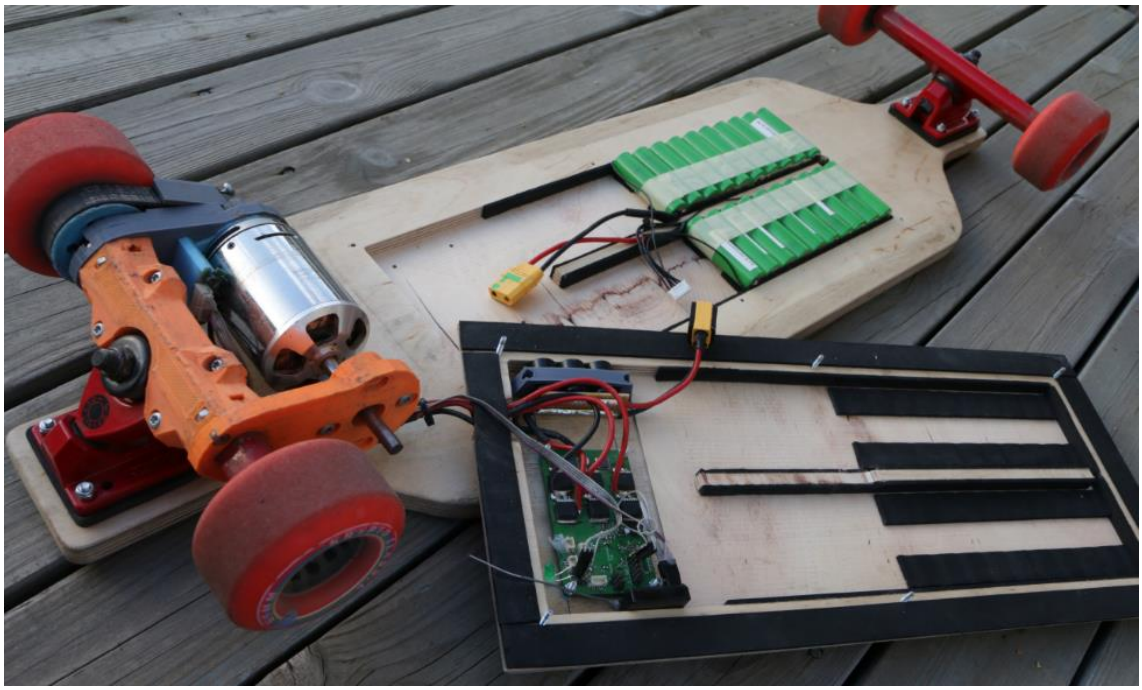
Laudan akku loppuu kesken ajon	Jos laudan akkua ei tyhjennetä aivan kokonaan, voidaan pitää radioyhteys päällä ja lauta vapaalla (eteneminen potkimalla) mutta sallia sähkömoottorin käyttö jarrutuksiin, koska se ei kuluta akun virtaa.
Ohjaimen akku loppuu kesken ajon	Ohjaimen akun varauksen loppumisessa on kaksi puolta: lauta pitäisi saada pysähtymään, mutta jos jarru jätetään päälle, niin lautaa ei voi käyttää potkimallakaan. Yksi ratkaisu olisi pysäyttää lauta sähkömoottorilla ja päästää lauta vapaalle viiveen jälkeen.

2.3 RoadRunner-projekti

Tässä opinnäytetyössä tehtävän kauko-ohjaimen tilasi Simo Sihvonen RoadRunner-sähkörullalautaprojektiaan varten. RoadRunner-laudalla (kuva 1) ei ole tällä hetkellä kaupallisia tavoitteita, vaan se on harrasteprojekti. Kun projekti valmistuu, järjestelmän kaupallisia mahdollisuuksia pohditaan uudelleen. Laudan maksiminopeus oli opinnäytetyön kirjoittamishetkellä 25 km/h ja toimintamatka 24 km. Akkukonfiguraatio oli 6S4P (6 kennoa sarjassa, 4 rinnan). Aiemmin käytössä olleella 4S5P-akkukonfiguraatiolla maksiminopeus oli 18 km/h ja toimintamatka 31 km. Molemmissa akkukonfiguraatioissa oli käytössä 3 500 mAh:n 18650 Li-ion-kennot. (Sihvonen 2017).

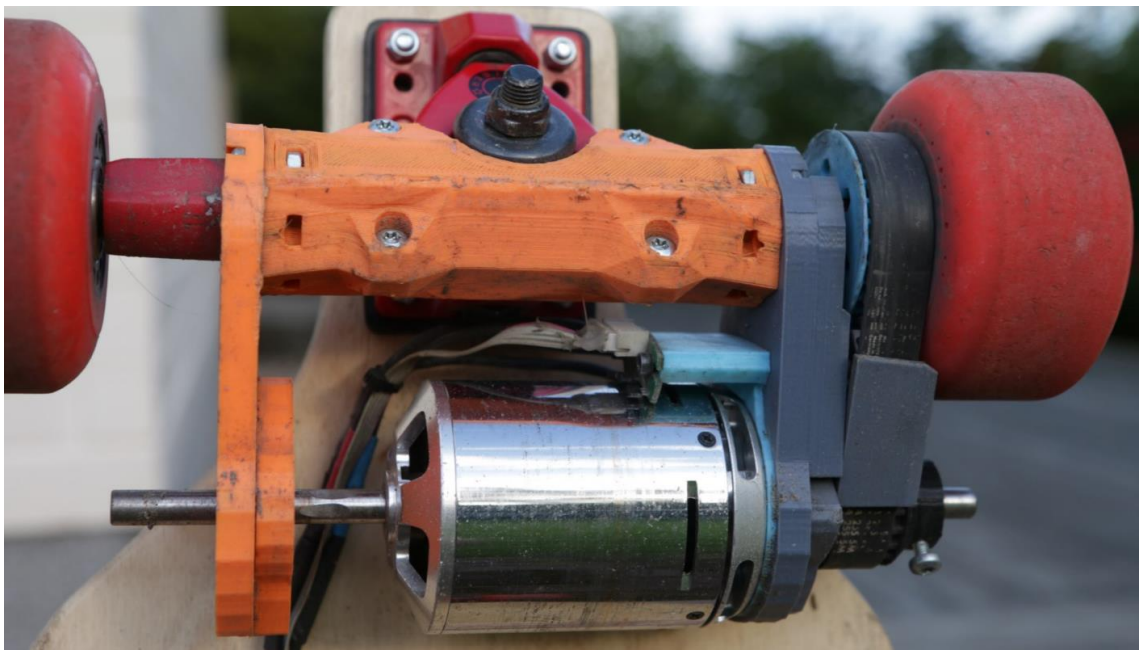
KUVA 1. RoadRunner-lauta (Sihvonen 2016)

Kuvassa 2 näkyy RoadRunner-laudan rakenne. Vihreät akkupaketit ja moottorinohjaus-järjestelmä sekä radiovastaanotin ovat laudan sisällä. Laudan sisälle mahtuu enemmän litiumionikennoja kuin kuvassa on kytkettynä.



KUVA 2. RoadRunner-laudan rakenne (Sihvonen 2016)

Moottori sekä hihnaveto on rakennettu takatrukin yhteyteen (kuva 3). Moottorin ja hihnan kiinnittämisessä on käytetty 3D-tulostettuja osia. Laudan moottori on 2250 W:n tehoinen hiiliharjaton Turnigyn SK3 AeroDrive 6374-192kv.



KUVA 3. RoadRunner-laudan voimansiirto (Sihvonen 2016)

RoadRunner-laudan ensimmäisessä versiossa käytettiin radio-ohjattavan pikkuauton kauko-ohjainta (Hobbykingin myymä *Quantum 2.4 GHz 3ch Pistol Grip Tx & Rx system*, kuva 4). Vastaanottimelta tuleva signaali oli R/C-autoille tyypilliseen tapaan servo-ohjaukseen tarkoitettua pulssinleveysmoduloitua signaalia. RoadRunner-lauta ei kuitenkaan voi ohjata moottoria suoraan servokäyttöön tarkoitetulla pulssinleveysmoduloidulla signaalilla, joten signaali piti purkaa digitaaliseksi.

R/C-auton kauko-ohjaimen käyttäminen oli kehityksen alkuvaiheessa hyvä ratkaisu, koska se oli edullinen (23,70 $), nopeasti käyttöönotettavissa ja ohjaimen ja vastaanottimen välinen yhteys oli luotettava. Huonoja puolia ratkaisussa olivat kuitenkin suuri koko (250 x 140 x 50 mm), vain yksisuuntainen dataliikenne ja vain kolmekanavainen analoginen ulostulo. Yksisuuntaisen liikenteen takia ohjaimella ei voinut näyttää mitään tietoja laudan tilasta. Ohjaimen radioliikenteen mahdollisesti käyttämästä taajuus/kanavahyppelystä ja modulaatiosta ei myöskään annettu mitään tietoa. Ohjaimen avulla ei myöskään pystytty tekemään mitään turvallisuusominaisuuksia esimerkiksi ohjaimen kädestä putoamisen varalta.

KUVA 4. Alkuperäinen kauko-ohjain

3 TAUSTATIEDOT

3.1 Radio-ohjaus

Radio-ohjauksessa on otettava huomioon useita tekijöitä, kun halutaan muodostaa luotettava yhteys. Huomioon on otettava lähettimen ja vastaanottimen etäisyys, antennien välinen kulma ja mahdollisesti välissä olevat esteet sekä mahdolliset häiriöt.

Lähetin on tässä projektissa sähkörullalaudan ohjaajan kädessä ja vastaanotin laudan sisällä. Lähettimen ja vastaanottimen väliseksi etäisyydeksi tulee käytännössä noin metri. Suurimmillaan etäisyys on silloin, kun ohjaaja pitää ohjainta päänsä päällä. Lyhyiden etäisyyksien ja välissä olevien esteiden takia vapaan tilan vaimennusta ei ollut hyödyllistä laskea eikä laudan ja muiden tekijöiden aiheuttamia vaimennuksia voinut helposti ennakoida.

Tilapäisten ulkopuolisten radiohäiriöiden aiheuttamat ongelmat voi ratkaista yksinkertaisesti lähettämällä paketin uudelleen muutaman kerran. Pidempiaikaisiin kapeakaistaisiin häiriöihin käytetään yleisesti kanavahyppelyä (FHSS) tai suorasekventointia (DSSS).

Kanavahyppelyssä lähetin ja vastaanotin vaihtavat jatkuvasti kanavaa, jolla radioliikenne tapahtuu. Jos lähetys epäonnistuu jollakin kanavalla kapeakaistaisen häiriön takia, se todennäköisesti onnistuu jollain toisella sekvenssin kanavista. Kanavahyppely sopii myös hyvin käyttöön tilanteissa, joissa usean radiolähettimen pitää jakaa sama taajuusalue. Vaikka taajuusalueella toimisikin useampi lähetin ja ne sattuisivat lähettämään samaan aikaan samalla taajuudella, todennäköisyys, että niiden sekvenssissä seuraavakin kanava olisi sama, on äärimmäisen pieni.

Suorasekventoinnissa käytetään huomattavasti kantataajuista signaalia korkeataajuisempaa signaalia moduloidun signaalin muodostamiseen. DSSS:llä voidaan usein parantaa vikasietoisuutta, mutta joissain tapauksissa DSSS vain huonontaa yhteyden laatua.

Antenneilla on jokin antennivahvistus. Antennivahvistus ei kuitenkaan tarkoita, että signaalin lähetysteho kasvaisi antennissa, vaan että signaali suuntautuu tietylle alueelle vahvempana ja muualle heikompana. Suuri antennivahvistus on siis hyvä käyttökohteissa,

joissa lähettimen ja vastaanottimen antennit pysyvät paikallaan (tai niiden voidaan varmistaa olevan jatkuvasti suunnattuna oikein toistensa suhteen). Käyttökohteissa, joissa ei voida ennakoida lähettimen ja vastaanottimen antennien suuntia suhteessa toisiinsa, kuten sähkörullalaudan kauko-ohjaimessa, antennivahvistuksesta on enemmän haittaa kuin hyötyä.

3.1.1 Radiolaki

Viestintävirasto ohjaa radiotaajuuksien käyttöä Suomessa. Taajuusalue 9 kHz–3000 GHz on kansallisesti ja kansainvälisesti säänneltyä aluetta (Radiotaajuudet ja niiden käyttö 2017).

Radiotaajuuksien käyttöön säännellyllä alueella ei tarvita radiolupaa, jos radiolaitteen käyttötarkoitus on luvasta vapautettu. Sähkörullalaudan kauko-ohjain luokitellaan viestintäviraston määräyksen 15AK pykälän § 10 mukaiseen kategoriaan *yleiset lyhyen kantaman radiolähettimet*, jotka on vapautettu radioluvasta tietyin edellytyksin. Viestintäviraston määräyksessä on listattu yleisille lyhyen kantaman radiolähettimille sallitut taajuudet ja lähetystehot. Yksi yleisten lyhyen kantaman radiolähettimien taajuusalue on 2,400–2,4835 GHz efektiivisen lähetystehon ollessa alle 10 mW EIRP. (Määräys luvasta vapaiden radiolähettimien yleistaajuuksista ja käytöstä 2016.)

Rikosnimike kyseistä lakia rikottaessa on tietoliikenteen häirintä, josta voidaan tuomita sakkoon tai vankeuteen enintään kahdeksi vuodeksi. Jos asiaton radioliikenne häiritsee hätäkutsuja tai muuta yhteiskunnallisesti tärkeää toimintaa, rikosnimike on törkeä tietoliikenteen häirintä, josta voidaan tuomita vähintään neljäksi kuukaudeksi ja enintään viideksi vuodeksi vankeuteen. (Rikoslaki 1995.)

Laki huomioidaan käytännössä valvomalla, että pysytään sallitulla taajuusalueella eikä lähetetä liian suurella teholla. Huomionarvoista on se, että säännöt eivät määrää antennille menevää tehoa, vaan antennin säteilykeilan maksimin tehotiheyttä. Koska tehotiheyden maksimiarvo ei saa ylittää vaatimuksen mukaista rajaa missään säteilysuunnassa, on antennivahvistus ja laitteen rakenteesta johtuva suuntaavuus otettava huomioon. Radiotaajuuksien käyttöä käsittelevät säännöt ja standardit löytyvät tarkemmin selostettuina viestintäviraston määräyksen M4V liitteestä *Taajuusjakotaulukko*.

3.1.2 Radiopiirin valitseminen

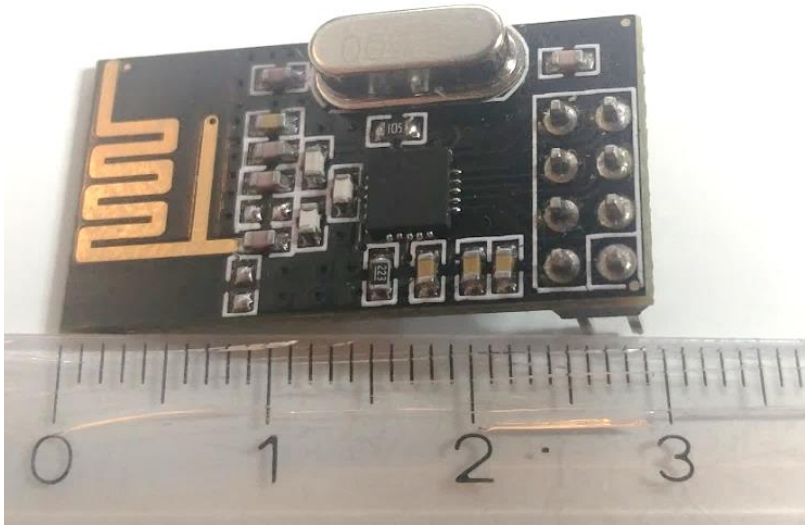
Kauko-ohjaimessa ajateltiin alun perin käyttää Bluetooth-pohjaista radiopiiriä langattomaan tiedonsiirtoon, mutta Bluetoothin luotettavuus herätti kysymyksiä. Bluetoothin kantama on tyypillisesti vain muutamia metrejä, vaikka Bluetooth-standardi sallii pidemmätkin kantamat. Omassa käytössä olleilla Bluetooth-laitteilla on ollut luotettavuusongelmia, ja internetistä löytyy useita videoita, joissa ihmiset törmäilevät Bluetoothia käyttävillä sähkörullalaudoilla yhteyden katketessa. Projektissa ei ollut myöskään erityistä tarvetta tukea Bluetoothia, koska kauko-ohjaimeen ei haluttu liittää muita laitteita kuin RoadRunner-lauta.

