Opinnäytetyö (AMK)

Tietotekniikka

Sulautetut järjestelmät

2010

Joni Linqvist

GPS-loggeri pelastuskoiratoimintaan

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautetut järjestelmät

Toukokuu 2010 | 47

Ohjaaja Jari-Pekka Paalassalo

Tekijä Joni Lindqvist

GPS-LOGGERI PELASTUSKOIRAN KÄYTTÖÖN

Työn tarkoituksena oli rakentaa prototyyppi GPS-loggerista, joka voidaan asentaa pelastuskoiran selkään ja näin tutkia jälkikäteen koiran liikkeitä pelastusalueella.

Prototyyppi sisältäisi GPS-moduulin, jonka tuottaman datan ATMega16L –mikrokontrolleri tallentaisi SD-muistikortille. SD-muistikortille data tallennettaisiin FAT16-tiedostojärjestelmään, jolloin sen käyttö olisi mahdollista mahdollisimman monessa erilaisessa päätelaitteessa. Prototyypin suunnittelussa kiinnitettiin erityistä huomiota virrankulutukseen sekä laitteen soveltuvuuteen erilaisissa käyttölämpötiloissa.

Protoyyppiä toteutettaessa havaittiin laitteistossa kuitenkin suunnitteluvirhe, joka esti täysin toimivan prototyypin toteutuksen. Prototyypin sijaan valmistettiin kuitenkin toimiva ohjelmointialustalla toimiva laite, joskin laite oli enemmänkin todiste konseptin toimivuudesta kuin valmis prototyyppi.

ASIASANAT:

FAT16, SD, GPS, AVR

BACHELOR´S THESIS | ABSTRACT

UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded systems

June 2010 | 47

Instructor Jari-Pekka Paalassalo

Author Joni Lindqvist

GPS-LOGGER FOR RESCUE DOG USE

Concept of this job was to create working prototype of a GPS-device that could be attached to a rescue dogs harness. This way the dogs trainer could inspect how the dog has been performing in the search area.

Prototype would include a GPS-module and created data would be collected by ATMega16L-microcontroller. Microcontroller would transfer collected data to Secure Digital memory card as a normal text file. Using FAT16 as file system of the memory card, one could be sure it would open in vast variety of every day devices including PDA's, mobile phones and computers.

Design phase of the protype concentrated mainly on power consuption and how to minimize it. Also one of the main elements of the design was to choose parts that could endure different temperatures.

Creation process of the prototype revealed one serious design flaw in the hardware and prototype created was never fully functional. Instead a proof of concept was made using a PV-M32 developement board. Created device was fully functional altough it only concetrated on writing the GPS-data to Secure Digital memory card.

KEYWORDS:

FAT, SD, MMC, GPS, AVR

Alkulause

Tämän työn tulos on monen asian summa, mutta erityisesti tähän johtaneelta taipaleelta ovat mieleeni jääneet seuraavat henkilöt joita haluan kiittää:

- Pentti Vahtera, jonka ylitsepursuava innostus opettamastaan aiheesta antoi upeat puitteet opiskelemiselle ja oivaltamisen riemulle
- JP Paalassalo, joka teki monimutkaisista asioista niin yksinkertaisia, että jopa minun kaltaiseni humanisti ymmärsi syiden ja seurausten suhteen.
- Raija Tuohi, jota ilman matematiikka olisi itselleni edelleenkin vain jännittäviä kirjainyhdistelmiä täysin epäloogisessa järjestyksessä.
- Avovaimoni Ansa Blomberg, joka ei kertaakaan (tietääkseni) suunnitellut heittävänsä tavaroitani ulos ikkunasta, vaikka asuntomme muuttui säännöllisin väliajoin allekirjoittaneen henkilökohtaiseksi elektroniikkalabraksi syövytyshappoineen ja juotosasemineen tai vaikka allekirjoittaneen kommunikaatiokyvyt madaltuivat ajoittain yksinkertaisiksi murahduksiksi ja ärähdyksiksi.
- Tärkeimpänä henkilönä haluan kiittää isääni Tapio Kuusimäkeä, jolle mikään ei koskaan tuntunut olevan mahdotonta, olipa kyse sitten puunkaatoraudan tekemisestä tai oman kiihdytysauton rakentamisesta. Vaikka kiinnostuksen kohteemme eivät samoja olekaan, epäilen vahvasti saman hulluuden ajavan meitä omiin tavoitteisiimme.

Nokialla 2.5.2010

Joni Lindqvist

SISÄLTÖ

KUVAT

TAULUKOT

Symbolit ja lyhenteet

ASCII	American Standard Code for Information Interchange. Yleisin tapa muuntaa kirjaimia numeroiksi ja toisin päin
BIOS	Basic Input Output System. Tietokoneen laitteiston käynnistyksestä vastaava järjestelmä.
CHS	Cylinder-Head-Sector. Etenkin kiintolevyjen käytössä yleinen tapa viitata fyysiseen osoitteeseen levyllä. Sittemmin korvautunut lähes täysin LBA-tyyppisellä notaatiolla.
GPS	Global positioning system. Reaaliaikainen yksisuuntainen satelliitteihin perustuva paikannusjärjestelmä
FAT	File Allocation Table. Taulukko, johon tallennetaan tiedostojen fyysinen sijainti levyllä. Samalla termillä voidaan viitata myös tiedostonhallintajärjestelmään ks. FAT16
FAT16	File Allocation Table. Microsoftin kehittämä yksinkertainen 16-bittisiin muistiosoituksiin perustuva tiedostonhallintajärjestelmä
IP67	Sähkölaitteiden tiiveyttä kuvaava eurooppalainen järjestelmä. Esimerkkinä IP67-luokka on täysin pölytiivis, sekä kestää suurella paineella suihkutettua vettä.
LBA	Logical Block Addressing. Yleisin tapa viitata massamuistilta löytyvään fyysiseen osoitteeseen.
NMEA	National Marine Electronics Associationin kehittämä kommunikointiprotokolla erilaisten merijärjestelmien (esim. sonar, autopilotti tai GPS-vastaanotin) väliseen datan siirtoon.
MBR	Master Boot Record. Kiintolevyn tai massamuistin ensimmäiset 512 B. MBR on osa IBM PC standardia ja on näin ollen arkkitehtuuririippuvainen ominaisuus.
MMC	Multimedia Card. Vanhahko muistokorttistandardi.
POC	Proof of Concept. Rajoitettu tai vaillinainen toteutus järjestelmästä, jolla osoitetaan järjestelmän pääkomponenttien toimivuus.
UART	Universal Asynchronous Receiver/Transmitter. Standardoitu asynkroninen tiedonsiirtoväylä

USART	Universal Asynchronous Receiver/Transmitter, kuin UART, mutta synkronisena. Alaspäin yhteensopiva UART:n kanssa
USB	Universal Serial Bus. Yleisesti käytössä oleva liitäntätapa tietokoneiden lisälaitteille
SD	Secure Digital. MMC:n korvannut yleisesti käytössä oleva muistikorttistandardi. Yhteensopiva MMC:n kanssa.
SDHC	Secure Digital standardin laajennus, joka tukee nopeampaa liityntää sekä kapasiteetiltaan suurempia kortteja.
SPI	Serial Peripheral Interface Bus. Motorolan kehittämä synkroninen tiedonsiirtoväylä.

Johdanto

Tämän työn lähtökohtana oli toteuttaa GPS-tallennin, joka soveltuisi mahdollisimman hyvin ulkokäyttöön. Idea laiteen toteuttamiseen heräsi kun huomasin tarvitsevani tietoa pelastusetsintää harjoittelevan koiran liikkeistä maastossa. Vastaavia projekteja on toteutettu aiemminkin, joten valmista lähdemateriaalia löytyi jonkin verran. Kuitenkaan yksikään toteutustapa ei täysin vastannut omia lähtökohtiani, joten tietoja ei voitu käyttää suoraan .

Laitteen tärkeimpiä vaatimuksia olisi ehdottomasti kohtuullisen pieni koko. Laitteen rakenteen pitäisi kestää iskuja ja hyvin erilaisia sääolosuhteita. Myös akun kestoon tuli kiinnittää huomiota, sillä jos laitetta käytettäisiin todellisessa pelastusetsinnässä, pitäisi akkujen kestää käyttöä vähintään 8 h ajan. Olisi myös tärkeää, että laitteen valmistuskustannukset pysyisivät kohtuullisina, jotta laitteen rikkoutuminen ei aiheuttaisi käyttäjälle merkittävää taloudellista tappiota.

Laitteen passiivisen luonteen vuoksi ei siihen ole kovinkaan perusteltua lisätä varsinaista käyttöliittymää tai nykyisestä sijainnista kertovaa näyttöä. Laite sen sijaan tallentaisi jatkuvasti koiran sijaintia massamuistiin, jolta pystyttäisiin myöhemmin piirtämään koiran kulkema reitti karttapohjan päälle ja analysoimaan koiran käytöstä maastonmuotojen tai hajujälkien perusteella. Todellisessa pelastusetsinnässä laitteen antamilla tiedoilla voitaisiin varmistua, ettei koira ole jättänyt kuoppia ja kallionkoloja tutkimatta ja näin ollen varmistua siitä, että koiran etsimä alue on oikeasti tutkittu.

Koska laitteen käyttäjäkuntaan tulisi kuulumaan teknisiltä taidoiltaan hyvin eritasoisia ihmisiä piti, tietojen saaminen laitteelta tehdä hyvin yksinkertaiseksi. Jos laite pitäisi kytkeä tietokoneeseen erillisellä kaapelilla vaatisi se käytännössä USB-kaapelin käyttöä. USB-järjestelmän monimutkainen toteutus tekisi lähtökohdasta helposti hyvin kalliin tai vaihtoehtoisesti ohjelmistoltaan kohtuullisen monimutkaisen ratkaisun. Yksinkertaisin vaihtoehto olisi käyttää standardia noudattavaa irroitettavaa massamuistiratkaisua. MMC/SD-kortti olisi

yksinkertaisen ja avoimen kytkentärajapintansa vuoksi juuri tällainen täydellinen ratkaisu. Kun paikkatiedot tallennettaisiin kortille FAT16-tiedostojärjestelmään, voitaisiin olla myös täysin varmoja siitä, että kortin sisältämän datan saa käytännössä auki millä tahansa älylaitteella tai tietokoneella, joilla myös karttoja pystytään esittämään.

Koska valmiita kirjastoja FAT-tiedostojärjestelmän käsittelyyn on useita, en pitänyt perusteltuna toteuttaa tiedostojärjestelmän käsittelyä itse. Aihe olisi ollut niin laaja, että sen toteuttaminen olisi helposti ollut yhtä monimutkainen prosessi, kuin itse loppujärjestelmän tekeminen.

Tässä opinnäytetyössä selvitän ensin millaiseen tulokseen tähdättiin ja mitä otettiin huomioon laitetta suunniteltaessa. Luku 2 (Käytetyt tekniikat) pyrkii antamaan lukijalle summittaisen kuvan siitä, millaisten tekniikoiden parissa opinnäytetyötä tehtiin. GPS-tekniikka rajattiin pois teoriaosuudesta, sillä nykyiset kehittyneet GPS-moduulit eivät vaadi sen käyttäjiltä lainkaan tietoa sen toimintaperiaatteista.

Luvut 3 Sähköisten osien valinta sekä 4 Ohjelmisto keskittyvät laitteen sähköisten ja ohjelmallisten komponenttien yhteistoimintaan ja niihin seikkoihin, jotka vaikuttivat komponenttien suunnitteluun ja valintaan. Luku 5 Toteutus ja testaus käsittelee ensisijaisesti laitteen toteutuksessa kohdattuja ongelmia ja pohditaan, miksi laitteen komponentit eivät toimineet niin kuin oletin. Luku 6 (Yhteenveto) keskittyy yksinomaan pohtimaan miksi valmis laite ei lopulta toiminut kuten haluttiin ja mitkä yksittäiset virheet suunnitteluprosessissa aiheuttivat nämä ongelmat.

.

Käytetyt tekniikat

Koska eri moduulien välisessä tiedonsiirrossa käytettiin yksinomaan yleisesti tunnettuja yksinkertaisia väyläratkaisuja, päätettiin tekniikoiden syvällinen toimintaperiaate rajata tämän opinnäytetyön ulkopuolelle. Kaikista käytetyistä väyläratkaisuista löytyy kuitenkin paljon helposti omaksuttavaa informaatiota lukuisista eri lähteistä.