Bluetoothia parempana vaihtoehtona pidettiin Nordic Semiconductorsin nRF24L01+-radiopiiriä. NRF-moduulien kantama on useita kymmeniä metrejä piirilevyllä olevalla antennilla ja jopa yli sata metriä paremmalla antennilla. Kauko-ohjaimeen ei varsinaisesti tarvittu pitkää kantamaa, mutta yhteyden on oltava luotettava.

Yhtenä vaihtoehtona oli myös 433 / 868 / 915 MHz:n taajuuksilla toimiva nRF905, joka olisi mahdollistanut paremman läpäisykyvyn. Jos projektissa oltaisiin päädytty Bluetoothiin, moduulina olisi voinut olla nRF51. NRF24L01+ valittiin testattavaksi ensimmäisenä, koska se oli helpoiten saatavilla ja sille oli parhaat tukimateriaalit. NRF24L01+:lle on olemassa valmis Arduino-kirjasto, ja nRF:ää on käytetty myös TAMKin laboratoriotyökursseilla.

3.1.3 NRF24L01+-radiomoduuli

Työssä käytetty nRF24L01+-radiopiiri oli tilattavissa internetistä valmiina moduulina, joka sisälsi piirin tarvitseman kiteen, passiivikomponentit, antennin ja liittimen (kuva 5). Kiinasta tilattuna yksi moduuli maksoi alle euron. Radiopiirin oleellisimmat tiedot on koottu taulukkoon 2.



KUVA 5. Radiomoduuli

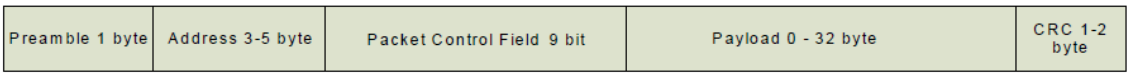
TAULUKKO 2. Radiopiirin tietoja (Nordic Semiconductors 2008, muokattu)

Taajuusalue	2,400–2,525 GHz
Käyttöjännite (sisääntulevat signaalit ≤ 3,6 V)	1,9–3,6 V
Käyttöjännite (sisääntulevat signaalit > 3,6 V)	2,7–3,3 V
Maksimijännite sisääntuloissa	5,25 V
Lähetysteho (valittavissa)	-18 dBm, -12 dBm, -6 dBm, 0 dBm
Virrankulutus aktiivisena	9,0–13,5 mA
Virrankulutus lepotilassa	26–320 µA
Virrankulutus poissa päältä	900 nA
Datansiirtonopeudet	250 kbps, 1 Mbps, 2 Mbps
Kaistanleveys (20 dB)	800 kHz, 1000 kHz, 2000 kHz
RX-herkkyys	-94 dBm, -85 dBm, -82 dBm

NRF:n käyttämä taajuus on valittavissa 126:sta 1 MHz:n välein olevilta kanavilta väliltä 2,400–2,525 GHz, ja radiopiirin 20 dB kaistanleveys on maksimissaan 2 MHz. Radiopiiri käyttää GFSK-modulaatiota. Radiopiirin käyttämiä kanavia täytyi rajoittaa, jotta pysyttiin laillisella alueella (2,400–2,4835 GHz). Radiopiirin ohjaamiseen käytettiin itse kirjoitettua C-kirjastoa, jossa käytettävät taajuudet oli rajattu välille 2,401–2,482 GHz. Taajuuksien rajaamisessa oli otettava huomioon lähetteen kaistanleveys. Radiomoduulin piirilevyllä oli *meandered inverted F* -tyyppinen antenni. (Nordic Semiconductors 2008.)

Radiomoduulille annetaan komentoja asettamalla piirin CSN alas ja antamalla ensin komentotavu SPI-väylällä, minkä jälkeen kirjoitetaan/luetaan haluttu määrä tavuja. Kun datansiirto on suoritettu, CSN asetetaan takaisin ylös. Radion toimintaa ohjataan erikseen CE-sisääntulolla. Moduulilla voidaan lähettää paketti nostamalla CE kymmeneksi mikrosekunniksi ylös, kun TX-payload rekisteri on täytetty. Jos moduulia halutaan käyttää kuuntelutilassa, CE pidetään koko kuuntelun ajan ylhäällä. Radiopiiri tukee maksimissaan 5,25 V:n sisääntulosignaaleja, mutta yli 3,6 V:n sisääntulojännitteillä sallittu käyttöjännitealue on rajoitetumpi. (Nordic Semiconductors 2008.)

NRF:n kanssa on mahdollista käyttää Nordic Semiconductorsin Enhanced ShockBurst -kehysrakennetta (kuva 6). Enhanced ShockBurst mahdollistaa automaattikuittaukset, automaattisen CRC-virheentarkistuksen ja dynaamisen tavumäärän. Enhanced ShockBurst -kehysrakenne on esitetty kuvassa 6. Preamble eli tahdistus on 01010101, jos osoitteen ensimmäinen bitti on 0, ja 10101010, jos osoitteen ensimmäinen bitti on 1. Packet Control Field sisältää tiedon paketin tavumäärästä (6 bittiä), paketin tunnistusbitit (2 bittiä) ja automaattisen kuittauksen kieltolipun. (Nordic Semiconductors 2008.)



KUVA 6. Enchanced ShockBurst -kehyksen rakenne (Nordic Semiconductors 2008)

3.2 Litiumioniakut

3.2.1 Terminologia

Litiumioniakkuja on kolmea eri varianttia, jotka on helppo sekoittaa keskenään: litiumioniakut, litiumionipolymeeriakut ja litiumpolymeeriakut. Vanhinta tekniikkaa näistä on litiumioniakku. Erona tekniikoilla on elektrolyyttinä käytetty materiaali: litiumioniakuissa elektrolyytti on nestemäinen, litiumionipolymeeriakuissa geeli ja litiumpolymeeriakuissa kiinteä. Litiumpolymeeriakkuja ei ole saatu käytännössä toimimaan huoneenlämmössä, joten ainoat yleisessä käytössä olevat tekniikat ovat litiumioni ja litiumionipolymeeri. (Battery University 2017.)

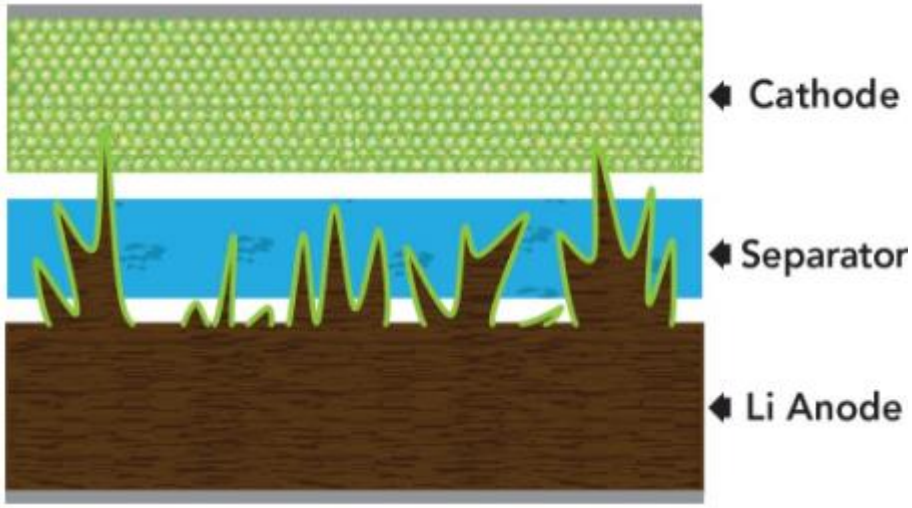
Litiumionipolymeeriakkujen kutsuminen litiumpolymeeriakuiksi on muodostunut yleiseksi käytännöksi. Myös tässä työssä litiumionipolymeeriakkuja kutsutaan litiumpolymeeriakuiksi.

Jokaisella akkutekniikalla on omat etunsa. Litiumioniakut ovat halvimpia valmistaa, mutta niitä voidaan käytännössä tuottaa vain lieriön muotoisina. Litiumpolymeeri- ja litiumionipolymeeriakuilla ei ole muodon suhteen samanlaisia rajoituksia. Litiumionipolymeeriakut voidaan tehdä esimerkiksi luottokortin muotoisiksi ja erittäin ohuiksi. Litiumpolymeeriakut olisivat oletettavasti turvallisempia kuin muut vaihtoehdot, jos tekniikka saataisiin toimimaan huoneenlämmössä. (Battery University 2017.)

3.2.2 Lataaminen

Litiumioniakkujen kapasiteetti ilmoitetaan yleensä milliampeerituntimääränä ja lataus- ja purkunopeudet C-arvolla. C-arvo kertoo, kuinka moninkertaisen lataus- tai purkuvirran akku kestää suhteessa akun kapasiteettiin. 2000 mAh:n akulla 1C tarkoittaisi 2 A:n virtaa, ja 3000 mAh:n akulla 2C tarkoittaisi 6 A:n virtaa.

Litiumioniakkuja ei saa yliladata eikä purkaa täysin tyhjiksi. Useimmat akkuvalmistajat kertovat alarajaksi 3,0 V ja ylärajaksi 4,2 V. Ylilataaminen ja akun täysin purkaminen aiheuttavat kemiallisia reaktioita, jotka aiheuttavat kapasiteetin laskua ja dendriittien muodostumista. Dendriitit ovat turvallisuusriski, sillä ne voivat kasvaa niin suuriksi, että ne voivat aiheuttaa oikosulun (kuva 7), joka voi johtaa akun syttymiseen palamaan. (Propwashed 2016.)



KUVA 7. Dendriittien aiheuttama oikosulku (SLAC National Accelerator Laboratory 2015)

3.3 3D-tulostaminen

3D-tulostimen käyttömahdollisuus tekee kauko-ohjaimen muovisen kotelon kehittämisestä huomattavasti nopeampaa, helpompaa ja halvempaa kuin se olisi muuten. Käteen sopivan noin 115 x 35 x 20 mm:n kokoisen kotelon tulostamiseen kuluu 1–2 tuntia. Tulostukseen kuluva aika ei vaadi aktiivista osallistumista tulostajalta, vaan samalla voi tehdä muita töitä, kunhan silloin tällöin tarkistaa, ettei tulostuksessa ole ongelmia. Edellä mainitun kokoisen kotelon tulostamisessa kuluu noin 3 metriä filamenttia, jonka halkaisija on 3 mm. Filamentin hinnan ollessa tällä hetkellä 20 € / 100 m yhden kauko-ohjaimen kotelon tarvitseman materiaalin hinnaksi tulee noin 60 senttiä.

3D-tulostimen käyttämisessä on myös se etu, että kotelon tekeminen vaatii ainoastaan osaamista 3D-mallintamisesta ja 3D-tulostimen käytöstä. Kotelon valmistamisessa ei tarvita kädentaitoja, koska 3D-malliin voi sisällyttää kaikki tarvittavat reiät ja komponenttien kiinnikkeet. Hyvällä 3D-tulostimella tulostettuja kappaleita ei tarvitse viimeistellä käsin, mutta huonolla tai väärin säädetyllä tulostimella tulostettuja kappaleita voi joutua poraamaan tai hiomaan käsin.

Prototyyppien 3D-tulostamisessa käytetään maissista valmistettua PLA-muovia, jota markkinoidaan biohajoavana. PLA-muovi hajoaa kuudessa kuukaudessa teollisessa kompostointilaitoksessa ja vaatii tarkat olosuhteet hajoamiseen. PLA-muovin hajoamisesta kotikompostorissa tai luonnossa ei löydy selkeää tietoa. PLA-muovia voitaneen kuitenkin

pitää ympäristön kannalta parempana kuin öljypohjaisia muoveja, joiden hajoamiseen menee sadoista vuosista tuhansiin vuosiin. PLA:sta ei myöskään vapaudu haitallisia aineita sen hajotessa toisin kuin öljypohjaisista muoveista.

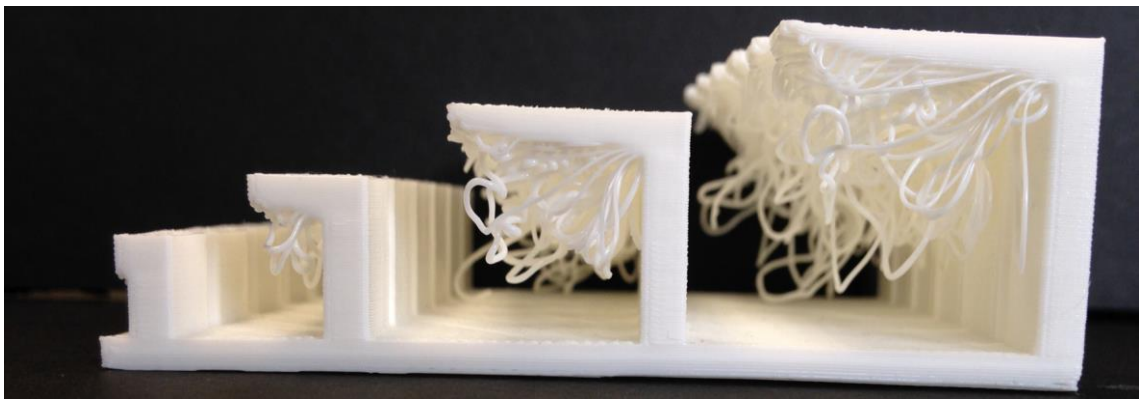
3.4 3D-mallintaminen

3D-mallintamiseen on saatavilla useita erilaisia ohjelmia. 3D-mallintamisohjelmat voidaan jakaa parametrisiin ja ei-parametrisiin. Parametrinen mallintaminen sopii hyvin tekniseen mallintamiseen, kun taas ei-parametrinen mallintaminen sopii hyvin pikaiseen hahmotteluun ja konseptointiin, jossa mitat eivät ole kriittisiä.

Parametrisessa mallintamisessa voidaan viitata toisiin mallin osiin ja määrittää esimerkiksi, että osan B pitää olla 30 mm leveämpi kuin osan A. Jos osan A leveyttä muutetaan mallinnuksen myöhemmässä vaiheessa, osa B mukautuu automaattisesti uusiin mittoihin.

Ei-parametrinen mallintaminen on nopeampaa kuin parametrinen. Ei-parametrisessa mallintamisessa mitään mittoja ei tarvitse sitoa toisiinsa, mikä nopeuttaa suunnittelua. Lisäksi ei-parametrinen mallintaminen vaatii vähemmän laskentatehoa.