UART

GPS-moduuli kommunikoi mikrokontrollerin kanssa UART-väylän kautta. UART on hyvin yksinkertainen asynkroninen väylätyyppi. UART muuntaa rinnakkaisessa muodossa olevan datan siirtorekisterin avulla sarjatyyppiseksi dataksi ja lähettää sen eteenpäin [1]. Vastaanottavan pään UART taas kokoaa datan sarjamuotoisesta datasta takaisin rinnakkaismuotoiseksi, jonka jälkeen data voidaan tallentaa muistipaikkaan esimerkiksi tavuna [1].

SPI

SPI-väylää käytetään tässä työssä mikrokontrollerin ja muistikortin välisessä tiedonsiirrossa. Vaikka SPI-väyläkin on UARTin tapaan sarjaväylä on se kuitenkin huomattavasti UARTia kehittyneempi. SPI koostuu neljästä signaalista, joista yksi määrittää laitteiden isäntä-orja-jaon, yksi hoitaa kellosynkronoinnin ja kaksi signaalia on varattu tiedonsiirtoa varten (data in / data out) [2]. Datan lähetys on kaksisuuntaista ja väylän jokaisella kellopulssilla voidaan vastaanottaa tai lähettää dataa [2].

2.1 FAT-tiedostojärjestelmä

FAT-tiedostojärjestelmä jakaantuu useisiin eri versioihin, joista tässä työssä keskitytään olennaisesti FATin 16-bittiseen versioon eli FAT16:een. FATistä löytyy myös 12- ja 32-bittiset versiot, joista ensin mainittua käytetään lähes pelkästään levykkeiden tiedostojärjestelminä. Jälkimmäinen taas on jonkin verran FAT16:sta

monimutkaisempi järjestelmä, joka sisältää mm. 256-merkkiset tiedostonimet sekä yli 2 GT:n kokoiset levyosiot. Kummallekaan ominaisuudelle ei työssäni ollut käyttöä. FAT32:ssa on myös jossain määrin epäselvä lisensointi, joka ei myöskään ole kovin suuri kannuste kyseisen tiedostojärjestelmän käyttöön [4].

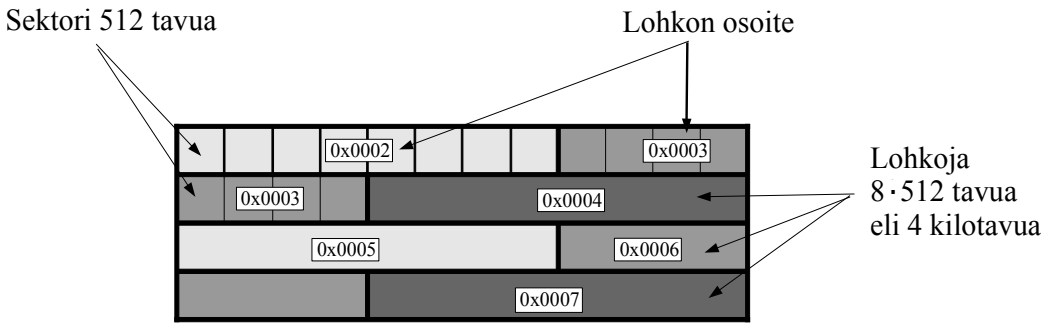
Seuraavassa alalluvussa käsitellään FAT16:n toiminta pääpiirteittäin. FAT16:n partitiojärjestelmään ei syvennytä kovin perusteellisesti, sillä massamuistia ei ole tarkoitus partitioida tai alustaa järjestelmässä, jolloin partitioiden luomiseen ei ole kovinkaan perusteltua paneutua. Myös Master Boot Recordiin (MBR) liittyviä yksityiskohtia ainoastaan sivutaan muutamassa yhteydessä. Hieman tavallisuudesta poiketen FAT-tiedostojärjestelmä käyttää tavujärjestyksenä käänteistä merkitsemismuotoa (little-endian), jossa vähiten merkitsevät bitit tallennetaan tavun alkuun[4]. Näin ollen esimerkiksi numero 5 olisi little-endian merkitsemistavalla 011 eikä totuttuun tapaan 110.

2.1.1 Osoiteavaruus

Koko levyjärjestelmä pohjaa sektoriajatteluun, jossa koko muistialue jaetaan saman suuruisiksi sektoreiksi (vrt. levyke tai kiintolevy). Sektorille on FAT16:ssa määritelty kiinteä 512 B:n sektorikoko. Jos jokainen sektori vastaisi yhtä muistiosoitetta olisi levyn maksimikoko ainoastaan 32 MB. Tästä syystä sektorit kootaan vielä lohkoiksi (Microsoft käyttää lohkosta nimitystä cluster), jotka pitävät sisällään tietyn määrän sektoreita. Lohkon suuruus vaihtelee kahden ja 32 KB:n välillä. Lohkoilla on kuitenkin maksimimääränsä, joka FAT16:sta tapauksessa on 2^{16} eli 65 536. Näistä tavuista kuitenkin menetetään muutama FATin levyjärjestelmän kirjanpitoon, joten todellinen enimmäismäärä on 65517 B. Tiedosto järjestelmän enimmäiskoko pystytään siis laskemaan kaavalla: lohkojen maksimimäärä × lohkon maksimikoko (65517 × 32 kB = 2096576 kB ~ 2 GB). Todellisuudessa osoiteavaruudesta menetetään muutama tavu. Itse osoite muodostuu pelkästään lohkon järjestysnumerosta, eikä varsinaisilla sektoreilla tehdä mitään normaalissa levyoperaatiossa (kuva 1). [3]

Lohkon koon kasvu aiheuttaa paljon hukkatilaa etenkin pienikokoisissa tiedostoissa. Koska levyjärjestelmässä voidaan viitata ainoastaan yksittäiseen

lohkoon, on tällöin tiedoston pienin mahdollinen koko suurimmalla lohkokoolla minimissään 32 kB riippumatta siitä mikä tiedoston koko fyysisesti on levyllä. Nykyisillä tallennuskapasiteeteilla ja tiedostoilla ongelma on kuitenkin varsin marginaalinen.



Kuva 1: Esimerkki sektorien ja lohkojen jakautumisesta muistialueelle. Lohkojen osoitteet alkavat aina osoitteesta 0x002, sillä lohkoja edeltävä osa on varattu massamuistin ja tiedostojärjestelmän määrittelyille.

2.1.2 Tiedostojen varaustaulukko (FAT)

FAT (file allocation table) eli tiedostojen varaustaulukko perustuu nimensä mukaan taulukkoon, johon on tallennettu jokaisen tiedoston fyysinen olinpaikka levyllä. Järjestelmä on itseasiassa hyvin yksinkertainen. Taulukon jokaista solua vastaa yksi lohkon osoite, johon on liitetty kyseisen lohkon arvo. Lohkon arvoja tutkimalla saadaan esimerkiksi selville onko kyseinen lohko vapaana tai mahdollisesti vioittunut.

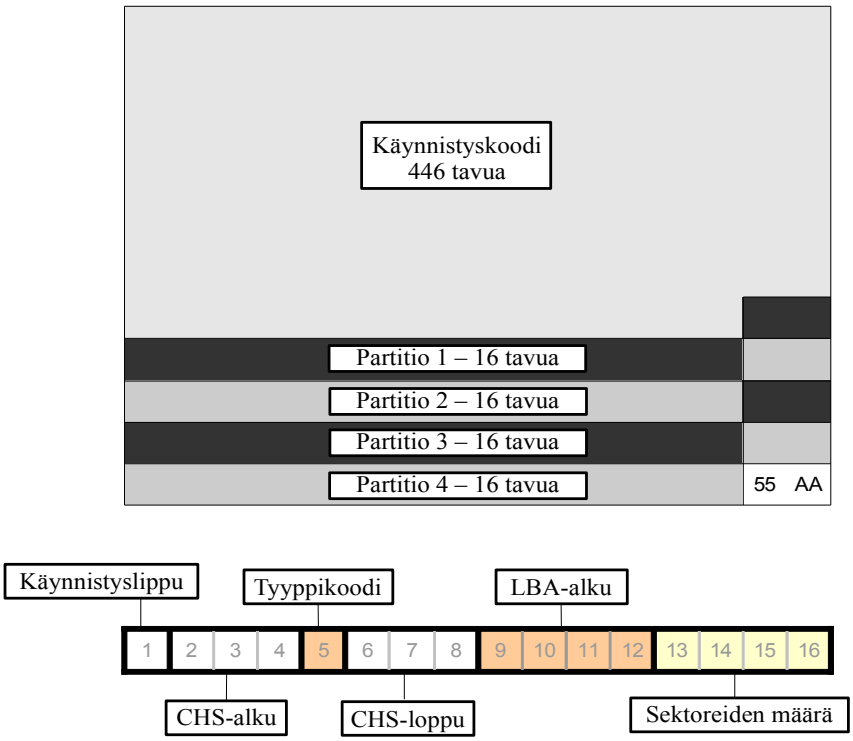
Lohkojen on mahdollista saada erilaisia arvoja (ks. Taulukko 1). Tiedostot kuvataan varaustaulukossa ketjuttamalla. Jos lohko on varattu, on siihen kirjoitettu tiedoston seuraavan lohkon osoite, tai jos lohko on tiedoston viimeinen, siihen on kirjoitettu tiedoston loppumisesta kertova arvo. Vapaalla lohkolla on niin ikään oma arvonsa ja fyysisesti vioittuneella tai muuten varatulla lohkolla omansa.

Taulukko 1: FAT-taulukon eri arvot [4].

Lohkon arvo	*Kuvaus*
0x0000	Vapaa lohko
0x0001	Arvo on varattu muuhun käyttöön
0x0002–0xFFEF	Lohko käytössä. Lohkon arvo osoittaa seuraavaan tiedoston käyttämään lohkoon
0xFFF0–0xFFF6	Arvo on varattu muuhun käyttöön
0xFFF7	Lohkossa yksi tai useampi vioittunut sektori tai lohko on varattu muuhun käyttöön
0xFFF8-0xFFFF	Tiedoston viimeinen lohko

2.1.3 Käynnistyssektori

Käynnistyessään levyjärjestelmä tarvitsee tiedon siitä, mistä tietokoneen käynnistymiselle kriittiset osat löytyvät. Levyjärjestelmän ensimmäiset varausyksiköt on varattu tätä tarkoitusta varten. Levyn ensimmäinen alue on käynnistyssektori (tästä eteenpäin MBR), joka pitää sisällään tietokoneen käynnistämiseen tarvittavan koodin. MBR:n ensimmäiset 446 B liittyvät ensisijaisesti tietokoneen käynnistykseen. Tämän jälkeen tulee 64 B kokoinen alue, joka pitää sisällään partitioinformaatiot (ks. kuva 2). MBR:ssä kiinnostaa ensisijaisesti partitiolohkon viides tavu, joka pitää sisällään käytettävän levyjärjestelmän tyypin sekä LBA:n aloitusindeksin. Myös sektoreiden määrä voidaan lukea muistiin, jotta voidaan olla varmoja, ettei kirjoitus tapahdu väärällä partitiolla. MBR:n kaksi viimeistä tavua sisältävät lopetuskoodin jonka sisältö on aina 0x55AA.



Kuva 2: MBR:n sisältö, jonka alla partitiolohkon sisältö

2.1.4 Partition käynnistyslohko

MBR:n osoittama LBA-osoite kertoo partition aloituskohdan, jonka alussa on aina partition oma käynnistyssektori. Käynnistyssektorin koko on 512 B ja se sisältää mm. käyttöjärjestelmän bootstrapin, tai ainakin osan siitä (jos levyä käytetään käyttöjärjestelmän käynnistämiseen). Levyltä lukemisen kannalta tärkeimmät asiat, eli partition lohkojen koko, sektorin koko, juurihakemiston sijainti jne. löytyvät myös käynnistyslohkolta. Taulukko 2 kertoo lukemisen ja kirjoittamisen kannalta olennaisten tietojen löytämisen.