3D-tulostaminen on otettava huomioon kotelon mallintamisessa. 3D-tulostuksessa on epätarkkuutta, johon täytyy varautua 3D-mallin kanssa. Esimerkiksi ruuvien kiinnitysreikien kannattaa olla noin 0,5 mm suurempia kuin käytettävän ruuvin halkaisija, jotta reikää ei tarvitse hioa suuremmaksi. 3D-mallissa täytyy pitää huoli, että malliin ei tule pitkiä kielekkeitä. 3D-tulostin joutuu kielekkeissä tulostamaan tyhjän päälle, mikä aiheuttaa ongelmia pidemmillä matkoilla (kuva 8). Joissakin tapauksissa kielekkeiden muodostuminen voidaan estää kääntämällä kappale toisin päin. Esimerkiksi T-kirjaimen tulostaminen oikein päin on vaikeaa, mutta väärin päin käännettynä siinä ei ole ollenkaan kielekkeitä.



KUVA 8. Epäonnistuneet kielekkeet 3D-tulostuksessa (Cockrell School of Engineering 2014)

3.5 Piirilevy- ja elektroniikkasuunnittelu

Kauko-ohjaimen piirilevyn ja kytkentäkaavion suunnitteluun käytettiin Autodeskin Eagle-suunnitteluohjelmistoa. Eaglesta on saatavilla ilmainen versio opiskelijoille ja yrityksille, joiden liikevaihto on alle 100 000 $ vuodessa. Ilmaista versiota käytettäessä kytkentäkaavion koko on rajattu kahteen sivuun, piirilevyn signaalikerrokset kahteen ja piirilevyn pinta-ala 80 cm^2:iin. Kyseiset rajoitukset eivät haitanneet kauko-ohjaimen suunnittelua, sillä kytkentä oli melko yksinkertainen ja piirilevyn oli oltava mahdollisimman pieni, mieluiten alle 15 cm^2, jottei se olisi haitannut kotelon suunnittelua.

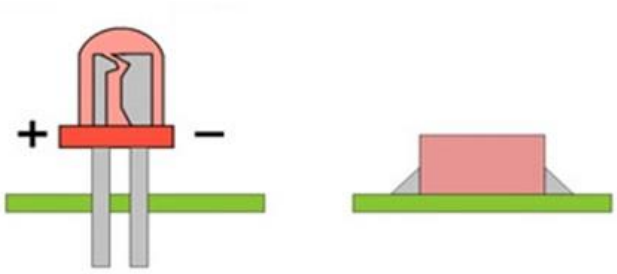
Piirilevyjen teettäminen on edullista ja halpaa, kun käytetään kiinalaisia piirilevypajoja. Piirilevypajalle lähetetään Eaglen generoimat gerber-tiedostot ja valitaan piirilevyn ominaisuuksia, kuten väri ja paksuus. 10 kappaletta 10 x 10 cm:n kaksikerrospiirilevyjä Seedstudiosta maksoi tämän opinnäytetyön kirjoitushetkellä 9,90 $ postikuluineen. Hinta sisälsi kaikki oleelliset palvelut, kuten poraamisen, silkkipainamisen ja piirilevyn jyrsinnän haluttuun muotoon.

Piirilevypajoilla on rajoituksia piirilevyjen ominaisuuksiin. Piirilevypajat pystyvät vain tiettyyn tarkkuuteen ja niillä on vain rajallinen määrä työkaluja käytössään. Piirilevyvetojen paksuuksiin ja välistyksiin on olemassa määritykset, ja käytettävien poranterien on oltava saatavilla. Useimmat piirilevypajat tarjoavat Eagleen drc-tiedoston, joka sisältää piirilevypajan suunnittelusäännöt. Drc-tiedoston avulla Eagle voi tarkistaa piirilevyn noudattavan annettuja suunnittelusääntöjä.

3.6 Komponenttivalintojen vaikutus piirilevysuunnitteluun

Elektroniikkakomponentteja valittaessa piti ottaa huomioon elektronisten asioiden lisäksi myös piirilevysuunnittelu. Koska kauko-ohjaimen oli tärkeää olla mahdollisimman pieni, käytettyjen komponenttien kuului viedä mahdollisimman vähän pinta-alaa piirilevyltä. Koska käytössä ei ollut mikroskooppia, kuumailmapuhallinta tai piirinvaihtokonetta, komponenttien täytyi olla helposti juotettavia. Käytännössä se tarkoitti, että pintaliitosvastuksissa pienin soveltuva koko oli 0603 (tuumakoko) ja prosessorin koteloksi sopi TQFP.

Vaikka läpijuotettavat komponentit ovatkin helpompia juottaa, ne eivät sovellu hyvin ahtaalle piirilevylle. Vaikka itse komponentin tarvitsema pinta-ala ei olisikaan merkittävästi suurempi kuin komponentin pintaliitosvastineella, läpijuotettava komponentti estää vetojen tekemisen ja komponenttien sijoittamisen vastapuolelle (kuva 9).



KUVA 9. Läpijuotettava vs. pintaliitos (New Star Vision)

Useimmille digitaalikomponenteille suositellaan laitettavaksi kytkentäkondensaattori jokaiseen käyttöjännitenastaan. Kytkentäkondensaattori suojaa sekä komponenttia muun kytkennän häiriöiltä, että muuta kytkentää komponentin tuottamilta häiriöiltä. Kytkentäkondensaattori kuuluu sijoittaa mahdollisimman lähelle käyttöjännitetuloa mahdollisimman lyhyillä ja leveillä kuparivedoilla. Helppo tapa pintaliitoskytkentäkondensaattorien sijoitteluun on laittaa ne vastakkaiselle puolelle suodatettavia virransyöttötuloja.

3.7 Hall-sensori

Ohjaimessa tarvitaan jokin asentoa mittaava sensori, jotta työntövoimaa voidaan säätää portaattomasti. Potentiometri on erittäin helppokäyttöinen ja edullinen komponentti,

mutta sen ongelmana on kontaktipintojen kuluminen. Potentiometrejä ei tämän takia suositella käytettäväksi käyttökohteissa, joissa niitä käänneltäisiin jatkuvasti. Kauko-ohjaimessa potentiometriin kohdistuisi jatkuvaa säätöä, jonka takia potentiometri voisi rajoittaa kauko-ohjaimen käyttöikää.

Parempi tekniikka jatkuvaa säätöä vaativaan kohteeseen on magneettikentän voimakkuutta mittaava Hall-sensori (Cain 2010). Hall-sensori ei ole suorassa kontaktissa pyörivään kappaleeseen, jolloin ainoita kuluvia osia ovat pyöritettävä nuppi ja sen laakerit. Hall-sensori on kuitenkin hieman potentiometriä monimutkaisempi vaihtoehto kotelon kannalta, koska se vaatii magneettien kiinnittämisen pyörivään nuppiin ja Hall-sensorin kiinnittämisen lähistölle sopivaan paikkaan. Potentiometrissä on kiinteästi mukana akseli, jolloin täytyy huolehtia vain sen kiinnittämisestä runkoon. Hall-sensoria varten taas pitäisi rakentaa jonkinlainen laakerointi tai vastaava nupin pyörittämistä varten, koska sensori ei sisällä mitään mekaniikkaa.

Kehityksessä kannattaa huomioida, että potentiometri ja tietyt Hall-sensorit käyttävät samoja liitäntöjä: käyttöjännite, maa ja ADC-linja. Tämän ansiosta kehityksessä voi alkuvaiheessa käyttää helpommin implementoitavissa olevaa potentiometriä ja myöhemmin vaihtaa sen ilman piirilevymuutosta Hall-sensoriin. Vaihto vaatii vain pieniä muutoksia koteloon ja ohjelmakoodiin.

4 KONSEPTOINTI

Kauko-ohjaimen konseptointi aloitettiin yhdessä Simo Sihvosen kanssa. Ensimmäiseksi tutkittiin markkinoilla olevien tuotteiden kauko-ohjaimia ja mietittiin, miten niitä voisi parantaa. Käyttökelpoisimmalta kauko-ohjaintyypiltä näytti sellainen, jossa kauko-ohjain oli kahvamainen ja jossa oli peukalolla käytettävä rulla nopeuden säätämiseen. Rullaa vastapäiselle puolelle voi sijoittaa turvallisuusominaisuutena kuolleen miehen kytkimen.

Konseptointivaiheessa päätettiin myös, että kauko-ohjain vain välittää laudalle tiedon rullan ja nappien tiloista ja kaikki ohjaus- ja turvallisuuslogiikka tehdään laudan puolella. Ohjaimeen haluttiin myös Micro-USB-liitin akun lataamista varten. Micro-USB-yhteensopivuuden ansiosta ohjaimen lataamiseen voi käyttää useimpia nykyaikaisia puhelimien latureita.

Ohjaimella haluttiin myös pystyä indikoimaan esimerkiksi ohjaimen akun tilaa ja yhteyden laatua. Tähän tarkoitukseen sopivaksi ajateltiin joko näyttöpaneelia, ledejä tai näiden yhdistelmää.

4.1 Prosessori

Prosessori, tai ainakin prosessoriarkkitehtuuri, oli yksi ensimmäisistä valinnoista, joita projektissa täytyi tehdä elektroniikan suhteen. Prosessorin I/O-kapasiteetti vaikuttaa muihinkin komponentteihin. Prosessoriarkkitehtuuriksi valittiin AVR helpon saatavuuden takia ja myös siksi, että siitä oli aikaisempaa kokemusta. Prosessoriarkkitehtuureja ei vertailtu, koska tiedettiin, että AVR-arkkitehtuuri kykenee kaikkiin projektin vaatimuksiin.

Käytettävissä olleissa Arduinoissa oli ATmega2560-prosessori, mutta se ei ollut käytännöllinen valinta kauko-ohjaimelle ison koon, kalliimman hinnan ja vaikeamman juotettavuuden takia. ATmega2560:ssa on huomattavasti enemmän muistikapasiteettia ja I/O-linjoja kuin kauko-ohjaimeen tarvittiin. Käyttökohteessa sopivampana valintana pidettiin ATmega328p:tä. ATmega328p:ssä oli vain muutama I/O-liitäntä enemmän kuin oli välttämätöntä, ja sen ohjelmamuisti riitti juuri ja juuri käyttämään Adafruitin muistisyöppöä OLED-kirjastoakin. ATmega328p sopi myös virrankulutuksensa ja käyttöjänniteoptioidensa puolesta paremmin akkukäyttöön.

Ensimmäiset testit päätettiin silti suorittaa käyttäen Arduinoja, joissa oli ATmega2560, koska niitä oli heti käytettävissä. Niille kirjoitettu koodi kävi myös ATmega328p:lle pelkillä porttimääritysten muutoksilla, kunhan ei käytetty ominaisuuksia, joita ATmega328p:ssä ei ole. Tämän projektin kannalta oleellisimmat erot prosessoreiden välillä on listattu taulukkoon 3.

TAULUKKO 3. ATmega328p:n ja ATmega2560:n oleellisimmat erot

	ATmega2560	ATmega328p
Kotelo	100-jalkainen TQFP	DIP-28 ja TQFP-32
FLASH (kB)	256	32
RAM (kB)	8	2
EEPROM (kB)	4	1
ADC-kanavia	16	8
I/O-linjat	86	23
Ajastimet	2 x 8 bit, 4 x 16 bit	2 x 8 bit, 1 x 16 bit
Maks. kellotaajuus	16 MHz	20 MHz
Min. käyttöjännite	4,5 V @ 0–16 MHz	2,5 V @ 8 MHz, 4 V @ 16 MHz
Virrankulutus 5 V @ 8 MHz, active	10 mA	5,2 mA
Virrankulutus 5 V @ 8 MHz, idle	2,7 mA	1,2 mA

4.2 Testit Arduinoilla

Konseptin testaaminen aloitettiin käyttämällä kahta Arduino Mega 2560 -kehitysalustaa. Molempiin Arduinoihin kytkettiin nRF24L01+-radiopiiri, joiden ohjaamiseen käytettiin Maniacbugin kirjoittamaa RF24-kirjastoa (Maniacbug 2016). Debuggaamiseen käytettiin Arduinojen sarjaportteja ja piirit ohjelmoitiin sarjayhteydellä Arduino IDEstä.

Yhteen Arduinoon kytketty radiopiiri toimi heti oikein ja antoi oikeanlaiset kuittaukset Arduinolle. Toiseen Arduinoon kytketty radiopiiri ei antanut vastaavia vastauksia, vaikka

kytkentä ja ohjelmakoodi olivat samanlaisia. Radiopiirien virransyöttöön (~10 mA) käytettiin Arduinoissa olevia 3,3 V:n lähtöjä. Molemmat Arduinot olivat kiinalaisia kopioversioita mutta eri valmistajalta tulleita. Toisen Arduinon 3,3 V:n regulaatio oli niin huono, että radiopiiri ei käynnistynyt ollenkaan. Virransyöttöongelma ratkaistiin käyttämällä pienlaitteille tarkoitettua säädettävää tasajännitelähdettä, josta saatiin 3,3 V.

Kun molemmat radiomoduulit toimivat halutulla tavalla, onnistuttiin Arduinojen välillä liikuttamaan dataa radiolinkin avulla. Yksi Arduino lähetti toiselle viestin, ja molemmat ilmoittivat lähettämänsä ja vastaanottamansa datan sarjayhteyden välityksellä tietokoneelle. Kun yksisuuntainen testi oli suoritettu onnistuneesti, alettiin Arduinojen välillä liikuttaa dataa kaksisuuntaisesti. Kaksisuuntainen dataliikenne toteutettiin käyttämällä nRF-moduulin tukemaa hyötydatan kuljettamista kuittauspakettien mukana.

Koska kauko-ohjaimeen haluttiin jonkinlainen mahdollisuus indikoida asioita, lisättiin testiin mukaan vielä SPI-väylällä ohjattava Adafruitin SSD1306 OLED-näyttö. Näyttö toimi Arduinolla Adafruitin kirjastolla. OLED-näyttö ja radiomoduuli olivat kytkettyinä samaan SPI-väylään, mutta se ei aiheuttanut ongelmia.

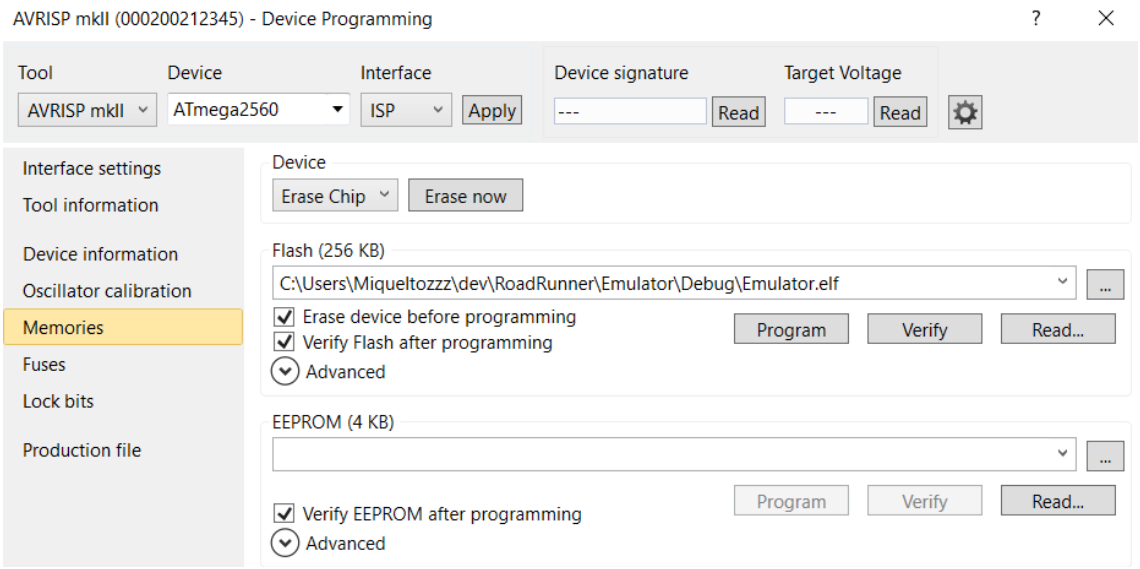
Arduino-testeillä osoitettiin, että radiomoduulit toimivat ja kaksisuuntainen tiedonsiirto onnistuu kuittauspakettien avulla. Lisäksi todennettiin, että nRF-moduuli ja OLED-näyttö pystyvät jakamaan saman SPI-väylän ongelmitta.