Taulukko 2: Partition käynnistyslohkon olennaisimmat osoitteet

Osoite	*Pituus*	*Kuvaus*
0x0B	2 tavua	Sektorin pituus bitteinä (lähes aina 512)
0x0D	1 tavu	Sektoreiden määrä lohkossa (1,2,4,8,16 tai 32)
0x0E	2 tavua	Varattujen sektorien lkm. (käytetään laskettaessa FATin sijainti)
0x10	1 tavu	FATien lukumäärä (yleensä kaksi)
0x11	2 tavua	Juurihakemiston tiedostojen maksimimäärä (yleensä 512)
0x13	2 tavua	Sektoreiden lukumäärä (kun median koko pienempi kuin 32Mb)
0x16	2 tavua	FATin varaamien sektorien määrä
0x20	4 tavua	Sektoreiden lukumäärä (kun median koko suurempi kuin 32Mb)
0x36	8 tavua	Levyjärjestelmän tyyppi (pitäisi olla FAT16, joskaan kaikki FAT16-ajurit eivät käytä kenttää)

2.1.5 Juurihakemisto

Juurihakemiston alku löytyy laskemalla ensin FATin alkamisosoite (määritetty käynnistyssektorissa osoitteessa 0x0E) ja lisäämällä siihen käytettyjen FATien määrä (partition käynnistyssektorin osoite 0x10) × FATin vaatima tila (partition käynnistyslohkon osoite 0x16). Juurihakemisto on partitiosektorissa määritetyn pituinen (yleensä 512 tiedostoa, partition käynnistyssektorin osoite 0x11) ja se sisältää kaikki juurihakemiston alihakemistot ja tiedostot. Tiedostojen nimet, ominaisuudet ja osoitteet tallennetaan 32-tavuiseen tiedostoalkioon (taulukko 3). Tiedoston ominaisuuksia määrittelee yksi tavu (0x0B), jossa tiedoston mahdolliset ominaisuudet ilmaistaan lippuina tavussa (taulukko 4). Juurihakemisto koostuu 512:sta 32 B:n mittaisesta tiedostoalkiosta. Se ei voi koskaan olla pidempi kuin partitiolohkossa määritelty maksimipituus.

Tiedoston nimiä tutkittaessa on huomattava, että tiedostonimen ensimmäistä tavua käytetään käytetään myös kertomaan tiettyjä alkion parametreja. Jos tiedostoalkion ensimmäinen tavu on:

- 0x00, on kyseinen alkio hakemiston viimeinen.
- 0x5E, on kyseinen alkio vapaa (tiedosto, joka on alunperin ollut alkiossa on poistettu).

Lisäksi tiedostonimi ei saa alkaa ASCII-merkillä 0x20 (välilyönti). Yhteensopivuuden vuoksi hyväksyttyjen merkkien tulisi olla: suurina kirjoitetut A-Z, numerot 1-9 sekä seuraavat erikoismerkit: #, $, %, &, ', (,), -, @.

Taulukko 3: Tiedostoalkion rakenne ja osoitteet [5]

Osoite	*Pituus*	*Kuvaus*
0x00	8 tavua	Tiedostonimi
0x08	3 tavua	Tiedostopääte
0x0B	1 tavu	Ominaisuusbitti
0x0C	1 tavu	Varattu Windows NT:lle
0x0D	1 tavu	Luotu (millisekunnit)
0x0E	2 tavu	Luomisaika
0x10	2 tavua	Luomispäivämäärä
0x12	2 tavua	Viimeksi käytetty
0x14	2 tavua	Varattu FAT32:ta varten
0x16	2 tavua	Viimeisin aika, jolloin tiedostoon kirjoitettu
0x18	2 tavua	Viimeisin päivämäärä, jolloin tiedostoon on kirjoitettu
0x1A	2 tavua	Alkamislohkon osoite
0x1C	4 tavua	Tiedoston koko tavuina

Lukemalla juurihakemiston tiedostoalkio voidaan selvittää halutun tiedoston alkamissektori, minkä jälkeen varaustaulukkoa hyväksi käyttämällä saadaan tietoon kaikki tiedostolle kuuluvat sektorit.

Taulukko 4: Tiedoston ominaisuustavun lippujen merkitykset [5][6]

Bitti	*Kuvaus*
0	Tiedostoa voi ainoastaan lukea (read only)
1	Tiedosto on piilotettu
2	Tiedosto kuuluu käyttöjärjestelmään
3	VolumeId/partition nimi (bitin asettaminen sallittu ainoastaan juurihakemistossa)
4	Tiedosto on hakemisto
5	Arkistointilippu, tiedosto muuttunut edellisen varmuuskopion jälkeen
6	Ei käytössä
7	Ei käytössä

Alihakemistoa varten levyltä varataan yksi sektori, joka tyhjennetään kaikista mahdollisista tiedostoalkioista. sektorin alkuun lisätään kaksi tiedostoa (**.** ja **..**), joista ensimmäinen osoittaa alihakemistoon itseensä ja toinen alihakemiston kantahakemistoon. Tämän jälkeen jokaiselle alihakemistoon kuuluvalle tiedostolle kirjoitetaan oma tiedostoalkionsa. Jos sektorin 512 tavua loppuvat kesken

varataan seuraava sektori ja jatketaan samalla tavoin. Varaustaulukkoon kirjoittamisessa ei hakemisto eroa millään tavoin normaalista tiedostosta. Hakemistolistauksen päättyminen ilmoitetaan tiedoston tavoin arvolla 0xFFFF ja useammalle sektorille levittäytyvien hakemistojen arvot viittaavat aina siihen sektoriin, jonne listauksen seuraavat tiedostoalkiot on kirjoitettu.

On kuitenkin erittäin tärkeää ymmärtää, että tiedostojärjestelmä itsessään on pelkästään levyjärjestelmän sisällön luokittelemista helpottava abstraktio, jonka tarkoitus on helpottaa tiedostojen ryhmittelyä sekä antaa niille nimi. Levyjärjestelmän lukeminen ja kirjoittaminen onnistuu myös ilman tiedostoabstraktiota pelkkää varaustaulukkoa hyödyntämällä. Tällöin tiedostot voidaan erottaa toisistaan ainoastaan sisältöä lukemalla, mutta tiedostojärjestelmän toimintaan sillä ei ole vaikutusta. Jokaisen tiedoston osien sijainti ja tiedoston koko voitaisiin myös selvittää pelkkää varaustaulukkoa lukemalla.

2.2 SD–kortin käsittely SPI-tilassa

SD-kortin käyttö SPI-tilassa on suoraviivaista ja yksinkertaista. Kommunikaatio tapahtuu sarjadatalla ja datan vastaanottoon ja lähetykseen käytetään omia erillisiä linjojaan.

Kommunikaatio kortille tapahtuu tietyntyyppisten käskyjen avulla, jossa datapaketti koostuu 2 b:n mittaisesta alustuksesta, 6 b:n mittaisesta käskyosuudesta, 32 b:n mittaisesta argumenttiosuudesta sekä 2 b:n mittaisesta tarkistussummasta. Tarkistussumman käyttö SPI-tilassa ei kuitenkaan ole välttämätöntä. [16] [17]

Käsky lähetetään kortille ja käskyn lähettämisen jälkeen linja nostetaan ylös. Tämän jälkeen odotetaan kortin lähettämää 8-bitin mittaista vastausta. Vastausta tutkimalla voidaan päätellä onko käskyn toteutus onnistunut tai onko käskyn toteutus vielä kesken. [16] [17]

SD- ja MMC-korttien käskykanta on yleisesti saatavilla ja löytyy esimerkiksi lähdeluettelosta löytyvästä SD-kortin spesifikaatiosta.

2.3 NMEA 0183

NMEA 0183 on määrittely, joka on kehitetty alun perin merellä käytettävien sähkölaitteiden väliseen kommunikointiin. Määrittely pitää sisällään niin rautatason liitäntäprotokollan kuin ASCII-koodauksen perustuvan viestitason.[18]

Bittitason liityntä hoidetaan sarjaliitännällä, jonka nopeus on 4 800 bps. Datarakenteessa on 8 databittiä sekä yksi lopetusbitti. Pariteettitarkastusta ei ole eikä myöskään kättelyä. [18]

Laitteen lähettämä viesti koostuu maksimissaan 84 B:n mittaisesta ASCII-koodatusta viestistä. Viesti alkaa aina dollarimerkillä, jonka jälkeen vuorossa on lähettäjälaitteen nimi sekä informaatio, jota laite lähettää. Viesti päättyy aina kahteen rivinvaihtomerkkiin (<CR><LF>), joita edeltää asteriskilla(*) alkava kolmen tavun mittainen tarkistussumma heksadesimaalimuodossa (esim. *3F). [19]

Esimerkiksi seuraavanlainen NMEA-viesti kertoo lähettäjän olevan ”GP” ja viestin kertovan kyseisen laitteen sijainnin (GGA). Pilkut erottavat tietoalkiot toisistaan kaikissa NMEA-viesteissä.