4.3 Testit koekytkentäalustalla

Kun konsepti oli saatu toimimaan Arduinoilla, seuraavaksi tavoitteeksi asetettiin saada ohjelma toimimaan irrallisella ATmega-prosessorilla. Tähän käytettiin DIP-28-koteloitua ATmega328-mikrokontrolleria koekytkentälevyllä.

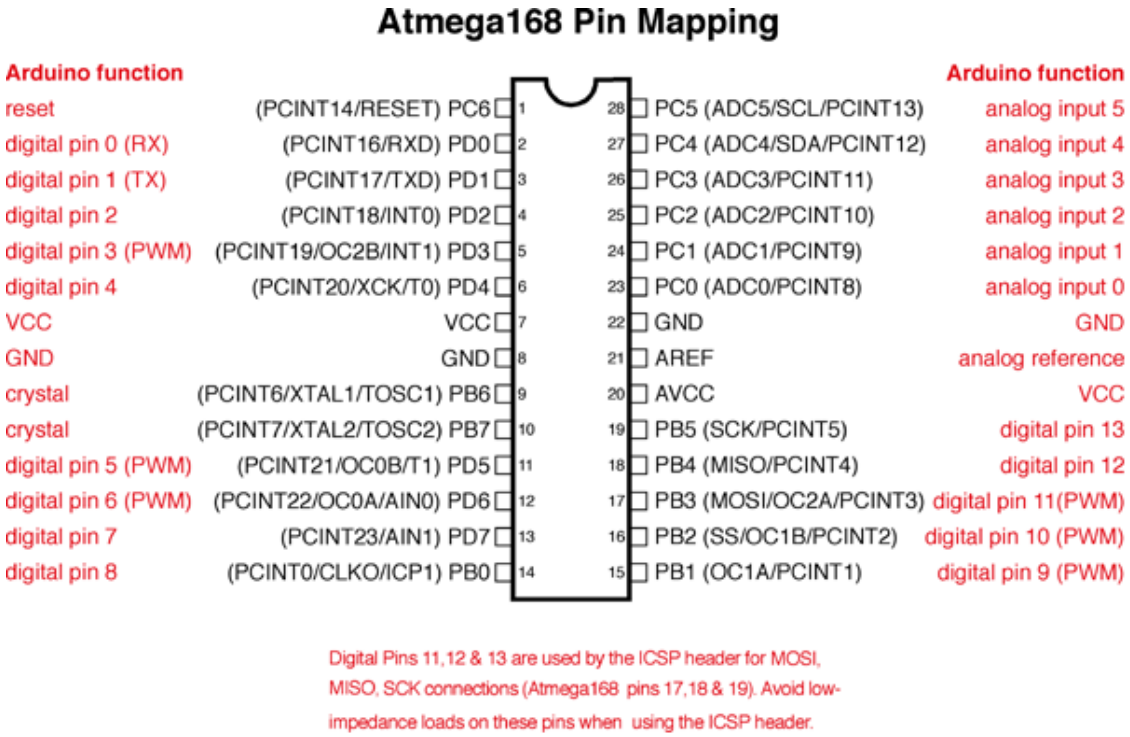
Ensimmäiseksi testattiin ohjelmakoodin siirto prosessorille, ja että prosessori aloittaa ohjelmakoodin suorituksen oikein. Testi toteutettiin kirjoittamalla sulautettujen maailman versio ”Hello World!” -ohjelmasta eli ohjelma, joka kääntää sekunnin välein yhden ulostulonastan tilaa. Ohjelma kirjoitettiin ja käännettiin Atmel Studiossa. Käännöstuloksena saatu heksatiedosto siirrettiin prosessorille Atmel Studion Device Programming -ominaisuuden avulla. Siirtämiseen käytettiin Olimexin AVR-ISP-MK2 -ohjelmointilaitetta.

Kun prosessorin ja ohjelmointilaitteen toiminta oli todettu toimivaksi, haluttiin saada radiopiiri toimimaan irrallisen prosessorin kanssa. Arduino IDEstä saa vietyä käännöstuloksen binääritiedostoksi, jonka voi siirtää Atmel Studiolla prosessoriin. Ohjelmointiprosessi on sama kuin Atmel Studiossa käännetyn koodin kanssa, ainoastaan prosessorille siirrettävän binääritiedoston polku vaihdetaan (kuva 10).



KUVA 10. Binääritiedoston polun asettaminen

Jotta Arduinon kanssa käytetty kytkentä pystyttiin toistamaan koekytkentäalustalla, piti selvittää, miten Arduino IDE:ssä käytetyt pinninumeroinnit vastasivat prosessorin datasivulla olevia pinnimäärityksiä (kuva 11). Tämä tieto löytyi Arduino-projektin kotisivuilta (Arduino 2017). ATmega168 ja 328(p) ovat pinnimääritysten suhteen identtiset.



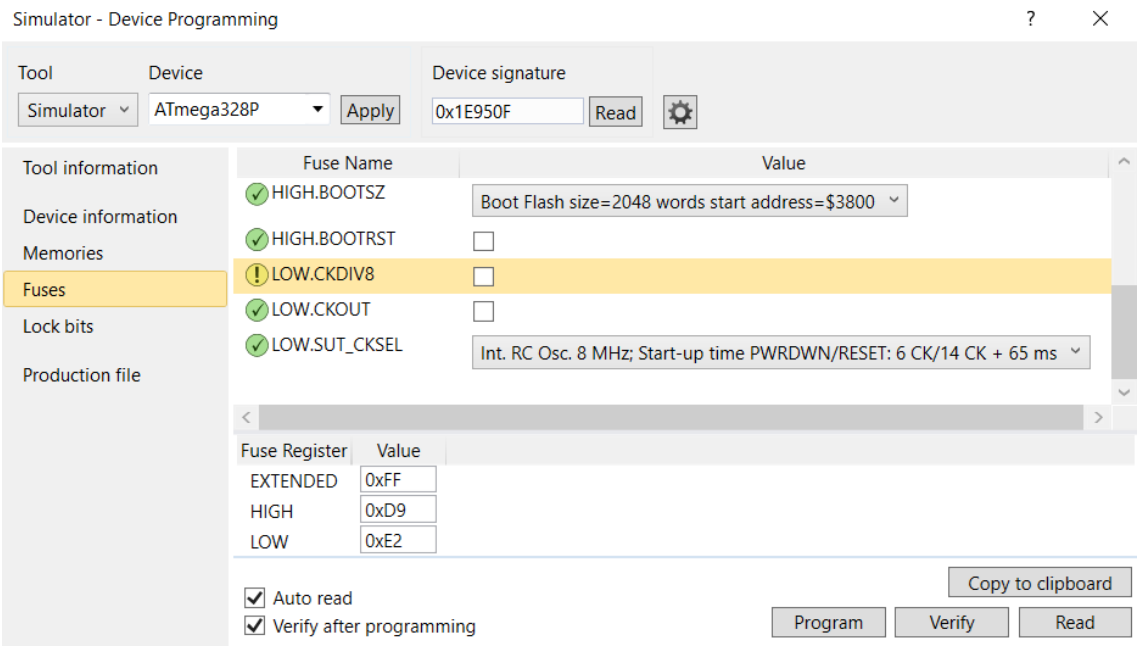
KUVA 11. Arduinon pinninumerointi ja datasivun mukainen nimeämiskäytäntö (Arduino 2017)

Käytetyissä Arduinoissa oli 16 MHz ulkoiset kiteet. Koska nyt käytettiin irrallista prosessoria, jossa ei ollut ulkoista kidettä, prosessori toimi sisäisen 8 MHz:n oskillaattorin avulla Arduinoissa käytetyn 16 MHz:n kiteen sijaan. Jotta ohjelmakoodi ei olisi suoriutunut puolella nopeudella, Arduino IDEn boards.txt-asetustiedostoa piti muokata. Tiedostoon lisättiin ATmega328p 8MHz -niminen valinta, joka kopioitiin samaa prosessoria käyttävän Arduino Nanon asetuksista, mutta kellotaajuus vaihdettiin 16 MHz:sta 8 MHz:iin (kuva 12).

```
nano8.name=Atmega328 8MHz

nano8.upload.tool=avrdude
nano8.upload.protocol=arduino

nano8.bootloader.tool=avrdude
nano8.bootloader.unlock_bits=0x3F
nano8.bootloader.lock_bits=0x0F

nano8.build.f_cpu=8000000L
nano8.build.board=AVR_NANO
nano8.build.core=arduino
nano8.build.variant=eightanaloginputs
```

KUVA 12. Arduino IDEn konfigurointitiedostoon tehdyt lisäykset

Prosessorissa oli oletuksena asetettu päälle LOW.CKDIV8 -fuse-bitti, jonka takia prosessori jakoi sisäisen oskillaattorinsa tuottaman kellotaajuuden kahdeksalla. Kyseinen fuse-bitti piti ottaa pois päältä, jotta prosessori toimi halutulla 8 MHz:n kellotaajuudella, eikä 1 MHz:n taajuudella. Fuse-bitit pystyi lukemaan ja asettamaan Atmel Studion Device Programming -toiminnolla (kuva 13). Toiminto sisälsi myös selkokieliset kuvaukset jokaisen fuse-bitin toiminnasta.



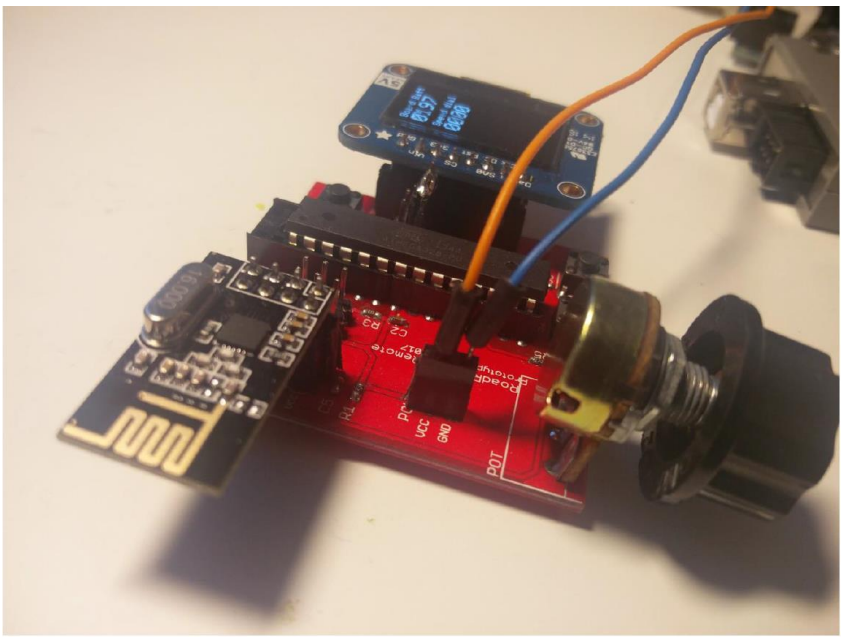
KUVA 13. Fuse-asetukset Atmel Studiossa

4.4 Ensimmäinen piirilevyversio

Kun kytkennän toiminta oli varmistettu koekytkentälevyllä, suunniteltiin prototyyppiversio piirilevystä. Tavoitteena ei ollut saada mitään järkevästi koteloon mahtuvaa aikaiseksi, vaan lähinnä vain harjoitella kaksikerrospiirilevyn suunnittelua. Koska tässä vaiheessa haluttiin käyttää jo testattua DIP-28-koteloitua prosessoria, piirilevyn kooksi valittiin 50 x 50 mm. Levyllä oli piikkirimaliittimet kaikkia ulkoisia liitäntöjä varten. Kokoonpantu piirilevy on nähtävissä kuvassa 14. Ensimmäisen piirilevyversion piirilevysuunnitelma on liitteessä 1 ja kytkentäkaavio liitteessä 2.

Tällä piirilevyllä tuli uusia ongelmia virransyötön kanssa: radiopiiri lopetti toimimisen satunnaisin väliajoin. Prosessorin reset-napin painaminen korjasi tilanteen. Yleismittarilla mitattuna käyttöjännite oli 3,3 V, mutta käyttämällä oskilloskoopin kertaliipaisutilaa havaittiin käyttöjännitteen laskevan välillä alle 2,7 V:n arvoon. Koska 3,3 V:n linjan takana oli nyt myös prosessori ja näyttö, virrankulutus oli kasvanut edellisiin testeihin nähden, eikä käytetty halpa tasajännitelähde kyennyt reguloimaan jännitettä enää tarpeeksi hyvin.

Virransyöttöongelma korjattiin tekemällä tilapäinen virtalähde vanhasta tietokoneen virtalähteestä. Ratkaisun huono puoli oli se, ettei virtaa voinut helposti rajoittaa. PC-virtalähteet pystyvät antamaan helposti useiden kymmenien ampeerien virran 3,3 V:n linjaan, jolloin kytkentöjen kanssa täytyy olla erityisen tarkkana komponenttirikkojen välttämiseksi.



KUVA 14. Ensimmäinen piirilevy

5 PROTOTYYPPI

5.1 Toinen piirilevyversio

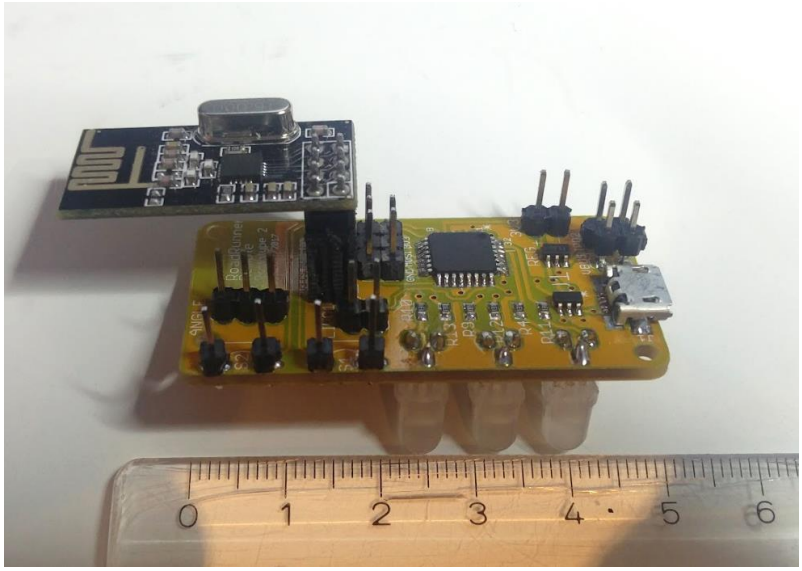
Kun konsepti oli todettu toimivaksi, tarvittiin piirilevy, jonka ympärille voisi suunnitella ja rakentaa kotelon. Jotta järjestelmää voitaisiin testata laudan kanssa, piirilevylle piti lisätä myös tarvittavat komponentit akun lataamiseen ja akkuvirralla toimimiseen. Tavoitteena oli siis rakentaa prototyyppi, jolla voi testata radiomoduulin toimintaa laudan kanssa käytännössä.