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
```

Kyseinen esimerkki kertoo mm. sijainnin rekisteröintiajan (12:35:19 UTC), laitteen latitudin (4807.038 N) ja longitudin (01131.000 E) sekä signaalin sen hetkisen tilan (1 = laitteella on yhteys GPS-satelliittiin).

NMEA-standardiin kuuluu useita kymmeniä erilaisia viestejä, joiden tietosisältö vaihtelee UTC-ajasta autopilotin ohjauskomentoihin. Vaikka NMEA 0183 -standardi onkin maksullinen, löytyy internetistä lukuisia sivuja, jotka listaavat kaikki NMEA-viestityypit.

Sähköisten osien valinta

Tämä luku käsittelee laitteiston suunnitteluprosessia ja valintakriteereitä. Koska laite tulisi olemaan käytössä hyvin erilaisissa sääolosuhteissa, se asetti komponenteille joitakin erityisvaatimuksia.

Komponenttivalinnat pohjautuvat pitkälti pääkomponenttien referenssipiirroksiin, joten laskemista vaativaa elektroniikkasuunnittelua ei työssä jouduttu tekemään. Jonkin verran laitteen elektronisia komponentteja sivutaan myös luvussa 4 Ohjelmisto.

3.1 GPS-vastaanotin ja antenni

Markkinoilla on satoja erilaisia GPS-vastaanottimia, joiden ominaisuudet vaihtelevat hyvinkin paljon. Ensisijaisena lähtökohtana olisi valita vastaanotin, jossa olisi oma sisäinen laskentalogiikkansa, jolloin vastaanotin tarjoaisi tulokset suoraan NMEA-muodossa. Koska GPS:n vaatimat laskutoimitukset vaativat monimutkaisia trigonometrisia laskutoimituksia, vaadittaisiin tällöin myös mikrokontrollerilta tiettyjen laskutoimitusten nopeaa laskemista [7]. Vastaanottimen tulisi myös kestää vaihtelevia lämpötiloja, erityisesti pakkasta. Myös vastaanottimen virrankulutus tulisi olla mahdollisimman pieni mutta herkkyys pitäisi samaan aikaan saada pidettyä mahdollisimman korkeana. Tämä rajasi tehokkaasti valittavana olevia GPS-vastaanottimia. Lisäksi vaatimus, että vastaanotin pitäisi pystyä hankkimaan suomesta yrityksestä, joka myisi komponentteja myös yksityishenkilölle, rajasi vaihtoehdot lopulta kouralliseen erilaisia vastaanottimia.

Rajausten jälkeen valittavana oli GlobalSatin ET- ja ES-sarja, josta löytyi vastaanottimia hintaluokasta 25 € - 100 € . Suurin ero vastaanottimien välillä oli lähinnä siinä oliko vastaanottimissa sisäänrakennettu antenni, antenniliitin vai pitikö antennin kytkentä ja siihen vaadittu elektroniikka toteuttaa itse ja pohjautuiko toteutus SiRFStar III –piiriin. SiRFStar III on SiRF Technology –yhtiön kehittelemä järjestelmä, jossa on kiinnitetty erityistä huomiota vastaanottimen herkkyyteen ja

kykyyn säilyttää satelliittilukitukset myös haastavissa olosuhteissa kuten metsissä ja korkeiden talojen läheisyydessä [8]. Oli siis melko selvää, että valinta kohdistui nimenomaan SiRFStar III –pohjaiseen vastaanottimeen. Vaatimukseni pakkasenkestolle rajasi vastaanottimia vielä lisää. GPS-antennien hintoja tutkiessani huomasin perusantennien hintojen olevan 10 – 20 € välissä. Erotus antennittoman ja halvimman antennillisen GPS-moduulin välillä kasvoi kuitenkin yli 30 €:oon, joten erillisellä antennilla varustettu GPS-moduuli tulisi huomattavasti edullisemmaksi. Päädyin lopulta GlobalSatin ET-312 vastaanottimeen (liite 2). Vastaanotin syöttäisi paikannusdatan UART-väylää pitkin NMEA-muodossa. Tällöin tiedon vastaanottaminen olisi helppoa kunhan mikrokontrolleri pitäisi sisällään valmiin liitännän UARTille.

Vastaanottimen sähköiset arvot ovat varsin hyvät, eikä vastaanottimen käyttämä virtakaan aiheuttanut ongelmia. Vastaanottimen heikkoutena voidaan todeta sen kohtuullisen pienitoleranssinen käyttöjännitevaatimus 3,3 V ±5%. Vastaanotin ei myöskään sisältänyt antenniliitintä, joten sähköinen toteutus olisi hoidettava itse. Joskin referenssisuunnitelmasta selvisi kuitenkin, ettei sähköinen toteutus vaatinut muuta kuin yhden ulkoisen kelan.

Antennin valinta oli myös kohtuullisen suoraviivaista. Markkinoilla on useita edullisia 3,3 V käyttöjännitteellä toimivia aktiiviantenneja, joiden liittäminen järjestelmään olisi yksinkertaista. Koska laite tultaisiin sijoittamaan koiran selkään, sopisi löytämäni mikroliuska-antenni (patch antenna / microstrip antenna) myös suuntakuvionsa puolesta erinomaisesti laitteeseen. Jos laite kytkettäisiin koiran kaulapantaan saattaisi mikroliuska-antennin suuntakuvio muodostaa ongelman, sillä sen suuntakuvio muistuttaa väärin päin käännettyä sydäntä. Laitteen valuessa esimerkiksi koiran kaulan alapuolelle antennin tehokkain vastaanottosuunta osoittaisi silloin suoraan maahan. Tätä ongelmaa ei kuitenkaan ole, sillä jokainen laitetta käyttävä koira suorittaa etsinnän erikoisvaljaissa, jolloin laite voidaan kiinnittää valjaisiin koiran selkään.

3.2 Massamuisti

Halusin laitteen olevan mahdollisimman helppokäyttöinen ja yhteensopiva mahdollisimman monen eri käyttöjärjestelmän ja laitteiston kanssa. Vaikka ratkaisuja kerättävän datan säilömiseen olisi vaikka kuinka paljon, oli ensimmäinen ongelmani pikemminkin siinä, miten datan saisi ulos laitteesta. Kaikkein helppokäyttöisin tapa olisi varmasti USB-liitännän käyttäminen, mutta koska USB-väylän implementointi vaatii huomattavan määrän komponentteja ja ohjelmistoa ympärilleen, ei se tuntunut mielekkäältä ratkaisulta [10].

Toinen vaihtoehto olisi sarjaportin käyttäminen. Mutta koska sarjaportti on nykyisistä tietokoneista lähes täysin kadonnut, vaatisi se käytännössä käyttäjältä erillisen USB-konvertterin hankkimista. Lisäksi datan siirtämiseksi sarjaportin ylitse tarvitaan erillinen ohjelma, jolloin vaatimus laitteen käyttöjärjestelmäriippumattomuudesta ei toteutuisi.

Yksinkertaisin ratkaisu on käyttää MMC– tai SD–muistikorttia. Molemmat muistikorttistandardit ovat pinniyhteensopivia toistensa kanssa ja lisäksi molemmat tukevat ainakin teoriassa yleisesti mikrokontrollerien hyödyntämää SPI-standardia datan välitykseen [11]. Jotta yhteensopivuus saataisiin maksimoitua, kirjoitetaan data kortille FAT16-tiedostojärjestelmään, jolloin tiedoston saa käytännössä auki missä tahansa laitteessa, jossa on MMC– tai SD–korttipaikka. FAT16:n kirjoittamiseen on tarjolla myös useita laitteistoriippumattomia kirjastoja, jolloin ohjelmiston toteutus olisi myös hyvin suoraviivaista.

3.3 Mikrokontrolleri

Koska laitteelta ei vaadittaisi kovinkaan suurta suorituskykyä, vaikutti itselleni tuttu Atmelin ATMega–sarja täyttävän kaikki olennaiset vaatimukset.

ATMega sarjaan kuuluu kymmeniä erilaisia piirejä, joissa muistien määrä (SRAM ja Flash) sekä niihin integroitujen oheiselektroniikan taso vaihtelee erittäin paljon. Tosin GPS:n määrittämä 3,3 V:n käyttöjännite rajasi sopivia kontrollereja melko selvästi. Lisäksi MMC:n tai SD:n käsittelyyn vaadittava yli 512 B:n keskusmuistin vaatimus rajasi mahdollisia kontrollereita vielä lisää.

Kaikki edellä mainitut seikat huomioon ottaen päädyin ATMega 16L:ään, jossa oli 1 kB keskusmuistia ja 16 kB flash-muistia. Lisäksi se tarjosi suoran tuen UART- ja SPI-liitännöille, joten datan siirtäminen oheislaitteille hoituisi kontrollerin omilla liitännöillä. Lisäksi ATMega 16L tarjosi laajan käyttöjännitteen välillä 2,7 – 5,5 V. Hiljattain Atmel muutti mallimerkintöjään, ja vastaava mikrokontrolleri löytyy nykyään mallimerkinnällä ATMega16A [20].

3.4 Virransyöttö

Virtalähteeksi valittiin 2 AAA-kokoista paristoa. Laitteen käyttöä ajatellen olisi tärkeää, että virran loppuessa esimerkiksi pitkissä etsinnöissä olisi virtalähteen vaihtaminen mahdollisimman helppoa. Paristojen saatavuus on hyvä, eikä järjestelmään tarvitsisi kehittää minkäänlaista latausjärjestelmää, jonka sisäänrakennettu akku olisi vaatinut.

Virran syöttöön etsittiin regulaattori tai pikemminkin jännitepumppu tai step-up – regulaattori, joka pystyisi muuntamaan paristojen antaman 1 – 3 V:n jännitteen järjestelmän vaatimaksi 3.3V käyttöjännitteeksi. Lisäksi regulaattorin hyötysuhde tulisi olla mahdollisimman korkea virran kulutuksen minimoimiseksi. Tällaisen regulaattorin löytäminen ei ollut kovinkaan helppoa, sillä ensialkuun rajattiin pois kaikki MSOP-kotelointia käyttävät komponentit niiden hankalan juotettavuuden vuoksi (MSOP-koteloinnissa pinnien jalat ovat n. 0.3mm päässä toisistaan). Valitettavasti tälläisia step-up -regulaattoreita en löytänyt, joten virransyöttöön valittiin MSOP8-koteloitu National semiconductorin valmistama LM2621. Se oli ainoa muut vaatimukseni täyttävä komponentti.

Step-up -regulaattori LM2621

LM2621 on ns. step-up -regulaattori, jonka toiminta perustuu kelaan varattavaan energiaan. Varattava energia voidaan tämän jälkeen muuntaa halutunlaiseksi jännitteeksi. Piirin etuna omassa sovelluksessani oli erinomainen, vähintään

80 %:n hyötysuhde myös suuremmilla kuormilla. Lisäksi se pystyisi ylläpitämään antojännitettä jopa 0,6 V:n sisääntulojännitteellä, tällöin tosin ampeerimäärät jäävät hyvin alhaisiksi. Tämä kuitenkin riittäisi pitämään mikrokontrollerin käynnissä ja kertomaan patterin loppumisesta käyttäjälle. [12] [13]

Regulaattorin antojännitettä säädetään regulaattorin yhteyteen liitettävillä vastuksilla. Jännitteen voi säätää välille 1.24-14V. Optimiolosuhteissa regulaattori pystyy syöttämään jopa 1 A:n virtaa.[12]

3.5 Kotelointi

Itse kotelon valinta rajattiin pois tästä työstä. Toimiva, valmis laite vaatisi tietysti IPV67-suojasuojaluokituksen omaavan kotelon, mutta suunnitteluvaiheessa pidin normaalia, pehmustettua pakasterasiaa toimivana kotelona. Se oli erittäin helppo aukaista, oli kohtuullisen kestävä ja pitäisi veden loitolla erinomaisesti. Lopulta prototyyppi ei joutunut ulkotilaan kuin kerran, jolloin pakasterasia täytti kaikki ennakko-odotukset.

Ohjelmiston suunnittelu

Ohjelmiston ytimenä toimisi Roland Riegelin kirjoittama ja ylläpitämä MMC/SD/SDHC card library, joka toteuttaa FAT-levyjärjestelmän hallinnan MMC, SD tai SDHC muistikorteille. Kirjasto tukee FAT16- ja FAT32-levyjärjestelmiä, joista ainoastaan FAT16-tukea tultaisiin käyttämään tässä työssä. FAT32 mahdollistaisi yli 2 Gt:n kokoisten korttien käytön. Mutta koska tallennettavaa dataa kertyy enimmillään 300 kt tunnissa (NMEA maksimiviestin pituus (82 merkkiä) x paikkatietopäivitysten määrä tunnissa (3600) = maksimidata), tallentaisi jo 2 GB muistikorttiin yli puoli vuotta yhtäjaksoista paikkadataa.

Kirjasto itsessään jakautuu kahteen osaan, joista ensimmäinen osa keskittyy FAT-levyjärjestelmän hallintaan ja toinen osa on ajuri MMC/SD/SDHC-korttien lukemiseen AVR-ympäristössä. Kirjaston käyttäjän kannalta sen ajuriosuus ei juuri vaikuta ohjelmointiin, mutta kirjaston porttaaminen muihin järjestelmiin onnistuisi ainoastaan ajurin uudelleenkirjoituksella.

Itse kirjoittamani koodin osuus järjestelmässä keskittyisi virranhallintaan, GPS:ltä tulevan paikkadatan siirtämiseen muistikortille sekä järjestelmän reaaliaikavaatimusten hallintaan.

4.1 Virranhallinta

Yksi FAT-tiedostojärjestelmän heikkouksista on sen äärimmäisen huono virheensietokyky. Jos virrat esimerkiksi katkeaisivat kesken kirjoitusoperaation, voisi pahimmassa tapauksessa koko tiedostojärjestelmä rikkoutua. Näin ollen esimerkiksi virtakytkin pitäisi toteuttaa siten, ettei se katkaisisi virtoja suoraan, vaan virtakytkimeen reagoitaisiin järjestelmän hallitulla alasajolla.

Koska valitun piirin virransäästöominaisuudet ovat hyvät, päädyttin ratkaisuun jossa virtakytkimen tilaa haistellaan ohjelmallisesti ja kytkimen signaalin muuttuminen käynnistää prosessin, jossa tiedostojärjestelmä suljetaan hallitusti ja

periferaalilaitteiden (GPS ja muistikortti) virran syöttö katkaistaan. Tämän jälkeen mikrokontrolleri asetetaan virransäästötilaan, jolloin virrankulutus tippuu noin 15µA:n[14]. Vaikka valitun step-up –regulaattorin hyötysuhde ei alhaisilla virroilla ole parhaimmillaan (~80%)[12], saavutetaan tälläkin ratkaisulla mielestäni riittävä energiatehokkuus.

Koska kyseessä on paristokäyttöinen laite, piti myös mahdollinen paristoiden loppuminen ottaa huomioon. Päädyin ratkaisuun, jossa paristoparin jännitettä tarkkaillaan säännöllisin väliajoin AD-muuntimella, ja kun jännite putoaa liian matalalle, käynnistetään järjestelmän hallittu alasajo. Koska piiriä itseään ei ole mahdollista sammuttaa täysin, menee järjestelmä virransäästötilaan myös paristojen loppuessa. Tämä kuitenkaan ei ole ongelma, sillä järjestelmän sammutuspiste voitaisiin asettaa kohtuullisen korkealla (n. 1,5 V), sillä regulaattorin virranantokyky järjestelmän vaatimalle 3,3 V:lle heikkenee huomattavasti tämän pisteen jälkeen. Mikrokontrollerin lepotilassa vaatimaa 15µA:a, regulaattori jaksaa syöttää aina 0,6 – 0,7 V jännitteeseen saakka.

4.2 Reaaliaikavaatimukset ja ohjelmiston toiminta

Varsinaisia reaaliaikavaatimuksia järjestelmällä ei ole. Ainoa aikakriittinen tehtävä on muistiblokin kirjoitus muistikortille ja SPI-väylän nopeuden ansiosta kirjoitus kestää pisimmilläänkin 1 Mhz:n kellotaajuudella joitakin millisekunteja. Myös UART-puskurin täyttymiseen on kyettävä vastamaan, mutta NMEA-standardin käyttämän 4 800 baudin siirtonopeus on niin alhainen, ettei tämäkään ole ongelma. Muut tehtävät toimivat periaatteella tehdään, kun ehditään.

Ohjelman kulku on järjestetty siten, että UART-puskurin täyttyminen laukaisee keskeytyksen. Keskeytyksessä UART-puskuriin kirjoitettu merkki tallennetaan NMEA-käskypuskuriin. Jos käsky tulee valmiiksi (UART-puskurin kaksi viimeistä merkkiä ovat rivinvaihtomerkit <CR><LF>), siirretään NMEA-käsky tarkastettavaksi. Jos käskyn tarkistussumma on sama kuin käskyssä, on käsky vastaanotettu oikein ja se kirjoitetaan SD-korttipuskuriin. Tämän jälkeen data kirjoitetaan SD-kortille alustuksessa avatun tiedoston loppuun.

Järjestelmän muut toiminnot pyörivät omassa while-silmukassa, johon järjestelmä automaattisesti palautuu keskeytyskoodin suorituksen jälkeen. Silmukka tutkii ensin virtakytkimen tilan ja vertaa sitä aiemmin tallennettuun tilaan. Näin saataisiin poistettua kytkimen mahdollinen heilunta. Jos kytkimen nykyinen tila sekä edellinen tila ovat samat käynnistetään järjestelmän hallittu alasajo kytkemällä pois keskeytykset. Tämän jälkeen tiedosto ja tiedostojärjestelmä suljetaan ja jännitteen syöttö GPS-moduulille sekä muistikortille katkaistaan. Tämän jälkeen virtakytkimen sisääntulon keskeytys asetetaan aktiiviseksi ja piiri ajetaan virransäästötilaan. Virransäästötilassa järjestelmä jää haistelemaan ulkoisia keskeytyspinnejä, joten virtakytkimen tilan vaihtuminen virransäästötilassa aiheuttaa keskeytyksen, joka herättää mikrokontrollerin virransäästötilasta.

Virtakytkimen tilan tutkimisen jälkeen suoritetaan AD-muunnos paristoparin jännitteestä ja jos paristoparin jännite alittaa ennalta määrätyn alasajojännitteen, suoritetaan sama edellä kuvattu järjestelmän hallittu alasajo.

Kun virtakytkimen ja käyttöjännitteen tutkiminen on suoritettu mikrokontrolleri asetetaan idle-tilaan. Tila ajaa kontrollerin sisäisen kellon alas, mutta ei sulje keskeytysjärjestelmää, joten GPS-moduulin aiheuttama keskeytys toimii herättää järjestelmän idle-tilasta.

Ohjelman toimintaa on kuvattu liitteinä olevissa vuokaavioissa (LIITE 1).

Toteutus ja testaus