Edellisessä versiossa ollut OLED-näyttö vaihdettiin kolmeen RG-lediin, koska OLED-näyttö vaikeutti kotelosuunnittelua huomattavasti. Kolmen ledin todettiin riittävän indikointitarkoituksiin tässä vaiheessa tarpeeksi hyvin. Piirilevyä piti pienentää huomattavasti edellisestä versiosta, jotta piirilevyn koko ei vaikeuttaisi kotelon suunnittelua. Suurin tilan viejä edellisessä versiossa oli DIP-28-koteloitu prosessori (~3,55 cm^2, läpijuotettava). Toisessa piirilevyversiossa käytettiin TQFP-32-koteloitua versiota samasta prosessorista (~0,90 cm^2, pintaliitos). Piirilevyn pinta-ala pienennettiin alle puoleen edellisestä versiosta, 25,00 cm^2:stä 12,48 cm^2:iin. Toinen piirilevyversio on esitetty kuvassa 15 kokoonpantuna. Toisen piirilevyversion piirilevysuunnitelma on liitteessä 3 ja kytkentäkaavio liitteessä 4.

Piirilevylle laitettiin tässäkin versiossa riviliittimet kytkimiä ja potentiometrejä varten, jolloin kytkimien malli ja sijainti eivät olleet sidotut piirilevyyn. Kytkimet kiinnitettiin suoraan kiinni runkoon, joten liittimien sijainti piirilevyllä ei ollut oleellista. Radiomoduulin alle jätettiin kupariton alue, jotta radiosignaali pääsisi paremmin läpi.

Toisesta piirilevyversiosta ilmeni yksi selkeä virhe sekä joitain ominaisuuksia, joita voi jatkossa parantaa. ATmega328p resetoidaan viemällä nollatila reset-nastaan. Ensimmäisessä piirilevyversiossa oli reset-kytkin ja reset-nastan ylösvetovastus. Toisesta piirilevyversiosta oltiin poistettu reset-kytkin, ja samalla ylösvetovastus oli vaihtunut vahingossa alasvetovastukseksi. Prosessoriin pystyi siirtämään ohjelman, mutta ohjelma ei käynnistynyt, koska prosessori oli pysyvästi resetissä. Ongelma korjattiin poistamalla alasvetovastus piirilevyltä ja käyttämällä vain prosessorin sisäistä ylösvetovastusta.

Kytkennässä käytetyt läpijuotettavat ledit eivät olleet loppujen lopuksi paras ratkaisu. Suurin ongelma ledien kanssa oli se, että ne olivat niin korkeita, että se vaikutti selvästi kotelonkin korkeuteen. Lisäksi ledit olivat yksi isoimmista tilan viejistä piirilevyllä. Piikkirima oli helppo ratkaisu kytkimien ja muiden komponenttien kiinnittämiseen levylle mutta ei paras mahdollinen. Jos johtojen päähän kiinnittää piikkiriman vastakappaleet, piirilevyn päällä tarvitaan vähintään 20 mm tilaa. Johdot voi juottaa suoraan kiinni piirilevylle, jolloin ne eivät vaadi merkittävästi korkeutta, mutta silloin johtojen irrottaminen on vaikeaa. Parempi ratkaisu olisi ollut laittaa piirilevyn reunalle pienet vaakasuuntaiset liittimet, kuten DF52 tai JST SH. Kyseisten liittimien käyttäminen olisi säästänyt tilaa piirilevyllä, vaatinut vähemmän korkeutta ja mahdollistanut silti kytkimien helpon irrottamisen. DF52-liittimiä on lisäksi saatavilla pintaliitosversioina, jotka säästäisivät tilaa vielä lisää.



KUVA 15. Toinen piirilevyversio, jossa radiomoduuli liitettynä

5.1.1 Akkujärjestelmä

Prosessorin ja radiopiirin käyttöjännite on 3,3 V. LiPo-akun jännite vaihtelee välillä 3,0–4,2 V. 3,3 V:n käyttöjännitteen voi muodostaa akkujännitteestä yksinkertaisen lineaariregulaattorin avulla akkujännitteen ollessa hieman suurempi kuin tarvittava käyttöjännite. Osa akun kapasiteetista jää tässä tapauksessa käyttämättä, mutta se ei ole iso ongelma. Regulaattorilta haluttuja ominaisuuksia olivat pieni virrankulutus ja pieni jännitehäviö.

Digikeyn valikoimasta löytyi STMicroelectronicsin SOT23-5 -koteloitu LD39015 joka täytti halutut kriteerit. Regulaattorin tärkeimmät ominaisuudet on listattu taulukossa 4.

TAULUKKO 4. LD39015:n ominaisuuksia (STMicroelectronics 2014, muokattu)

Ulostulojännite	3,3 V
Jännitehäviö	80 mV
Ulostulovirta	150 mA
Virrankulutus	18 µA
Käyttöjännite	3,38–5,5 V

Akkua haluttiin pystyä lataamaan puhelimen Micro-USB-laturilla. Micro-USB-latureista tulee 5 V:n jännite, ja virtaa ne kykenevät antamaan laturista riippuen 500 mA–2 A. Akkua ei voinut kytkeä suoraan puhelimen laturiin, vaan väliin piti laittaa sopiva LiPo-latauspiiri. Latauspiiri kontrolloi latausvirtaa ja estää akun ylilatautumisen.

5.1.2 LiPo-laturin toiminta

LiPo-laturin latausprofiili on esitetty kuvassa 16. Kun LiPo-laturiin kytketään käyttöjännite, piiri tarkistaa ensin kennojännitteen. Jos kennojännite on alle LOWV-rajan (2,5 V), piiri tulkitsee akun täysin tyhjennetyksi ja aloittaa latauksen huoltosyklillä. Tässä tilassa lataamiseen käytetään vain 20 % maksimivirrasta. (Texas Instruments 2016.)

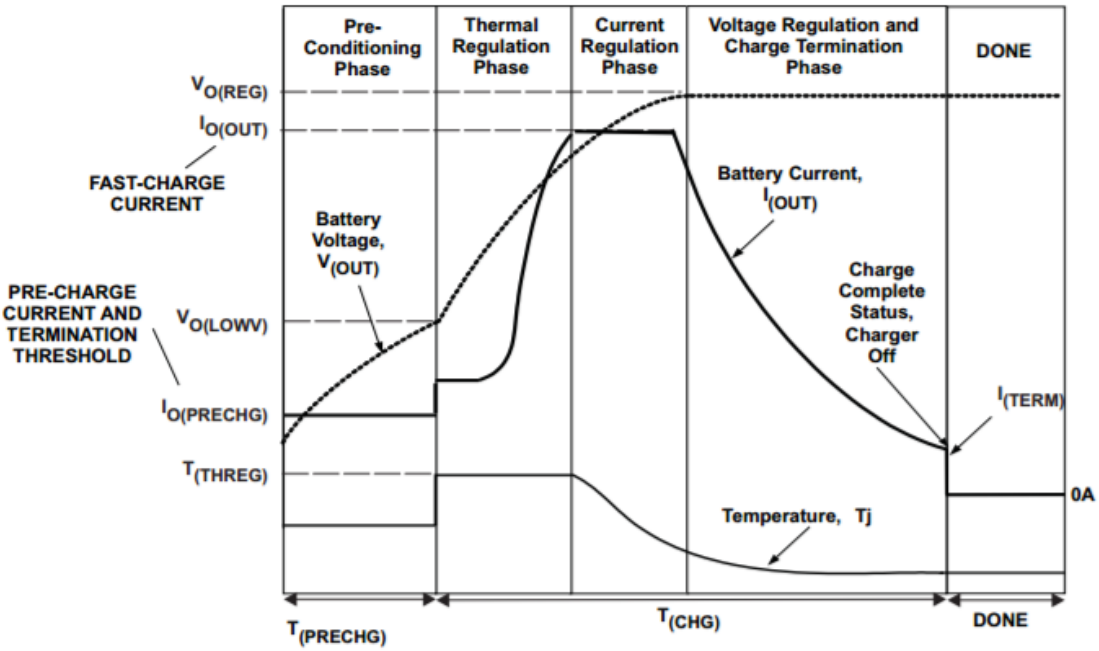
Kun akku on latautunut yli LOWV-rajan, latausprosessi siirtyy pikalataustilaan. Käytettävän virran määrä ohjelmoidaan ISET-vastuksella. Suurin osa akun kapasiteetista ladataan tässä vaiheessa. Jos latauspiiri kuumenee liikaa, se rajoittaa latausvirtaa. I_{SET}-vastus mitoitetaan kaavan 1 avulla. (Texas Instruments 2016.)

$$R_{ISET} = \frac{K_{ISET}}{I_{OUT}} \quad (1)$$

Kaavassa 1 R_{ISET} on I_{SET}-vastuksen resistanssi ohmeina, I_{OUT} on haluttu maksimilatausvirta ampeereina, ja K_{ISET} on vahvistuskerroin, jonka arvo riippuu latausvirrasta. Kaukoohjaimen prototyypissä käytettiin hyllyssä ollutta 150 mAh:n LiPo-akkua, jolle 1C latausvirta oli siten 150 mA. K_{ISET} valittiin datasivun *Electrical characteristics* -taulukon

sarakkeesta *Fast charge current factor* kohdasta 50–800 mA, jossa sen arvo oli 540 (Texas Instruments 2016). Sijoittamalla vahvistuskertoimen ja latausvirran kaavaan saatiin tulokseksi 3 600 Ω.

Kun akku on latautunut vakiovirtavaiheessa regulaatiojännitteelle (4,2 V) asti, laturi siirtyy vakiojännitetilaan. Kun latausvirta laskee vakiojännitetilassa 10 %:n arvoon maksimista, piiri tulkitsee akun ladatuksi ja lataus lopetetaan. (Texas Instruments 2016.)



KUVA 16. BQ21040:n latausprofiili (Texas Instruments 2016)

BQ21040 sisältää useita turvaominaisuuksia itsensä, teholähteen ja akun suojaamiseksi. Latauspiiri tunnistaa sekä akun että virranasetusvastuksen oikosulut. Piiri valvoo myös omaa lämpötilaansa ja sisältää option akun lämpötilan valvomiselle. Jos syöttävä jännite laskee, latauspiiri vähentää latausvirtaa teholähteen suojaamiseksi. Latauspiiri sisältää myös 10 tunnin ajastimen ylilataamisen estämiseksi. (Texas Instruments 2016.)

5.2 Ohjelma

Kauko-ohjaimen ohjelmakoodi on esitetty liitteessä 5. Ohjelmassa käytettiin itse kirjoitettua kirjastoa nRF-moduulin ohjaamiseen. Liitteen 5 lopussa on linkit sekä kauko-ohjaimen että nRF-kirjaston Bitbucket repositoryihin.

Kaikki prosessorin rautamääritykset tehtiin *remote.c* ja *NRF24L01P.c* -tiedostojen alussa olevilla *#define*-määrityksillä konfiguraatiomuutosten helpottamiseksi. Kauko-ohjaimen ohjelma käyttää samaa radiopaketin asetustiedostoa kuin emulaattori ja lauta, *RoadRunner_Protocol.h*, jossa on määritelty radiolla välitettävien datapakettien pituus ja sisältö. Tämän ansiosta paketin sisältöön tehdyt muutokset ja lisäykset päivittyvät helposti sekä lautaan että kauko-ohjaimeen.

Ohjelma on ajastinkeskeytysohjattu. Prosessori suorittaa asetetun intervallin välein keskeytysohjelman, joka asettaa keskeytyslipun *g_irq1* ylös. Pääohjelmasilmukassa on tarkistus, jossa tutkitaan keskeytyslipun sisältö. Jos lippu on asetettuna, siirrytään suorittamaan ajastettua ohjelmaosiota ja lippu asetetaan nollaksi. Kauko-ohjain lähettää asetetun intervallin välein paketin laudalle. Paketti sisältää ohjaimen akun tilan, kytkimien tilat ja potentiometrin tilan. Ohjelma laskee myös onnistuneita ja epäonnistuneita paketteja ja välittää sen tiedon laudalle. Kauko-ohjain saa laudan automaattisen kuittauspaketin mukana tiedon laudan akun tilasta.

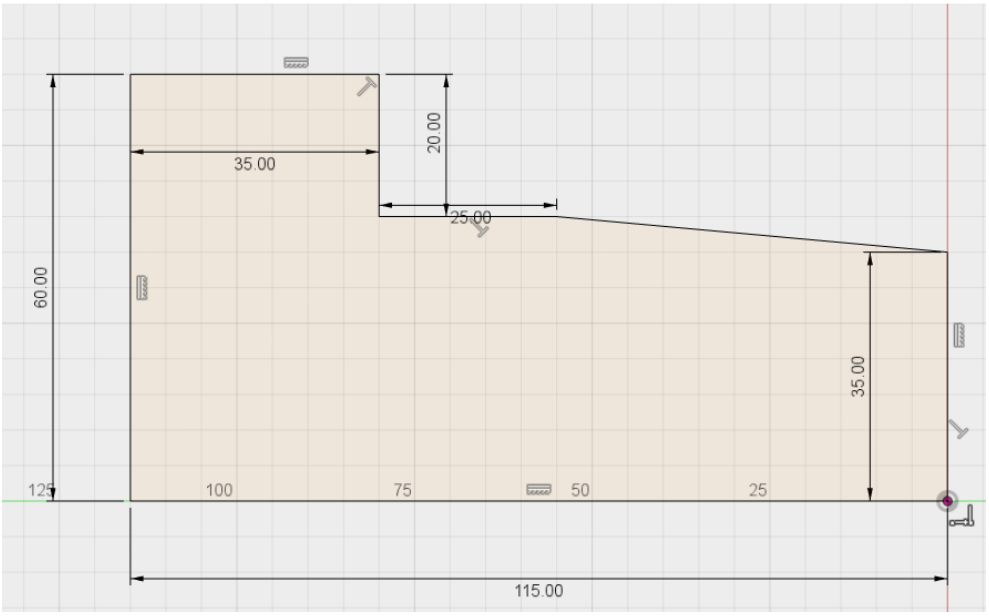
Kauko-ohjain ilmaisee yhteyden tilaa, ohjaimen akun tilaa ja laudan akun tilaa RG-ledien avulla. RG-ledit saa palamaan helposti vihreänä, keltaisena ja punaisena. Kun ohjaimen akkua ladataan, toinen led-valo välkkyy punaisena. Kun akku ei ole latauksessa, valo palaa tasaisen vihreänä, keltaisena tai punaisena akun varauksen mukaisesti. Laudan akun tila ilmaistaan kolmannella ledillä muuten samoin kuin ohjaimen akun tila, mutta laudan akun lataamista varten ei ole vilkkumistoimintoa.

Pääohjelma analysoi jatkuvasti yhteyden tilaa. Virheen tapahtuessa *transmit_errors*-muuttujan arvoa nostetaan kymmenellä. Muuttujan maksimiarvoksi on rajattu 105. Lähetyksen onnistuessa muuttujan arvoa lasketaan yhdellä. Pääohjelmasilmukassa ajetaan aliohjelmaa *transmit_led_control(uint8_t errors)*, joka ohjaa yhteyden tilaa ilmaisevaa lediä. Ledi palaa vihreänä, kun virhelukema on alle 15, keltaisena välillä 16–64 ja punaisena kun lukema on 65 tai suurempi.