Vaikka hyvin suunniteltu on sanonnan mukaan puoliksi tehty, ei vanha kansanviisaus tässä tapauksessa pitänyt ihan paikkaansa. Toteutuksen edetessä kohdattiin lukuisia ongelmia, joista jotkin kulminoituvat lopulta niin pahoihin rautaongelmiin, ettei laitetta saatu täysin toimimaan. Pahimmaksi ongelmaksi muotoutui lopulta SD-kortin kirjasto-osion toimimattomuus, joka tosin johtunee ensisijaisesti tekijän taitamattomuudesta elektroniikan saralla.

Lopulta järjestelmästä toteutettiin POC-versio (Proof of Concept), jolla GPS-moduulin syöttämä data tallennettiin muistikortille. POC:iin toteutuksessa käytettiin itsesuunniteltua piirilevyä, joka liitettiin moduuliksi Microsalon valmistamaan PV-M32 ohjelmointialustaan. Järjestelmä toimi odotetulla tavalla, eikä ole mitään syytä olettaa, etteikö järjestelmä toimisi myös virheettlmän laitteiston kanssa.

5.1 Piirilevy

Piirilevyn suunnittelu on täysin itse suunnittelemani. Koska opinnäytetyötä aloittaessani tietoni piirilevysuunnittelusta olivat täysin olemattomat, jouduin opiskelemaan piirilevysuunnittelun perusteet alusta alkaen. Eräällä tapaa piirinlevyn suunnittelu oli yksi projektin vaativimmasta tehtävistä, koska koko teoriapohja oli itselleni täysin tuntematon. Piirilevy kävi läpi useita kehityssyklejä, kunnes komponentit löysivät pikkuhiljaa omat paikkansa. Piirilevy valmistettiin lopulta ElectroForge-nimisessä yrityksessä.

Suurin virhe piirinlevyn suunnittelussa sattui ajatuskatkoksen vuoksi, sillä mikrokontrolleri päätyi piirilevylle vahingossa peilikuvanaan. Ongelma aiheutui hahmotusongelmastani; en oivaltanut että suunnittelu-ohjelma piirtää pintaliitoskomponentit eri puolelle piirilevyä läpiporattavien komponenttien kanssa. Normaaleille (vastukset, kondensaattorit jne.) komponenteille tästä ei luonnollisestikaan koitunut haittaa, mutta 32-jalkaisen piirin jalat olivat piirilevyllä täysin väärässä paikassa. Ongelmasta selvittiin kääntämällä mikrokontrollerin jalat

pinseteillä 180 °. Tämän jälkeen mikrokontrollerin pohja osoitti ylöspäin. Taivutus onnistui ongelmitta, eikä yksikään jalka vioittunut prosessissa.

Muita tekemiäni virheitä olivat vastuksien ja diodien reikien etäisyyden väärinarviointi. Koska halusin virtalähteen vievän mahdollisimman vähän tilaa piirilevyltä, arvioin vastuksien pituudet liian lyhyiksi, jolloin levyä kalustaessani huomasin joutuvani jättämään komponentit koholle, koska komponentit olivat millin pari pidempiä kuin reikien etäisyys.

Todellista tarkkaavaisuutta vaati regulaattorin juottaminen. MSOP8-koteloinnin jalkojen väli oli sen verran pieni, että kokemus ja kunnolliset välineet ovat välttämättömiä juotostyön onnistumiselle. Juottaminen oli hyvin haastavaa, mutta onnistui lopulta erinomaisesti.

5.2 Virtalähde

Virtalähde oli projektiin ryhtyessäni osa, jonka toimintaan olin perehtynyt kaikkein vähiten. Koko virtalähde perustui datalehtien referenssisuunnitelmiin, mutta prototyypin vaatimaan 3,3 V regulointiin ei löytynyt valmista referenssikytkentää. Jouduin tekemään prototyypin kytkennän yhdistelemällä useampaa referenssikytkentää.

Valmiiksi juotettuna virtalähde kuitenkin toimi esimerkillisesti. Tähtäimeksi asetettu jännitteen korotus 3,3 V:iin toteutui niinkin hyvin, että regulaattori antoi ulospäin 3,32 V:n jännitettä. Virtalähteen virransyöttökyky riitti ohjaamaan mikrokontrolleria ja GPS-moduulia alle 1.5V:n antojännitteellä, jolloin kaikki asettamani vaatimukset järjestelmälle täyttyivät.

Lievä pettymys järjestelmän suhteen oli normaalin AAA-alkaliparistoon kestävyys. Sisätiloissa testattuna peruslaatuinen alkaliparisto kesti ainoastaan joitakin tunteja, jolloin laitteen toimivuus pakkasella olisi huomattavasti lyhyempi. Lisäksi sarjaan kytketyt paristot kuluivat hyvin epätasaisesti. Yön yli kestäneessä rasitustestissä paristoiden jännite-ero oli lähes 1 V. Toinen oli noin 0,3 V, toisen ollessa noin 1 V. Toisaalta nämä ongelmat liittyvät enemmän paristojen ominaisuuksiin, kuin varsinaiseen virtalähteeseen.

Suurin ongelma virtalähteessä oli kuitenkin sen koiria karkottava vaikutus. Ihmeekseni huomasin, että normaalisti jaloissa pyörivä koirani ei tullut lähimainkaan työhuonettani projektin edistyessä. Syyn selvittäminen kesti itseltäni useamman viikon, kunnes eräänä päivänä huomasin, että koira poistui huoneesta aina kun laitteeseen kytkettiin virta. Koteloin prototyypin pakasterasiaan ja kiinnitin sen koiran selkään lyhyen lenkin ajaksi. Koiran ruumiinkieli viittasi siihen, että koiraa häiritsi jokin ulkoinen asia. Koiran käytös palasi normaaliksi vasta kun kytkin laitteen pois päältä.

Ilman erillisiä mittalaitteita en pysty vahvistamaan seuraavaa teoriaa, mutta epäilen koiran käytöksen olevan syynä virtalähteen synnyttämän korkeataajuisen äänen. Koska regulaattori toimii jännitettä katkomalla, se todennäköisesti aiheuttaa korkeataajuuksista äänisignaalia ympäristöönsä. Koiran kuuloaisti ylittää oman kuuloaistimme selvästi ja ulottuu joidenkin arvioiden mukaan jopa 100 kHz:iin [15]. Todennäköisesti virtalähde tuottaa ääntä jollain tällaisella taajuudella ja näin ollen koira aistii virtalähteen synnyttämän äänen. Koska koiran käytös on selkeästi kytköksissä laitteen päällä oloon, en keksi äänen lisäksi mitään muuta syytä koiran käytökseen. Koira ei vierasta muita sähkölaitteita ja GPS-moduuli ainoastaan vastaanottaa signaaleja. Koira ei myöskään reagoi mihinkään muuhun GPS-laitteeseen samalla tavoin. Ainoaksi vaihtoehdoksi jää siis ääni. Toki koiran voisi totuttaa ääneen, mutta koska pelastuskoiran pitää kyetä käyttämään kaikkia aistejaan, ei kyseinen virtalähde sovellu kovin hyvin tähän käyttötarkoitukseen.

5.4 Mikrokontrollerin ohjelmointi

Ohjelmistioa suunnitellessani olin unohtanut kokonaan hyvin kriittisen ominaisuuden prototyypistä. Kun piirilevy oli koottu ja juotettu, huomasin etten ollut lainkaan ottanut huomioon miten saisin ohjelmoitua mikrokontrollerin. Alkuperäinen ajatus oli ollut ohjelmoida mikrokontrolleri rinnakkaisportilla. Huomasin kuitenkin, etten omistanyt ainuttakaan tietokonetta, jossa olisi ollut rinnakkaisportti.

Onneksi satuin muistamaan keskusteluni opettaja Pentti Vahteran kanssa käymäni keskustelun PV-M32 ohjelmointialustan käyttämisestä kontrollerin ohjelmointiin.

Voisin kytkeä käytössäni olevan PV-M32 -ohjelmointialustan mikrokontrolleriin ja ohjelmoida kontrollerin tällä menetelmällä. Tein tarvittavat kytkennät koekytkentälevylle ja laitteen mikrokontrolleri tunnistui ongelmitta ohjelmointiohjelmistona käyttämässäni AVR Studiossa.

Mikrokontrollerin ohjelmoimisessa ongelmaksi muodostui kuitenkin sen irrotus kannasta. Kantaa ei ole suunniteltu siihen, että piiri irrotetaan siitä jatkuvasti uudelleenohjelmointia varten, jolloin kontaktit alkoivat hiljalleen löystyä. Ensimmäisenä kontakteista lopetti toimintansa virheiden etsintään käytettävän ledin kontakti. Pian tämän jälkeen myös yksi SPI-linjan kontakteista lopetti toimintansa. Tässä vaiheessa aloin olla jo melko vakuuttunut, että ongelma oli ohjelmiston sijaan laitteistossa, joten en alkanut edes pohtimaan miten saisin 32-pinnisen kannan irrotettua piirilevystä rikkomatta kuitenkaan piirilevyn johtimia. Koska piirilevy oli kaksipuoleinen ei uuden piirilevyn valmistaminen omilla välineilläni myöskään onnistunut.

Jälkiviisaana on helppo todeta, että oikea ratkaisu olisi ollut ohjelmointiliittimen kolvaaminen suoraan kannan nastoihin ja liittää eri jännitteen omaavat järjestelmät toisiinsa yksinkertaisen jännitejakokytkennällä. Näin piirin turhalta irrottamiselta oltaisiin vältytty ja järjestelmän virheiden etsintä olisi ollut moninkerroin nopeampaa kuin ohjelmiston virheiden etsintä yhden ainoan ledin ja käyttämättömien pinnien avustuksella. Kytkennän etu olisi ollut myös siinä, että se olisi mahdollistanut mikrokontrollerista löytyvän JTAG-yhteyden käyton.

5.5 Mikrokontrolleri ja SD-muistikortti

Mikrokontrollerin ja SD-muistikortin yhdistäminen aiheutti työssä kaikkein suurimmat ongelmat. Kommunikaatio SD-kortin ja mikrokontrollerin välillä toimi näennäisesti moitteetta. Kortti kätteli kontrollerin kanssa, tiedostojärjestelmä saatiin auki, eikä edes juurihakemiston tiedostolistaus tuottanut ongelmaa. Sen sijaan tiedoston luonti tai tiedostoon kirjoittaminen aiheuttivat koko järjestelmän kaatumisen. Etsin ohjelmiston virheitä monta tuntia, mutta ongelmalliselle käytökselle ei näyttänyt löytyvän minkäänlaista selitystä. Järjestelmä lopetti toimintansa satunnaisesti aina kortille kirjoittamisen yhteydessä. Kun sitten vielä

mikrokontrollerin kannan kontaktit lakkasivat toimimasta aloin olla vakuuttunut, että vika tosiaan oli laitteistossa.

Asiaa nyt pidemmän aikaa pyöritelleenä, uskon ongelman johtuvan päätöksestäni käyttää mikrokontrollerin omaa sisäistä oskillaattoria kellopulssin tuottamiseen. Tämä oskillaattori ei ole kovin tunnettu vakaudestaan, joten loogisin syy olisi ettei oskillaattori pysty tuottamaan tarpeeksi vakaata kelloa SD-kortin käyttöön. Kaikki toimivat operaatiot, aiheuttavat hyvin vähän kommunikaatiota kortin suuntaan. Sen sijaan kirjoitusoperaatiossa ja tiedoston luonnissa pääpaino on datassa, joka lähetetään mikrokontrollerilta kortille päin. Sinällään tällä ei pitäisi olla vaikutusta, sillä datansiirto tahdistetaan erillisellä kellolinjalla. Toisaalta SD-standardi määrittää kyllä kellopulssin laskulle ja nousulle maksimikeston, joten jos kellolinja ei tuota kunnollista kanttiaaltoa, tahdistus ei toimi. Toinen seikka, joka tukee tätä teoriaa on fakta, ettei järjestelmä suostunut edes kättelemään muistikortin kanssa ennen kuin sisäinen kello nostettiin 4 MHz:iin. Kättely kuitenkin tapahtuu maksimissaan 400 kHz:n kellolla, joten ilmeisesti SPI-protokollasta huolehtiman raudan kellotaajuuden jakaja korjaa kelloa niin, että kellopulssi kelpaa myös SD-kortille.

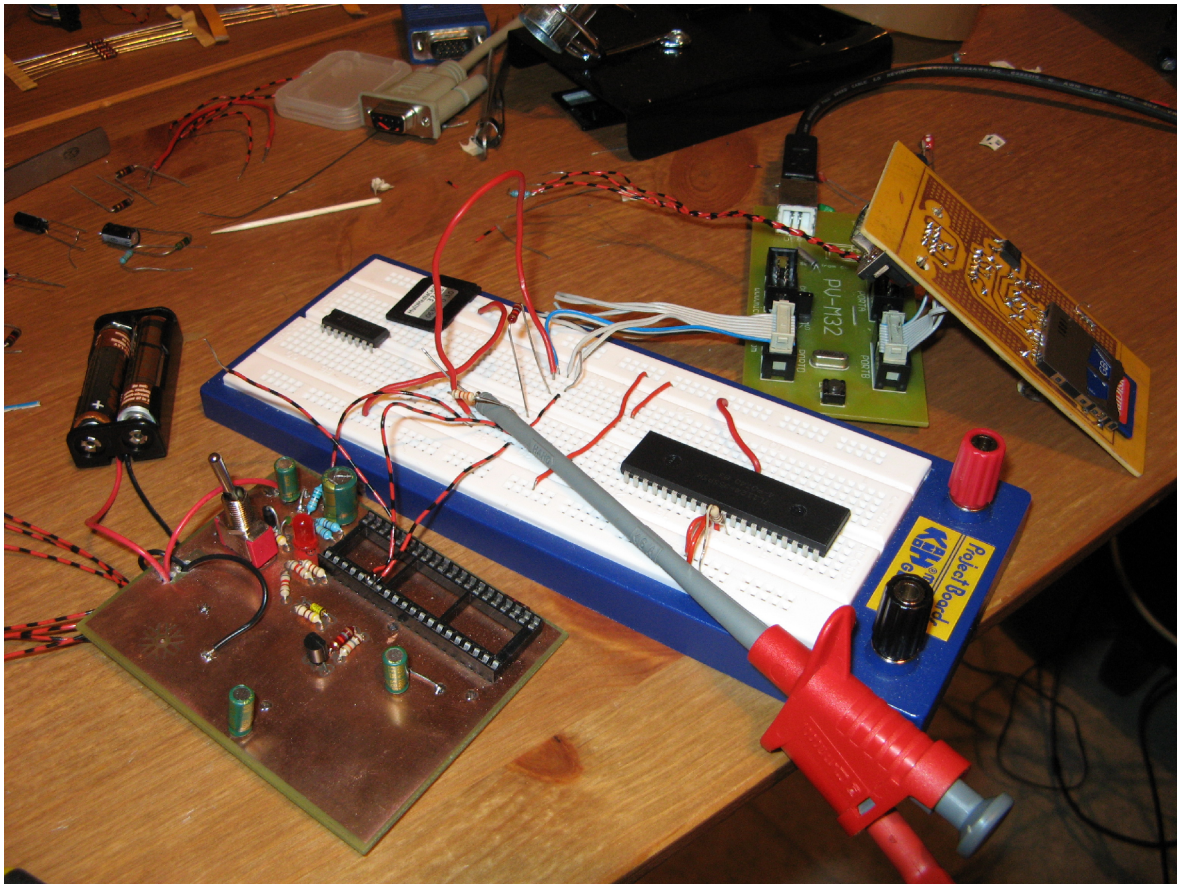
Vaihtoehtoinen teoria järjestelmän toimimattomuudelle, on regulaattorilta saatavan jännitteen heiluminen. Valitettavasti käytössäni ei työtä tehdessäni ollut oskilloskooppia, jolla ongelman olisi saanut helposti varmennettua. Toisaalta itse epäilen tätä teoriaa, sillä GPS-moduuli ja mikrokontrolleri toimivat moitteetta, eikä SD-kortin pitäisi olla kovin tarkka saamastaan jännitteestä.

5.6 POC-järjestelmän teko

Koska halusin kuitenkin saada lopullisen varmistuksen sille, oliko vika raudassa vai ohjelmistossa, ryhdyin rakentamaan yksinkertaista järjestelmää käytössäni olevan PV-M32 -ohjelmointialustan päälle. Kyseisessä alustassa on ulkoinen 4 Mhz:in kellokide, joten kellopulssin vakaus on täysin eri luokkaa kontrollerin oman sisäisen kellon kanssa.

Käyttämäni FAT-kirjaston sivuilta löytyi myös esimerkkikytkentä, jonka kopioin sellaisenaan käyttööni. Suunnittelin kytkennän pohjalta piirilevyn, jonka sitten valmistin itse lämmönsiirtomenetelmällä ja syövyttämällä piirilevyaihiosta.

Tekemäni SD-moduuli toimi ensiyrittämällä. Tiedostoon kirjoittaminen ei aiheuttanut ongelmia. Kun kortille kirjoitus oli kunnossa kytkin järjestelmään vielä alkuperäisellä piirilevyllä olleen GPS-moduulin vetämällä kytkentälangan piirilevyn kannasta ohjelmointialustan UART-sisääntuloon (ks. kuva 3). Myös tämä kytkentä toimi ongelmitta. Lopuksi juotin GPS-moduuliin antennin, jotta sain varmistuksen, että myös aktiiviantenni toimisi kytkennässäni. Tämäkin toimi ongelmitta.



Kuva 3: POC-moduulin lopullinen kytkentä. POC-moduuli oikealla(kellertävä piirilevy), PV-M32 keskellä (vihertävä piirilevy) ja prototyyppi äärimmäisenä vasemmalla. Antennia ei kuvaa ottaessa oltu vielä juotettu kiinni prototyyppiin. Koekytkentälevyä käytettiin ainoastaan signaalien yhdistämiseen. PV-M32:n UART-TX -linjan jännite on pudotettu kytkennässä jännitejakokytkennällä 5 voltista 3.3 volttiin. Kytkentälevyn oikealla ja vasemmalla reunalla olevat piirit eivät liity kytkentään.

5.5 Muut komponentit

Valitettavasti järjestelmän muiden ominaisuuksien testaaminen jäi vähemmälle raudan ongelmallisuuden vuoksi. Virransyöttöön käytettyä transistoria en tosin päässyt testaamaan kuin teorian tasolla. Jo piirilevyn kasausvaiheessa onnistuin juottamaan transistorin anturan irti piirilevystä käyttäessäni aivan liian kuumaa kolvia. Foliovetoa korjatessani kuumensin myös transistorin rikki. Koska virransyötön hallinta ei ollut päällimmäisenä mielessäni, ohitin koko transistorin hyppylangalla, jotta GPS-moduuli sekä SD-kortti saivat virtaa.

Yhteenveto

Moni asia meni tässä projektissa ei mennyt suunnitelmieni mukaan. Suurin osa aiheutui huonosta suunnittelusta ja tekijän kokemattomuudesta. Jälkikäteen mietittynä oikea suunta olisi ollut tehdä järjestelmän virheen etsinnästä huomattavasti helpompaa ja tutustua paremmin komponenttien asettamiin vaatimuksiin, kuitenkin on sanottava, että projektia suunnitellessani tietotaitoni tarvittavien asioiden hahmottaminen ei ollut riittävä.

Täysin odottamaton ongelma oli myös koiran reaktio laitteeseen. En usko, että tätä ongelmaa olisi osannut aavistaa edes kokenut elektroniikkasuunnittelija.

Järjestelmään kuuluvan mikrokontrollerin ohjelmointi oli alusta saakka suuri ongelmakohta. Mikrokontrollerin jatkuva irrottaminen ohjelmointiin käytetystä koekytkentalevystä tai piirikannasta oli todella aikaa vievää ja turhauttavaa. Usein koodiin tuli tehneeksi aivan liikaa muutoksia, jolloin virheiden etsintä muuttui mahdottomaksi, koska muuttuneita asioita oli yksinkertaisesti liikaa hahmotettavaksi. Lopulta tehokkain virheenetsintä tapa oli metodi, jossa pyrin vain löytämään missä kohtaa koodia järjestelmä kaatui. Kun järjestelmän kaatumiset osoittautuivat vielä täysin satunnaisiksi, oli vian paikantaminen ilman kunnollisia virheenetsintätyökaluja ja oskilloskooppia mahdoton tehtävä.

Jälkiviisaana on helppo sanoa, että laitteiston osuus ongelmiin olisi pitänyt olla itsestään selvä jo ensimmäisistä kokeiluista lähtien. Kirjasto on useiden henkilöiden testaama, joten ainoa järkevä syy ongelmiin olisi laitteistossa. Toisaalta ymmärrykseni elektroniikasta on kuitenkin sen verran hataralla pohjalla, etten voinut käsittää vian voivan olevan laittestopohjainen. Toimihan muistikortin kättely ongelmitta ja kaikki ratkaisut, jotka löytämissäni ohjeissa kuvattiin liittyivät nimenomaan muistikortin kättelyssä tapahtuviin ongelmiin. Olin vakuuttunut ongelmien olevan softassa, kunnes PV-M32 –moduulini toimi ensimmäisellä yrityksellä.

Järkevämpi lähtökohta projektille olisi ollut myös moduulin rakentaminen ennen varsinaisen prototyypin valmistamista. Koodin olisi voinut viilata kuntoon ohjelmointialustassa, jolloin virheiden etsintä olisi ollut paljon helpompaa. Jos valmiiksi tehty koodi ei olisi toiminut prototyypissä, olisi vika ollut helppoa kohdentaa juuri toimimattomaan laitteistoon. Tekemällä prototyypin ensin tulin ahnehtineeksi yhtälöön liian monta tuntematonta ja tämä kostautui lopulta toimimattomalla prototyypillä.