NRF-moduulin ohjauskirjasto vaatii siirrettävän datan *uint8_t*-taulukkona. NRF-kirjasto myöskin antaa saapuneen datan *uint8_t*-taulukkona. Koska siirrettävä data on kaksitavuisia arvoja, data täytyy muuntaa aina ennen lähetystä ja lukemisen jälkeen sopivaan muotoon. Kauko-ohjaimen ohjelma sisältää *uint8_to_uint16* ja *uint16_to_uint8* -funktiot konversiota varten.

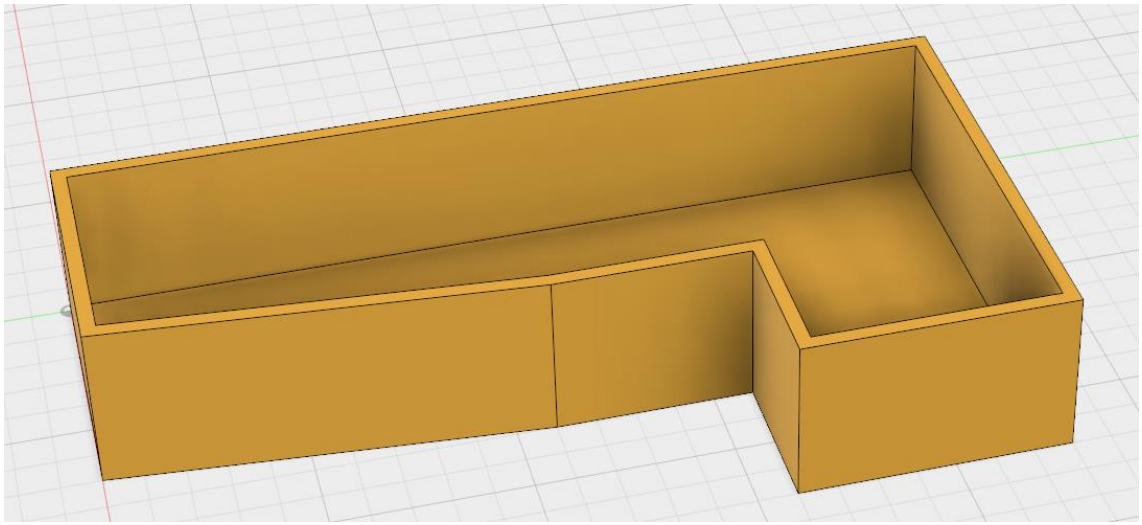
5.3 Kotelo

Piirilevyä varten piti tehdä kotelo, jonka kanssa voi tehdä alustavia testejä laudan kanssa. Alustavia testejä varten tehdyn kotelon ei tarvinnut mahtua taskuun tai olla ergonominen. Kotelon ainoa tarkoitus oli pitää komponentteja sen verran kasassa, että laudan ohjaaminen on mahdollista yhdellä kädellä. Ohjaimesta päätettiin tehdä kahva, jossa on suurempi osio yläosassa. 3D-mallintaminen aloitettiin tekemällä yksinkertainen 2D-piirustus (kuva 17). Ohjaimen yläosa on kuvassa vasemmalla. Kaikista komponenteista oli otettu mitat piirustusta varten, jotta kotelosta pystyttiin suunnittelemaan sopivan kokoinen.



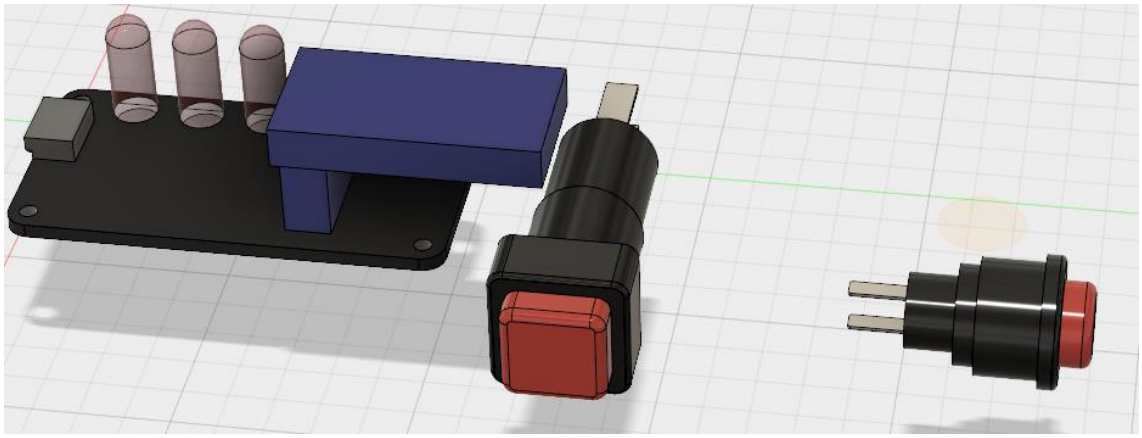
KUVA 17. 2D-piirustus

2D-piirrustus nostettiin 3D-muotoon Fusion 360:n *extrude*-toiminnolla. Extruden jälkeen kappaleeseen piirrettiin ohuet seinämät *offset*-toiminnon avulla ja seinämät nostettiin 3D-muotoon (kuva 18).



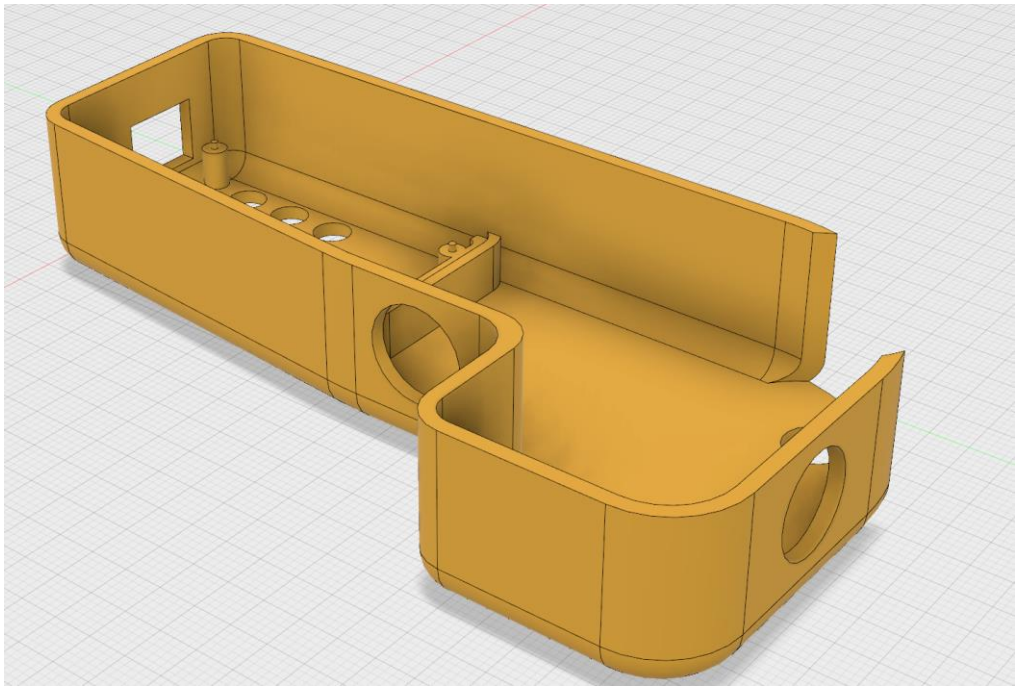
KUVA 18. Yksinkertainen 3D-muoto

Seuraavaksi mallinnettiin kaikkien käytettyjen komponenttien ulkomuodot (kuva 19).



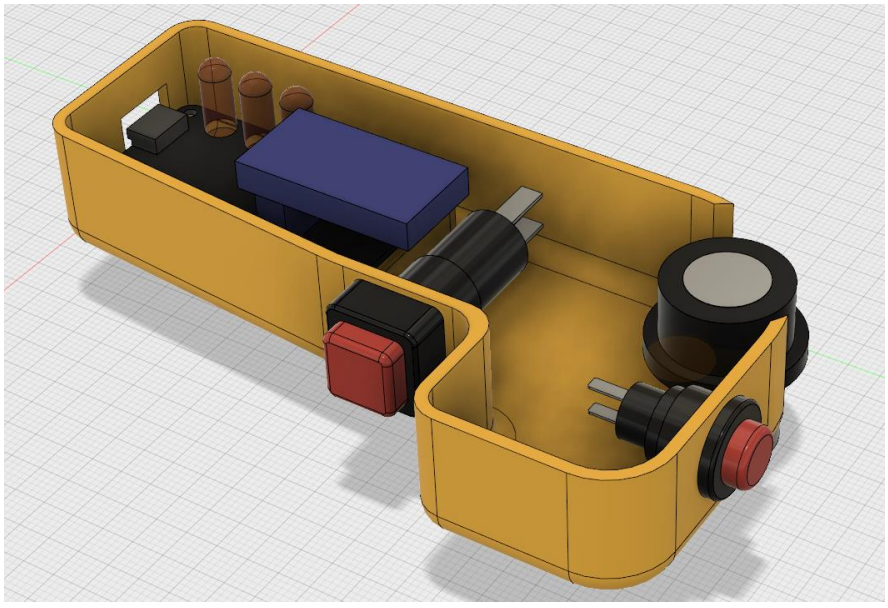
KUVA 19. Komponenttien 3D-mallit

Komponenttien mallintamisen jälkeen runkoon tehtiin pyöristykset sekä tarvittavat reiät ja korokkeet komponentteja varten (kuva 20).



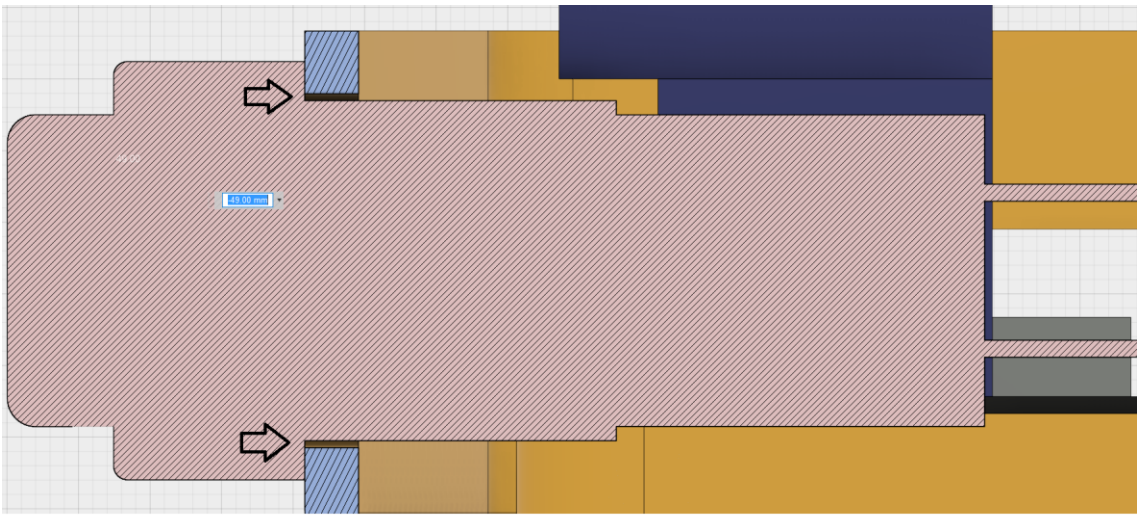
KUVA 20. Runko kiinnitysreikien ja korokkeiden kanssa

Kaikki komponentit siirrettiin asianmukaisille paikoilleen suhteessa koteloon (kuva 21). Komponenttien sijoittelun ansiosta 3D-mallista oli helppo tarkistaa, että mitkään komponentit eivät mene päällekkäin ja että kaikkialla on tarpeeksi suuret välistykset.



KUVA 21. 3D-malli rungon ja osien kanssa

Fusion 360:ssa on myös *section analysis* -toiminto, jolla voi tarkastella 3D-mallin poikkileikkausta. Kytkimiä varten tehtyjen reikien pelivara oli helppo tarkistaa poikkileikkauksesta (kuva 22). Pelivara on korostettu kuvassa nuolien avulla.



KUVA 22. Poikkileikkaus kuolleen miehen kytkimestä

YHTEENVETO

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa sähkörullalaudan kauko-ohjaimen prototyyppi. Työn lopputuloksena oli toimiva prototyyppi, jossa oli kaikki oleelliset ominaisuudet: langaton tiedonsiirto, akkujärjestelmä, tarvitut kytkimet ja potentiometri ohjaamista varten sekä kotelo. Ohjaimen toiminta testattiin lautaa emuloivan järjestelmän kanssa mutta ei varsinaisen laudan kanssa, koska vuodenaika ei ollut sopiva ulkona rullalautailuun.

Työssä käytettiin inkrementaalista kehitystapaa, eli käytännössä työssä toteutettiin aina juuri ne ominaisuudet, joita sillä hetkellä tarvittiin, eikä mitään ylimääräistä. Tämä kehitystapa auttoi saamaan tuloksia nopeasti ja esti työn ominaisuuksien määrää kasvamasta liian suureksi. Jos työhön olisi ottanut mukaan enemmän ominaisuuksia, työ tuskin olisi valmistunut aikataulussa.

Opinnäytetyön lopputuloksena valmistunut ohjaimen prototyyppi sisältää kaikki vaaditut ominaisuudet, mutta siinä on vielä huomattavasti parantamisen varaa. Prototyypin tarkoitus ei kuitenkaan ollut saada kaikkea kerralla oikein vaan lähinnä kehittää alusta, jonka päällä jatkokehitystä on helppo tehdä. Prototyypin piirilevy suunniteltiin siten, että se on pienikokoinen ja kaikki ulkoiset komponentit kiinnitetään johdoilla. Tämän ansiosta saman piirilevyn avulla voidaan testata tulevaisuudessa useita eri kotelovaihtoehtoja.

Piirilevyyn jäi vielä pienentämisvaraa, eivätkä läpijuotettavat ledit olleet loppujen lopuksi tilankäytön kannalta hyvä vaihtoehto. Myös piirilevyllä olevat kytkimien, akun ja potentiometrin liittimet pitäisi vaihtaa joihinkin kompakteihin liittimiin, kuten JST XH tai DF52. Ohjelman parissa voidaan tehdä vielä useita parannuksia, kuten taajuushyppelyn käyttöönotto.

Kauko-ohjaimen kehitystä jatketaan tänä kesänä, kun prototyypin toimivuus päästään testaamaan oikean laudan kanssa. Jos radioliikenne ohjaimen ja laudan välillä toimii luotettavasti, piirilevyä aletaan optimoida ja siihen lisätään uusia ominaisuuksia.

LÄHTEET

Archambeault, Bruce. Design Tips: PCB Decoupling Capacitor Mounting Top – Bottom – Either? Luettu 2.2.2017. http://www.emcs.org/acstrial/newsletters/fall06/design-tips.pdf

Arduino. ATmega168/328 – Arduino Pin Mapping. Luettu 3.1.2017. https://www.arduino.cc/en/Hacking/PinMapping168

Battery University. Luettu 15.3.2017. http://batteryuniversity.com/learn/archive/is_lithium_ion_the_ideal_battery

Cain, Paul. Pot vs. sensor 2010. Piher International. Luettu 3.1.2017. http://www.mouser.com/pdfDocs/Piher_Pot_vs_Sensor.pdf

Cockrell School of Engineering, The University of Texas at Austin 2014. Tips for Designing 3D Printed Parts. Luettu 9.4.2017. https://innovationstation.utexas.edu/tip-design/

Maniacbug. RF24. GitHub-repository. Luettu 10.12.2016. https://github.com/maniacbug/RF24

New Star Vision. Difference between LED SMD LED and traditional (PTH). Luettu 9.4.2017. http://www.newstarvision.it/en/led-smd-pth.html

Nordic Semiconductors 2008. nRF24L01+ Product Specification v1.0. Luettu 3.2.2017. https://www.nordicsemi.com/eng/content/download/2726/34069/file/nRF24L01P_Product_Specification_1_0.pdf

Propwashed 2016. How LiPo Batteries Explode. Luettu 16.4.2017. http://www.propwashed.com/how-lipo-batteries-explode/

Rikoslaki 21.4.1995/578.

Sihvonen, Simo 2017. Haastattelu 8.4.2017. Haastattelija M. Jokinen.

Sihvonen, Simo 2016. The current iteration of my electric longboard. Luettu 9.4.2017. http://imgur.com/a/Mjodj

SLAC National Accelerator Laboratory 2015. Study Finds a Way to Prevent Fires in Next-Generation Lithium Batteries. Luettu 9.4.2017. https://www6.slac.stanford.edu/news/2015-06-17-study-finds-way-prevent-fires-next-generation-lithium-batteries.aspx

STMicroelectronics 2014. LD39015 datasivu Rev 4. Luettu 9.4.2017. http://www.st.com/content/ccc/resource/technical/document/datasheet/a7/93/c3/c8/f7/1a/42/a4/CD00173477.pdf/files/CD00173477.pdf/jcr:content/translations/en.CD00173477.pdf

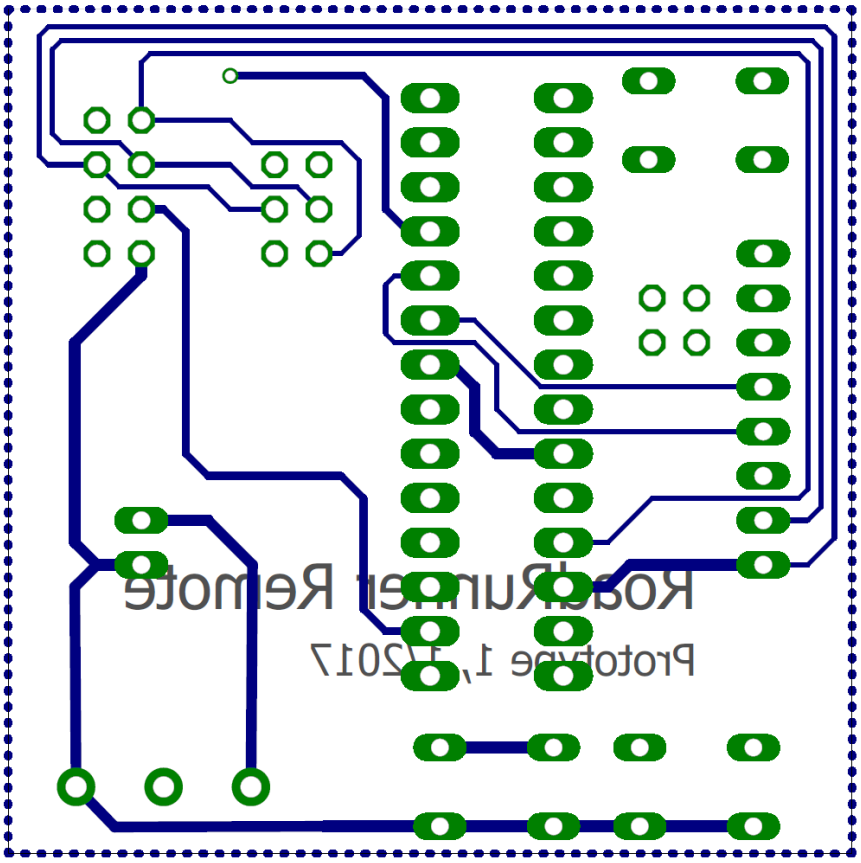
Texas Instruments 2016. BQ21040 datasivu. Luettu 9.4.2017. http://www.ti.com/lit/ds/symlink/bq21040.pdf

Viestintävirasto. Määräys luvasta vapaiden radiolähettimien yleistaajuuksista ja käytöstä, 15 AK/2016 M. 2016. Luettu 9.4.2017. https://www.viestintavirasto.fi/attachments/maaraykset/Maarays_15AK.pdf

Viestintävirasto. Radiotaajuudet ja niiden käyttö. Luettu 8.4.2017. https://www.viestintavirasto.fi/taajuudet/radiotaajuuksienkaytto.html

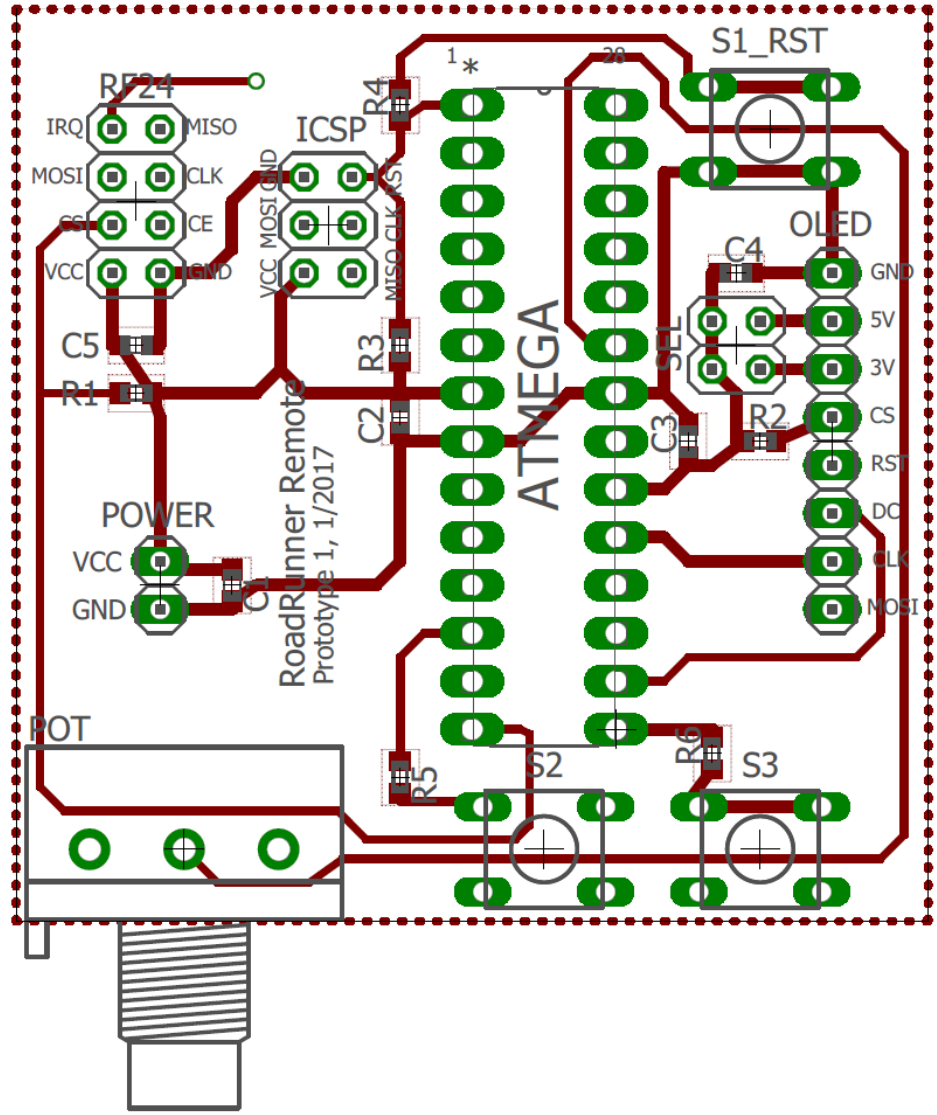
LIITTEET

Liite 1. Piirilevyversion 1 piirilevysuunnitelma



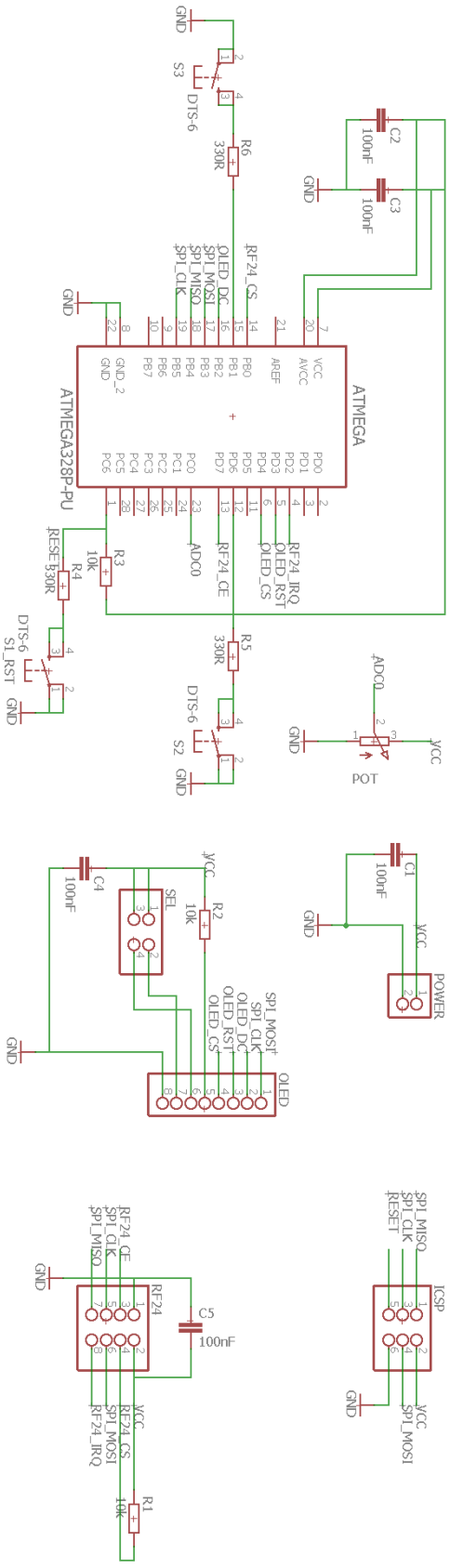
Piirilevyn alapuoli

(jatkuu)

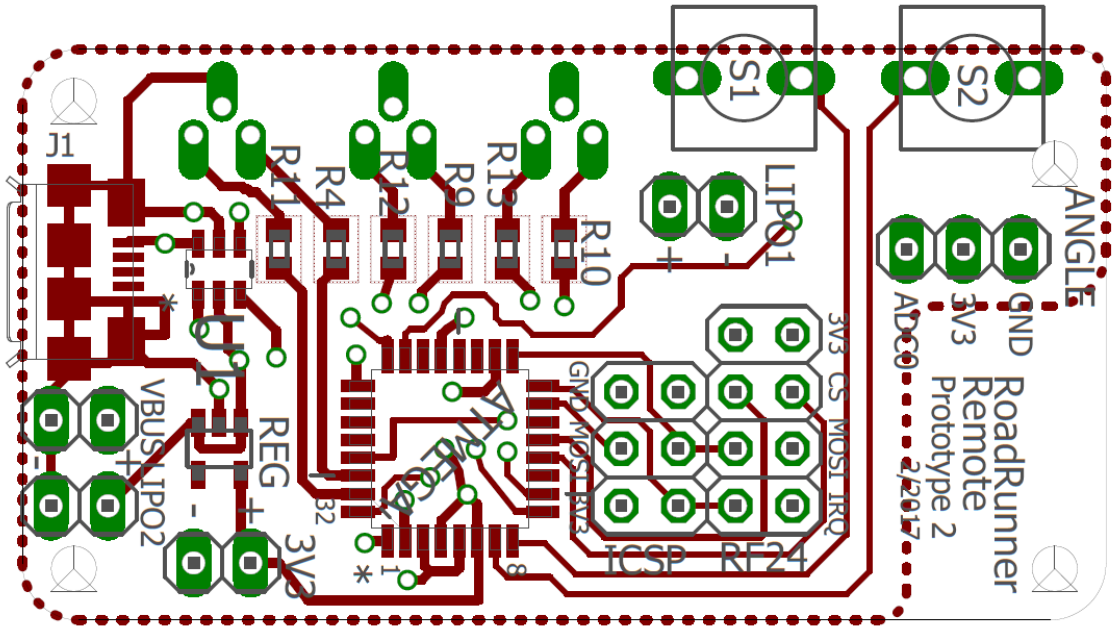


Piirilevyn yläpuoli

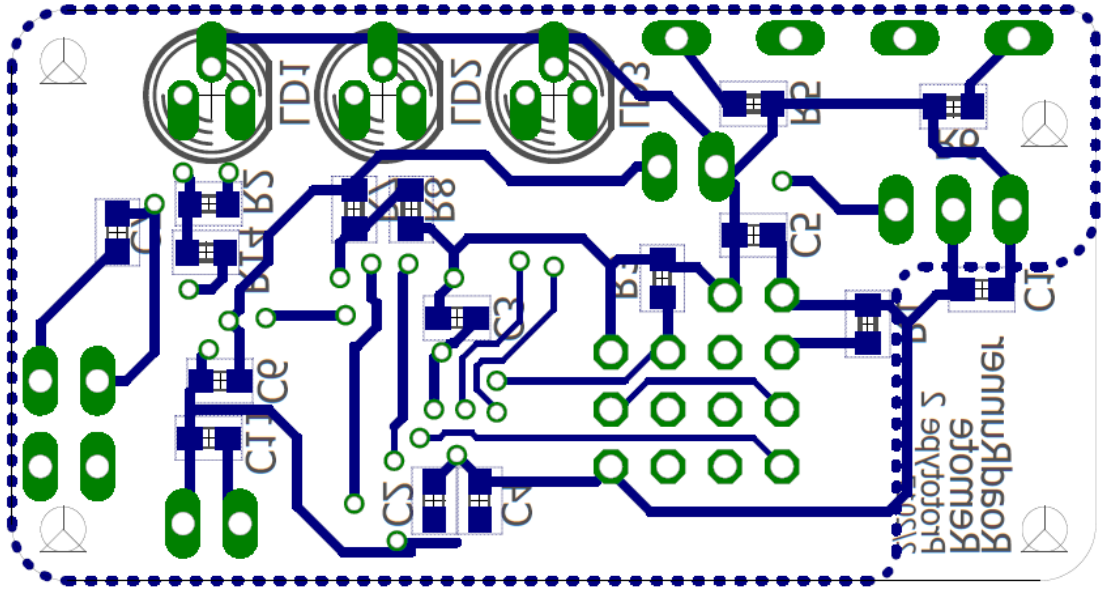
Liite 2. Piirilevyversion 1 kytkentäkaavio



Liite 3. Piirilevyversion 2 piirilevysuunnitelma

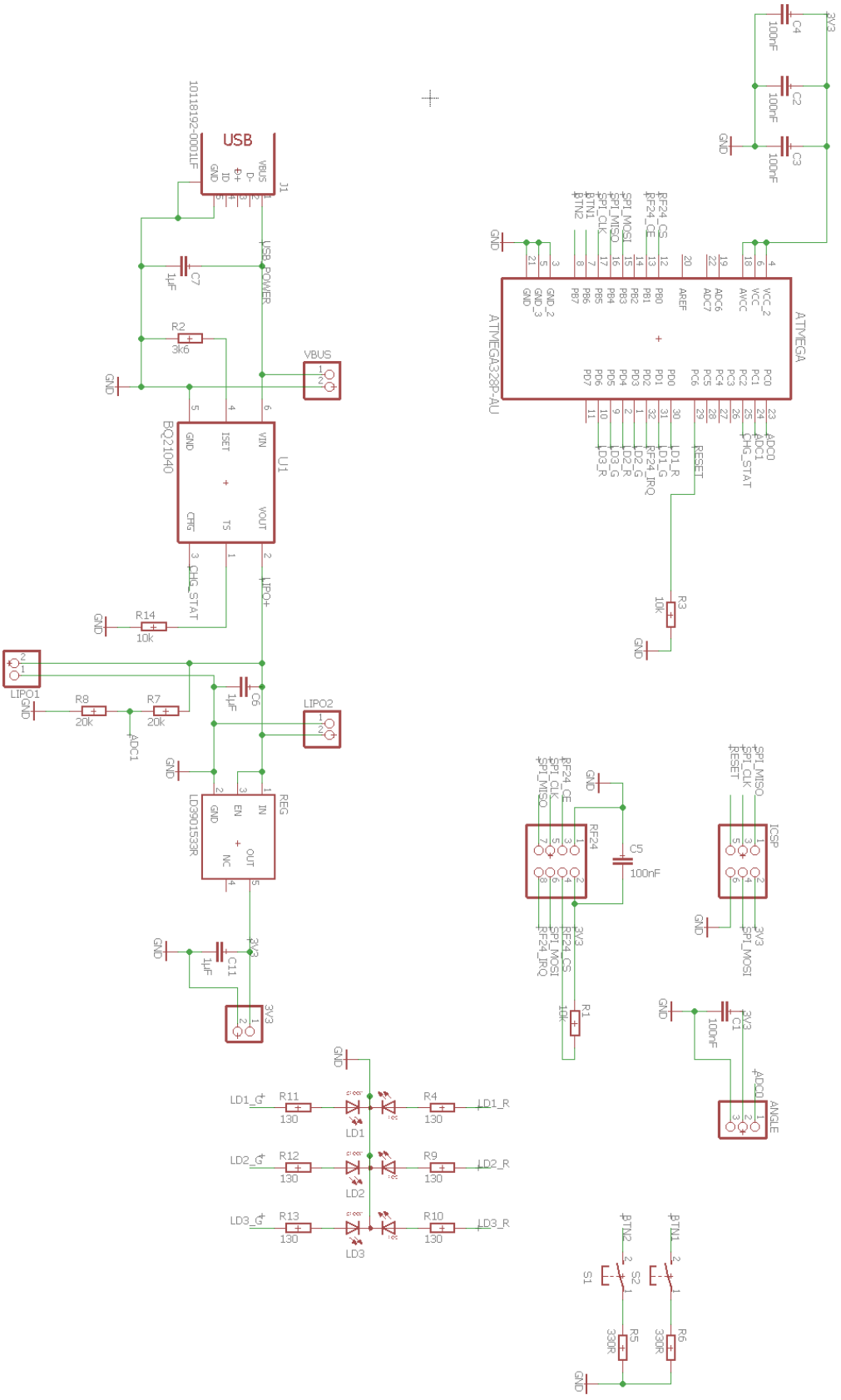


Piirilevyn yläpuoli



Piirilevyn alapuoli

Liite 4. Piirilevyversion 2 kytkentäkaavio



Liite 5. Lähdekoodi


```
/*
 * Remote.c
 *
 * Created: 16.3.2017 17.10.29
 * Author : Miqueltozzz
 *
 */
#define F_CPU 8000000UL
//Leds
#define LED_PORT PORTD
#define LED_DDR DDRD
#define LED1G PD0
#define LED1R PD1
#define LED2G PD4
#define LED2R PD3
#define LED3G PD6
#define LED3R PD5
//ADCs
#define SPEED_DIAL_ADC_PIN PC0
#define LIPO_ADC_PIN PC1
//Buttons
#define BTN_REG PINB
#define BTN1_PIN PB6
#define BTN2_PIN PB7
//Other
#define CHG_STATUS_REG PINC
#define CHG_STATUS_PIN PC2
//Interrupts
#define OCR1A_FREQ 20     //Frequency of wanted timer 1 compare A interrupt. Limited to about 2 Hz - 60 kHz
#define OCR1A_INTERVAL (F_CPU / 64 / OCR1A_FREQ)    //Calculates increment value for the OCR1A register

#include <avr/io.h>       //IO definitions for the MCU set in project settings
#include <util/delay.h>   //Needed for _delay_ms(...)
#include <stdbool.h> //For boolean operators
#include <avr/interrupt.h>     //For timer interrupts
#include "NRF24L01P/NRF24L01P.h"    //For the radio module
#include "../RoadRunner_Protocol.h"

void setup();
void uint16_to_uint8(uint16_t input, uint8_t array[]);
uint16_t uint8_to_uint16(uint8_t input[]);
void transmit_remote_packet(uint16_t remote_packet[]);
void read_board_packet(uint16_t board_packet[]);
uint16_t read_adc(uint8_t adc_num);
bool read_pin(uint8_t register, uint8_t pin);
void radio_setup();
void transmit_led_control(uint8_t errors);
void remote_battery_led_control(uint16_t charge);
void board_battery_led_control(uint16_t charge);
uint8_t read_charge_level();

//Global variables
volatile bool g_irq1 = 0; //Timer interrupt flag, modified in an interrupt so must be volatile

ISR (TIMER1_COMPA_vect)
{
     OCR1A += OCR1A_INTERVAL;
     g_irq1 = true;
}

int main(void)
{
     setup();  //DDR setup, radio module setup, ADC setup
     uint16_t remote_packet[REMOTE_PACKET_LENGTH] = {0};
     uint16_t board_packet[BOARD_PACKET_LENGTH] = {0};
     uint8_t transmit_errors = 0;
     while (1) //Main program loop
     {
          if(g_irq1)//if IRQ flag is set
          {
               g_irq1 = false; //Clear IRQ flag
               //Load variables to remote_packet and transmit it
               remote_packet[REMOTE_SPEED_DIAL_E] = read_adc(SPEED_DIAL_ADC_PIN);
               remote_packet[REMOTE_BATTERY_E] = (uint16_t)read_charge_level();
               remote_packet[REMOTE_DEADMANS_E] = !read_pin(BTN_REG, BTN1_PIN);
               remote_packet[REMOTE_FUNCTION_E] = !read_pin(BTN_REG, BTN2_PIN);
               transmit_remote_packet(remote_packet);
```

```
//Wait for transmission status IRQ
                while(1)
                {
                     if(nrf_transmit_fail())
                     {
                          remote_packet[REMOTE_ERROR_COUNT_E]++;
                          if(transmit_errors < 105)
                          {
                               transmit_errors += 10;
                          }
                          break;
                     }

                     else if(nrf_transmit_success())
                     {
                          remote_packet[REMOTE_FRAME_COUNT_E]++;
                          if(transmit_errors > 0)
                          {
                               transmit_errors--;
                          }
                          break;
                     }
                }
                if(nrf_data_available())  //If received ACK payload
                {
                     read_board_packet(board_packet);
                }
                transmit_led_control(transmit_errors);
                remote_battery_led_control(read_charge_level());
                board_battery_led_control(board_packet[BOARD_BATTERY_E]);
           }    //IRQ1 end
      }    //Main program loop end
}    //Main end

void setup()
{

      radio_setup();
      //Pull-up setup
      PORTB |= (1<<BTN1_PIN) | (1<<BTN2_PIN);
      PORTC |= (1<<CHG_STATUS_PIN);
      //Timer/Interrupt setup
      TIMSK1 |= (1<<OCIE1A);    //Enable timer 1 Compare A interrupt
      TCCR1B |= (1<<CS10) | (1<<CS11);    //Timer 1 prescaler: 64
      OCR1A = OCR1A_INTERVAL;   //Initial Timer1 compare A interrupt target
      sei();    //Enable interrupts
      //Set DDR for leds
      LED_DDR |= (1<<LED1R) | (1<<LED1G) | (1<<LED2R) | (1<<LED2G) | (1<<LED3R) | (1<<LED3G);
      //Configure ADC
      ADCSRA |= (1<<ADEN); //Enable ADC
      ADMUX &= 0x00;  //clear ADMUX
      ADMUX |= (1<<REFS0); //Set reference voltage to VCC
}

void radio_setup()
{
      uint8_t address[] = {0,1,2,3,4};    //Same address can be used for RX and TX for now
      _delay_ms(100); //Radio module startup time from power off
      _nrf_spi_init();
      nrf_enable_auto_ack(0, true);  //Enable auto-ack for pipe 0
      nrf_enable_data_pipe(0, true); //Enable data pipe 0
      nrf_set_address_width(5); //Set 5 byte address width
      nrf_set_channel(1);  //Set RF channel 1, 2,401GHz
      nrf_set_transmit_power(3);     //Set transmit power, 0 dBm
      nrf_set_data_rate(0);     //Set 250 kbps data rate
      nrf_set_retransmit_delay(750); //Set 750 us retransmission delay
      nrf_set_retransmit_count(15);  //15 transmission retries
      nrf_set_rx_address(0,address); //Set RX data pipe 0 address
      nrf_set_tx_address(address);   //Set TX address
      nrf_set_payload_length(0, 5);  //Set 5 byte payload length for RX data pipe 0
      nrf_set_crc_length(2);    //Set 2 bytes CRC
      nrf_enable_ack_payload(true);
      nrf_enable_dynamic_payload_length(0, true);    //Enable dynamic payload length for data pipe 0
      nrf_power_up(); //Go from powerDown state to standby
      _delay_ms(5);   //PowerDown->standby delay
}
```

```
//Splits one uint16 to a two element uint8 array
void uint16_to_uint8(uint16_t input, uint8_t output[])
{
     output[0] = (input>>8);
     output[1] = (input & 0x00FF);
}

//Combines two uint8s from an 2-element array to one uint16
uint16_t uint8_to_uint16(uint8_t input[])
{
     uint16_t output = 0;
     output |= (input[0]<<8);
     output |= (input[1]);
     return output;
}

//Converts the remote_packet to uint8 array and then transmits it, takes about 90 us when no retransmissions
void transmit_remote_packet(uint16_t remote_packet[])
{
     uint8_t temp_array[REMOTE_PACKET_LENGTH*2];
     for(int a=0; a<REMOTE_PACKET_LENGTH; a++)
     {
          uint16_to_uint8(remote_packet[a], &temp_array[a*2]);
     }
     nrf_write_tx_payload(temp_array, sizeof(temp_array));
}

//Reads RX payload and converts it to uint16 array, takes about 100 us
void read_board_packet(uint16_t board_packet[])
{
     uint8_t temp_array[BOARD_PACKET_LENGTH*2];
     nrf_read_rx_payload(temp_array, sizeof(temp_array));
     for(int a=0; a<BOARD_PACKET_LENGTH; a++)
     {
          board_packet[a] = uint8_to_uint16(&temp_array[a*2]);
     }
}

//Reads the ADC and return the value, takes about 10 us to complete
uint16_t read_adc(uint8_t adc_num)
{
     ADMUX &= 0xf0;  //Clear bits 0-3
     ADMUX |= adc_num;    //Select ADC to use
     ADCSRA |= (1<<ADSC); //Start conversion
     while((ADCSRA & (1<<ADSC)));   //Wait until conversion is ready (ADSC = 0)
     return ADC;     //Return the 10-bit value
}

//Returns true if pin is high, takes about 500ns to complete
bool read_pin(uint8_t reg, uint8_t pin)
{
     if(reg & (1<<pin))
     {
          return true;
     }
     else
     {
          return false;
     }
}

void transmit_led_control(uint8_t errors)
{
     if(errors < 15) //OK
     {
          LED_PORT |= (1<<LED1G);
          LED_PORT &= ~(1<<LED1R);
     }
     else if(errors <65)  //Some errors
     {
          LED_PORT |= (1<<LED1G) | (1<<LED1R);
     }
     else
     {
          LED_PORT &= ~(1<<LED1G);
          LED_PORT |= (1<<LED1R);
     }
}
```

```
//Blinks red if charging, otherwise shows charge status (red=empty, amber = half, green = full)
void remote_battery_led_control(uint16_t charge)
{
     static uint8_t blink_counter = 0;
     if(!read_pin(CHG_STATUS_REG, CHG_STATUS_PIN)) //If charging
     {
          LED_PORT &= ~(1<<LED2G);
          if(blink_counter > 0)
          {
               blink_counter--;
          }
          else
          {
               blink_counter = 10;
               LED_PORT ^= (1<<LED2R);
          }
     }
     else
     {
          if(charge < 33)
          {
               LED_PORT |= (1<<LED2R);
               LED_PORT &= ~(1<<LED2G);
          }
          else if(charge < 66)
          {
               LED_PORT |= (1<<LED2R) | (1<<LED2G);
          }
          else
          {
               LED_PORT &= ~(1<<LED2R);
               LED_PORT |= (1<<LED2G);
          }
     }
}

//Charge parameter value 0(empty) - 100 (full)
void board_battery_led_control(uint16_t charge)
{
     if(charge < 33)
     {
          LED_PORT &= ~(1<<LED3G);
          LED_PORT |= (1<<LED3R);
     }
     else if(charge < 66)
     {
          LED_PORT |= (1<<LED3G) | (1<<LED3R);
     }
     else
     {
          LED_PORT &= ~(1<<LED3R);
          LED_PORT |= (1<<LED3G);
     }
}

//Returns remote battery charge level as 0(empty) - 100 (full).
uint8_t read_charge_level()
{
     #define FULL_BATTERY 651  //4.2 V
     #define EMPTY_BATTERY 527 //3.4 V
     uint16_t adc_result = read_adc(LIPO_ADC_PIN);  //Returns 0-1023 as 0-6.6V
     if (adc_result <= EMPTY_BATTERY)
     {
          return 0;
     }
     else if(adc_result >= FULL_BATTERY)
     {
          return 100;
     }
     else //Convert value to 0-100
     {
          adc_result -= EMPTY_BATTERY;
          return ((adc_result * 100) / (FULL_BATTERY-EMPTY_BATTERY));
     }
}
```

RoadRunner_Protocol.h:

```
//General
#define PROTOCOL_VERSION       0

//Board
#define BOARD_PACKET_LENGTH         3
//Element indexes for data
#define BOARD_PROTOCOL_E       0
#define BOARD_FRAME_COUNT_E         1
#define BOARD_BATTERY_E             2

//Remote
#define REMOTE_PACKET_LENGTH   7
//Element indexes for data
#define REMOTE_PROTOCOL_E      0
#define REMOTE_FRAME_COUNT_E   1
#define REMOTE_ERROR_COUNT_E   2
#define REMOTE_BATTERY_E       3
#define REMOTE_DEADMANS_E      4
#define REMOTE_FUNCTION_E      5
#define REMOTE_SPEED_DIAL_E         6
```

Kauko-ohjaimen ajantasainen ohjelmakoodi: https://bitbucket.org/Miqueltozzz/roadrunner

NRF-kirjaston ajantasainen ohjelmakoodi: https://bitbucket.org/Miqueltozzz/nrf24l01p