Kun tarkastelen projektia tässä vaiheessa, en kuitenkaan voi sanoa sen menneen kovinkaan pieleen. Vaikka prototyyppi ei lopulta toiminutkaan odotusten mukaisesti, koen oppineeni todella paljon ennen kaikkea elektroniikasta ja tämäntyyppisten projektien suunnittelusta juuri tekemieni virheiden kautta. Tätä kirjoittaessani pohdiskelen jo miten korjaan tulevaa GPS-loggeriiprototyyppiä, jotta se varmasti toimii haluamallani tavalla. Lisäksi mielessä pyörii useita muita mikrokontrolleriprojekteja, joihin en olisi uskaltanut ennen tätä työtä tarttua niiden näennäisesti vaikealta tuntuvan elektroniikan takia.

Lähteet

[1] Universal asynchronous receiver/transmitter [WWW-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/UART (Luettu: 29.10.2009)

[2] Serial Peripheral Interface Bus [WWW-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus (Luettu: 29.10.2009)

[3] Example File Systems, Andrew S. Tanenbaum [WWW-dokumentti]. Saatavilla: http://www.informit.com/articles/article.aspx?p=25878&seqNum=4 (Luettu: 29.10.2009)

[4] File Allocation Table [WWW-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/File_Allocation_Table (Luettu: 31.10.2009)

[5] FAT16 File System, Maverick Operating System Documentation [WWW-dokumentti]. Saatavilla: http://www.maverick-os.dk/FileSystemFormats/FAT16_FileSystem.html (Luettu 9.11.2009)

[6] Understanding FAT32 Filesystems, Paul Stoffregen [WWW-dokumentti]. Saatavilla: http://www.pjrc.com/tech/8051/ide/fat32.html (Luettu: 31.10.2009)

[7] Prasad, Ramjee; Ruggieri, Marina, *Applied Satellite Navigation Using GPS, GALILEO, and Augmentation Systems,* Artech House, 2005, 308 s.

[8] SirfStar III, [WWW-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/SiRFstar_III (Luettu: 9.11.2009)

[9] Tranter, William H., *Wireless Personal Communications: Channel Modeling and Systems Engineering,* Kluwer Academic Publishers , 1999, 266 s.

[10] Universal Serial Bus Specification Revision 2, [WWW-dokumentti]. Saatavilla: http://www.usb.org/developers/docs/usb_20_052709.zip (Luettu: 9.11.2009)

[11] MultiMediaCard, [WWW-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/MultiMediaCard, (Luettu 9.11.2009)

[12] National Semiconductor, LM2621 Low Input Voltage, Step-Up DC-DC Converer [Datalehti]. Saatavilla http://www.national.com/pf/LM/LM2621.html#Overview, (Luettu 19.11.2009)

[13] H. Honkanen, Hakkurireguloinnin periaatteet. [WWW-dokumentti]. Saatavilla http://gallia.kajak.fi/opmateriaalit/yleinen/HonHar/ma/REG_HAKKURIREGULOINNIN%20PERIAATTEET.pdf, (Luettu 19.11.2009)

[14] Atmel corporation, AtMega16L datasheet [pdf]. Saatavilla http://www.atmel.com, (Luettu 21.12.2009)

[15] The Physics Factbook, Frequency Range of Dog Hearing, toimittanut Glenn Elert [WWW-dokumentti]. Saatavilla http://hypertextbook.com/facts/2003/TimCondon.shtml, (Luettu 28.4.2010)

[16] SD Group, SD Specifications Part 1 Physical Layer Simplified Specification [PDF] saatavilla http://www.sdcard.org/developers/tech/sdcard/pls/, (Luettu 14.1. 2010)

[17] How to Use MMC/SDC [WWW-dokumentti]. Saatavilla: http://elm-chan.org/docs/mmc/mmc_e.html, (Luettu 14.1.2010)

[18] NMEA 0183 [WWW-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/NMEA_0183, (Luettu 14.1.2010)

[19] Dale DePriest, NMEA data [WWW-dokumentti]. Saatavilla : http://www.gpsinformation.org/dale/nmea.htm, (Luettu 14.1.2010)

[20] Atmel, Atmel Products [WWW-dokumentti]. Saatavilla http://www.atmel.com/dyn/products/devices_v2.asp?family_id=607#760 (Luettu 28.5.2010)

Liitteet

LIITE 1 Ohjelman suunnitteluvaiheessa tehdyt vuokaaviot

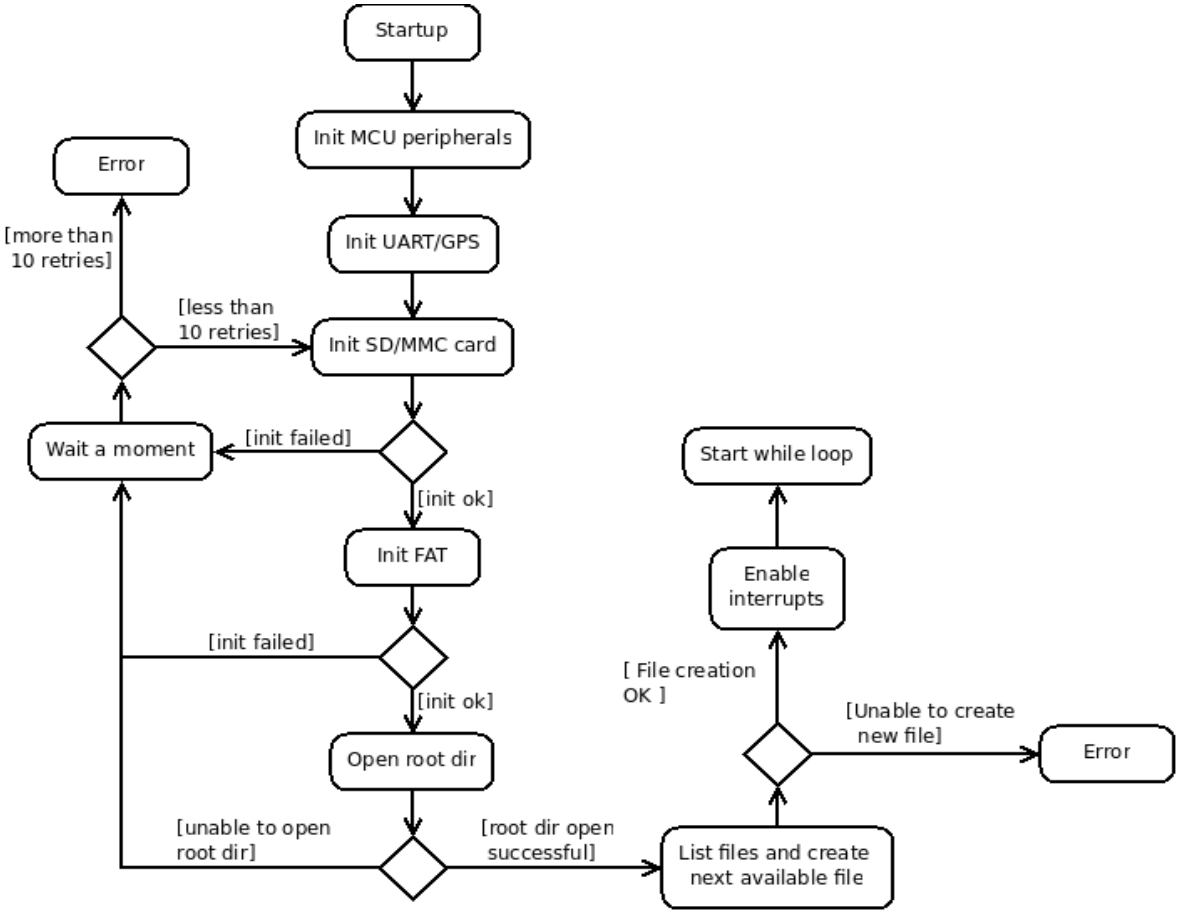
LIITE 2 Valittujen komponenttien sähköisiä ominaisuuksia

LIITE 3 Piirilevyt sekä kytkentäkaaviot

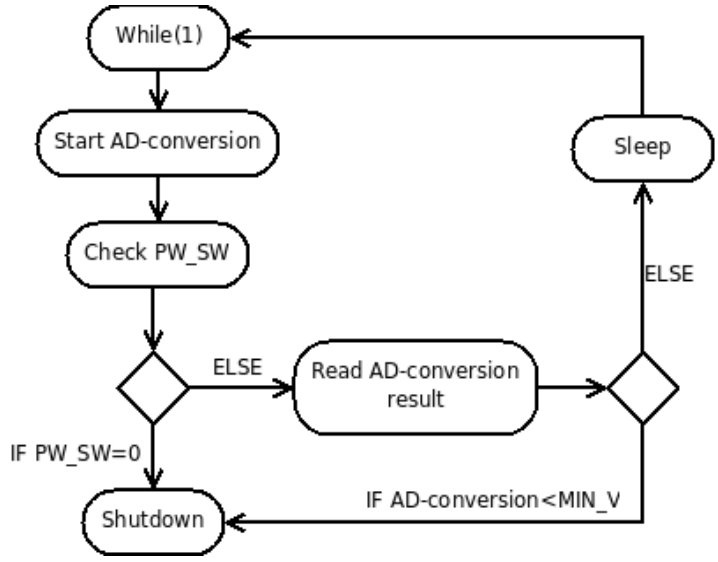
LIITE 1 / 1 (2)

OHJELMAN SUUNNITTELUVAIHEESSA TEHDYT VUOKAAVIOT

Alustus:

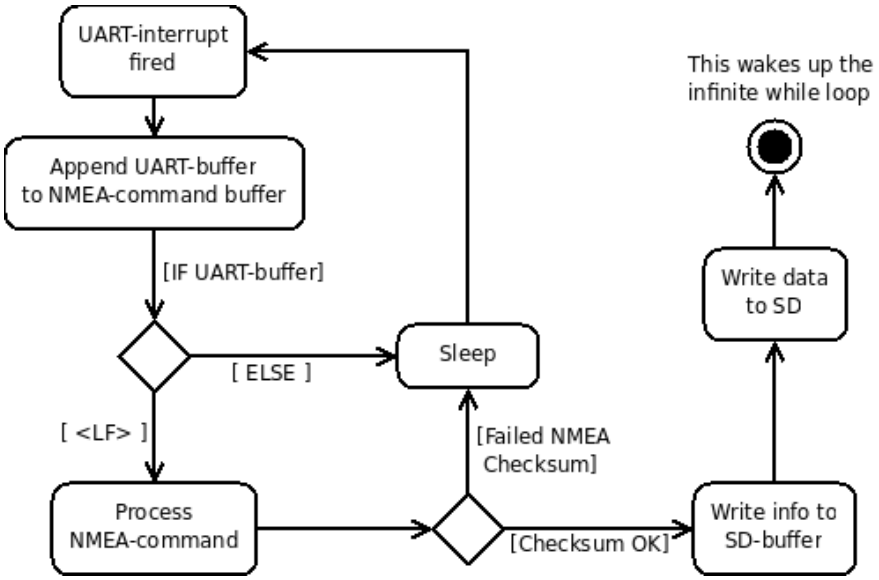


While-silmukka:



LIITE 1 / 2 (2)

Keskeytys:



National Semiconductor

March 2005

LM2621
Low Input Voltage, Step-Up DC-DC Converter

General Description

The LM2621 is a high efficiency, step-up DC-DC switching regulator for battery-powered and low input voltage systems. It accepts an input voltage between 1.2V and 14V and converts it into a regulated output voltage. The output voltage can be adjusted between 1.24V and 14V. It has an internal 0.17Ω N-Channel MOSFET power switch. Efficiencies up to 90% are achievable using the LM2621.

The high switching frequency (adjustable up to 2MHz) of the LM2621 allows for tiny surface mount inductors and capacitors. Because of the unique constant-duty-cycle gated oscillator topology very high efficiencies are realized over a wide load range. The supply current is reduced to 80µA because of the BiCMOS process technology. In the shutdown mode, the supply current is less than 2.5µA.

The LM2621 is available in a Mini-SO-8 package. This package uses half the board area of a standard 8-pin SO and has a height of just 1.09 mm.

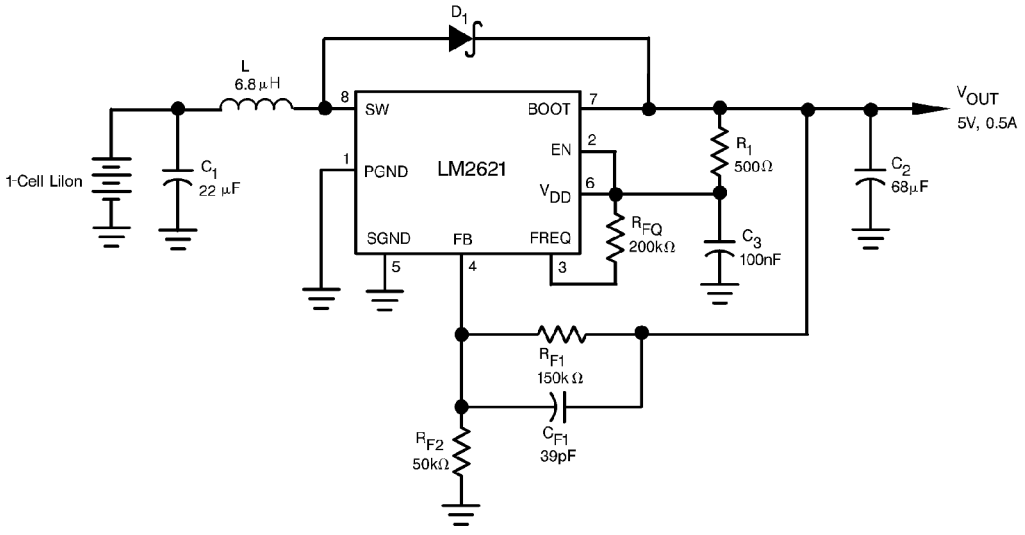
Features

- Small Mini-SO8 Package (Half the Footprint of Standard 8-Pin SO Package)
- 1.09 mm Package Height
- Up to 2 MHz Switching Frequency
- 1.2V to 14V Input Voltage
- 1.24V - 14V Adjustable Output Voltage
- Up to 1A Load Current
- 0.17 Ω Internal MOSFET
- Up to 90% Regulator Efficiency
- 80 µA Typical Operating Current
- <2.5µA Guaranteed Supply Current In Shutdown

Applications

- PDAs, Cellular Phones
- 2-Cell and 3-Cell Battery-Operated Equipment
- PCMCIA Cards, Memory Cards
- Flash Memory Programming
- TFT/LCD Applications
- 3.3V to 5.0V Conversion
- GPS Devices
- Two-Way Pagers
- Palmtop Computers
- Hand-Held Instruments

Typical Application Circuit



10093412

1. Product Information

■ Product Part I.D. **ET-312**

■ Product Description

The ET-312 GPS engine board is low cost but maintains high reliability and accuracy making it an ideal choice for integration with OEM/ODM systems.

■ Product Features

- ✓ SiRF GSC2x high performance GPS Chip Set
- ✓ Very high sensitivity (Tracking Sensitivity: -159dBm)
- ✓ Extremely fast TTFF (Time To First Fix) at low signal levels
- ✓ Single-sided component installation, easy to mount on another PCB board
- ✓ Supports the NMEA 0183 and SiRF Binary protocols
- ✓ WAAS/ EGNOS support
- ✓ Foliage Lock for weak signal tracking
- ✓ Compact in size
- ✓ All-in-view 20-channel parallel processing
- ✓ Enhanced algorithm for navigation stability
- ✓ Superior urban canyon performance
- ✓ RoHS compliant

■ Product Specifications

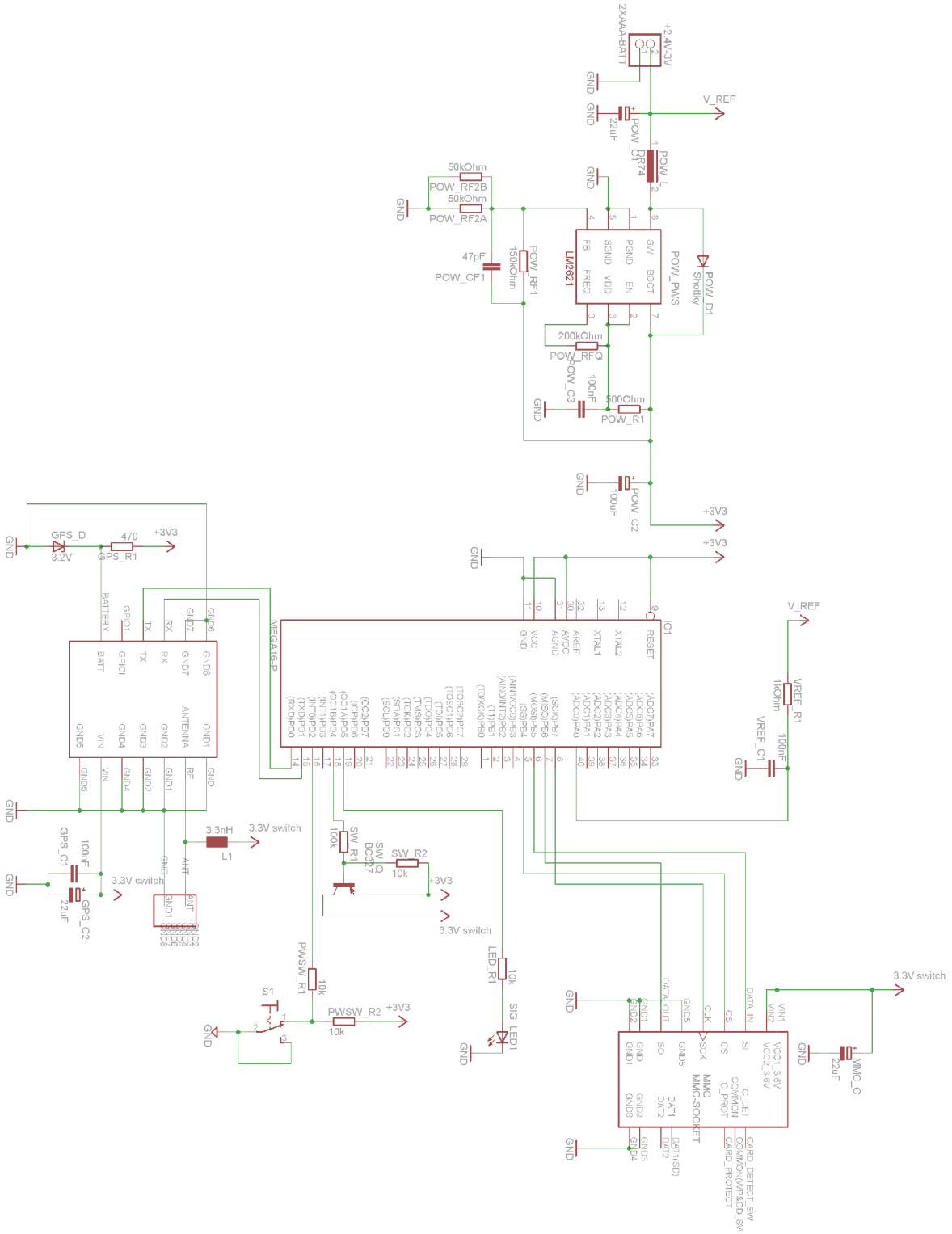
GPS Receiver	
Chipset	SiRF Star III/FLP Single
Frequency	L1, 1575.42 MHz
Code	1.023 MHz chip rate
Protocol	Electrical Level: TTL level, Output Voltage Level: 0V~2.85V Baud Rate: 4,800 – 57,000 bps Output Message: NMEA 0183 GGA, GSA, GSV, RMC (VTG, GLL optional) Active antenna
Channels	20
Sensitivity	-159dBm

Cold Start	42 seconds average
Warm Start	38 seconds average
Hot Start	8 second average
Reacquisition	0.1 second average
Accuracy	Position: 10 meters, 2D RMS 5 meters, 2D RMS, WAAS enabled Velocity: 0.1 ms Time: 1μs synchronized to GPS time
Maximum Altitude	18,000 meters (60,000 feet) max
Maximum Velocity	515 meter/second (1000 knots) max
Maximum Acceleration	4G
Datum	WGS-84
Jerk Limit	20m/sec **3
Physical Characteristics	
Dimensions	1.1" x 0.8" x 0.11"
	(28mm x 20mm x 2.9mm)
DC Characteristics	
Power Supply	3.3VDC, ±5%
Power Consumption	42mA (Continuous Mode)
	25mA (Trickle Power Mode)
Environmental Range	
Humidity Range	5% to 95% non-condensing
Operation Temperature	-40F to +176F (-40C to 85C)

LIITE 3 / 1 (2)

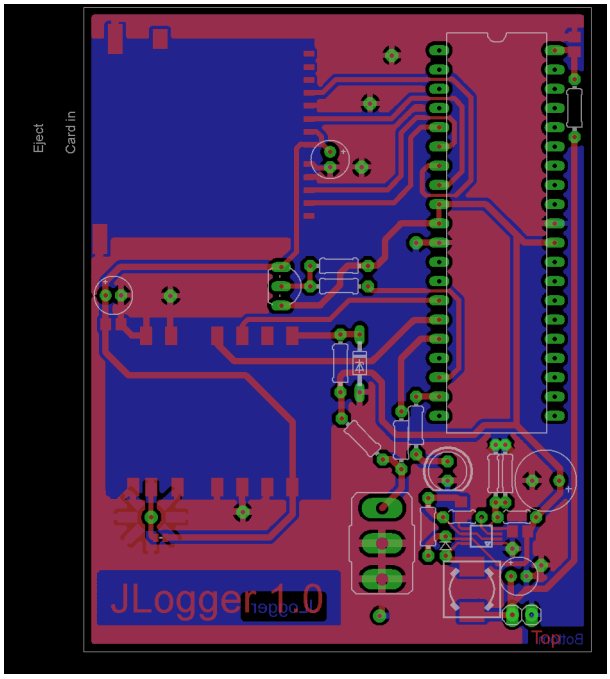
PIIRILEVYT SEKÄ KYTKENTÄKAAVIOT

Kytkentäkaavio

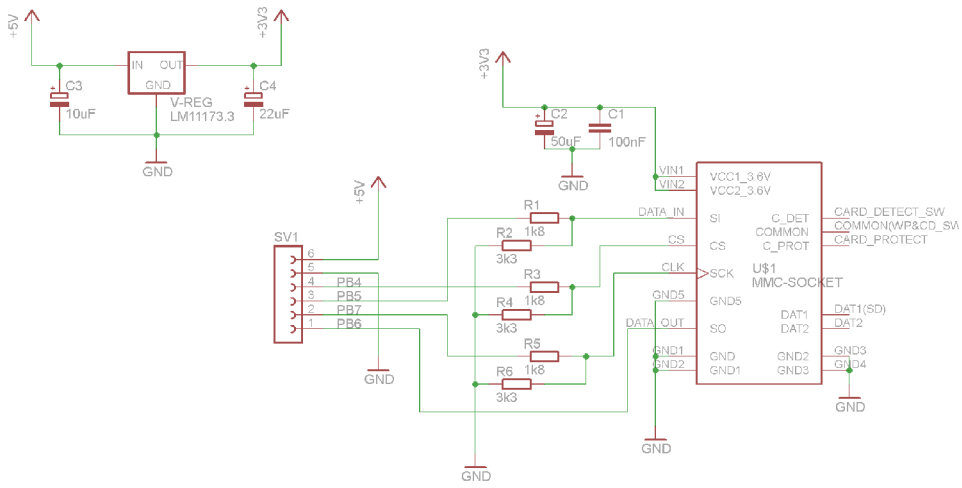


LIITE 3 / 2 (2)

Piirilevy



POC-moduulin kytkentäkaavio



POC-moduulin piirilevy: