Fjernovervåkningssystem for Shell Eco-marathon kjøretøy

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

PROSJEKTOPPGAVE

Kandidatens navn: Anders Guldahl

Fag: Teknisk kybernetikk

Oppgavens tittel (norsk): Fjernovervåkningssystem for Shell Eco-marathon kjøretøy

Oppgavens tittel (eng.): Remote monitoring system for Shell Eco-marathon vehicle

Oppgavens tekst:

Oppgaven omfatter utvikling av en telemetrimodul for trådløs sanntidsovervåkning og logging av relevante data fra Shell Eco-marathon-kjøretøyet. Generelt inkluderer oppgaven følgende punkter:

- Utarbeide krav- og designspesifikasjon for ovennevnte overvåkningsystem
- Detaljspesifikasjon og implementasjon av dedikert maskin- og programvare
- Test, dokumentasjon og diskusjon av resultater

Oppgaven gitt: 17. august 2009

Besvarelsen leveres: 19. desember 2009

Utført ved Institutt for teknisk kybernetikk

Veileder: Jo Arve Alfredsen (ITK, NTNU)

Trondheim, 4. september 2009

Jo Arve Alfredsen
Faglærer

Sammendrag

Shell Eco-marathon er en årlig bilprototype-konkurranse mellom studentlag fra høyskoler og universiteter i Asia, Europa og Amerika. Våren 2010 skal NTNU delta for 3. gang. Denne oppgaven går ut på å spesifisere krav, designe, lage og teste en telemetri-modul for eco-marathon-bilen som skal delta i konkurransen i 2010. Modulen er designet for å løse problemer tidligere deltakerteam fra NTNU har hatt med logging av måledata. Modulen vil gjøre det mulig å endre systemparametere i sanntid. Dette vil være nyttig både under utvikling, test og i konkurransesammenheng med den miljøvennlige bilen.

Ulike trådløse teknologier og automotive kommunikasjonsbusser er vurdert mot hverandre, GPS (Global Positioning System) er tatt i bruk for å knytte posisjonsdata til målinger, og mulighet for lagring av alle måledata til minnekort er implementert. Telemetri-modulen er ment å bli en del av et mer omfattende elektronikksystem som skal utvikles utover våren før konkurransen i mai. Dette innebærer blant annet at valg av kommunikasjonsbuss og fysisk sammenkobling mellom modulene er fastlagt i denne oppgaven.

Forskjellige softwareløsninger for mikrokontrollere og GPRS-modul (General Packet Radio Service) er vurdert. Software-drivere til en Atmel-mikrokontroller er implementert for blant annet CAN-bus (Controller Area Network), FAT-filsystem (File Access Table), minnekort og seriell kommunikasjon med en GPRS-modul. Elektronikk for modulen er utviklet, kretskort er laget og dette er igjen montert i en solid og kompakt boks klar for montering i bilen.

Arbeidet har blant annet innebært detaljert design av en DC-DC-spenningsomformer med beskyttelsesløsning mot overspenninger, implementasjon av seriell kommunikasjon med et minnekort, en GPRS-modul og andre CAN-bus-noder. Telemetri-modulen er grundig testet, både software-drivere og hardware er funnet å fungere som tiltenkt. Problemer som har dukket opp under arbeidet er beskrevet i detalj og i stor grad løst underveis.

Innhold

Figurer

Tabeller

Forkortelser

ADC	Analog to Digital Converter
CAN	Controller Area Network
DC-DC	Direct Current - Direct Current
EEPROM	Electrically Erasable Programmable Read-Only Memory
ESR	Equivalent Series Resistance
FAT	File Access Table
FLIP	Flexible in system programmer
FTP	File Transfer Protocol
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile communications
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
I/O	Input/Output
I2C	Inter-Integrated Circuit
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LIN	Local Interconnect Network
NMEA	National Marine Electronics Association
PDP	Packet Data Protocol

PTC	Positive Temperature Coefficient
RF	Radio Frequency
RJ	Registered Jack
RMC	Recommended Minimum Sentence C
SD-kort	Secure Digital-kort
SMTP	Simple Mail Transfer Protocol
SPI	Serial Peripheral Interface
TCP/IP	Transmission Control Protocol/Internet Protocol
TDMA	Time Division Multiple Access
TTFF	Time To First Fix
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus

1 Innledning

Shell Eco-marathon er en årlig bilprototype-konkurranse mellom studentlag fra høyskoler og universiteter i Asia, Europa og Amerika. Det arrangeres billøp i hver av de tre verdensdelene på våren. Konkurransen går ut på å kjøre en gitt distanse i løpet av en gitt tid med minst mulig drivstofforbruk. Energikilden kan enten være konvensjonelt drivstoff som bensin, diesel eller naturgass, eller det kan være alternative energikilder som hydrogen, solceller eller bio-brensel. Måleenheten for drivstofforbruk er kilometer kjørt per liter 95 oktan bensin, forbruk og kjørestrekning per liter blir regnet om for de forskjellige energikildene.

Konkurransen har to kategorier; en som heter Prototype, der det meste er lov med tanke på design, sikkerhet og krav til kjøretøyet. Den andre kategorien heter Urban Concept og er den klassen NTNU deltar i. I Urban Concept-klassen stilles det strenge krav til bilen, både funksjonelle krav og sikkerhetskrav. Bilen må i hovedsak ha mange av detaljene en vanlig bil må ha, som lys, fot-betjening av gass og brems og plass til litt bagasje. NTNU deltok første gang i 2008, både da og i 2009 har teamet hatt en tverrfaglig sammensetning. I 2010 arrangeres konkurransen i Lausitz, Tyskland.

Bilen

Da bilen ble laget i 2008 var fokuset å spare så mye vekt som mulig. Hele karosseriet er laget i karbonfiber og fremdriftssystemet bestod av en elektromotor med tilhørende hydrogen-brenselcelle. Karosseriet har fulgt bilen fra 2008, men fikk mange aerodynamiske forbedringer i 2009. Fremdriftssystemet ble i stor grad byttet ut i 2009-versjonen med ny brenselcelle og spesiallaget elektromotor. Teamets fokus for bilen i 2010 er å redusere bilens prototype-preg.

Eksempel på prototypedetaljer som trenger forbedring er et IKEA-hengsel som dørhengsel og rattoverføringen laget av tauverk. Et annet forbedringspotensiale er driftssikkerheten, bilen ble i utgangspunktet ikke bygget for å tåle veldig mange testkjøringer, framvisninger på messer og lignende. Under løpet i 2009 var det med et nødskrik at bilen klarte å fullføre et godkjent løp.

Telemetri-modul

Etter samtaler med teammedlemmer fra 2009 kom det fram at en savnet funksjon ved bilens elektriske system var mulighet for trådløs overføring av systemparametere i sanntid. Det ble ganske tungvint å overføre en logg og lese den etter hvert løp eller hver testkjøring. En mulighet for at team-medlemmer kan sitte i pitten og følge med på bilens parametere i sanntid hadde vært en stor fordel under testkjøring og under løpet. Ved feil på bilen under

et løp vil det være mye lettere for resten av teamet å hjelpe sjåføren over mobiltelefon fordi alle systemparametere er tilgjengelig for hele teamet.

Bilen har allerede en håndholdt GPS-modul (Global Positioning System) for sjåføren, som blant annet beregner rundetider, viser bilens fart og strekning kjørt. Men GPS-data er ikke tilgjengelig for resten av teamet, og informasjonen måtte overbringes muntlig over mobiltelefon. En modul som også har mulighet for å overføre GPS-posisjonen er derfor sett på som svært nyttig. Å lage denne telemetri-modulen er denne oppgavens hovedfokus. Telemetri (Tele + metri) betyr fjernmåling og det er akkurat dette modulen skal utføre, "telemetri-modul" er derfor et passende navn.

2 Kravspesifikasjon

For å designe et system som løser de oppgaver det er ment å løse er det viktig å sette opp spesifikasjonskrav på forhånd. Dette vil gjøre designarbeidet enklere da en vet hva en må forholde seg til. Det er også svært vanskelig å bli ferdig med et produkt dersom kravene endres når detaljdesignfasen har begynt. For telemetri-modulen har de viktigste aspektene som trådløs kommunikasjon, kommunikasjon med resten av bilen og muligheten for logging vært sett på under kravspesifikasjonen.

2.1 Trådløs kommunikasjon

Et absolutt krav til telemetri-modulen er at den er trådløs og kan overføre data fra bilen til en basestasjon. Basestasjonen er tenkt plassert i nærheten av banen bilen kjører på. Det er også en fordel med to-veis kommunikasjon, da det åpner for justering av parametere fra sidelinjen. Rekkevidden i åpent landskap bør være noen titalls kilometer, ettersom banen har en utstrekning på flere kilometer. Signalene kan bli dempet både av bygninger og interferens.

Systemet må fungere selv om telemetri-modulen er i bevegelse, basestasjonen trenger ikke være flyttbar, men det er en fordel om den også kan være i bevegelse. Hastighetene er ikke større enn maks 40 km/t, men det kan være interessant å bruke modulen i vanlige biler som kan kjøre vesentlig fortere. Kravene er oppsummert i tabell 1.

1	To-veis trådløs kommunikasjon.
2	Rekkevidde på noen titalls kilometer.
3	Fungerer ved hastigheter på minst 40 km/t, helst enda raskere.
4	Minst 9600 bit/s båndbredde.
5	Enkel tilkobling til mikrokontroller.

Tabell 1: Krav til telemetri-modulens trådløse forbindelse

2.1.1 Overføringshastighet

Et raskt overslag for å beregne nødvendig overføringshastighet for radiolinken ble gjort som følger; En oppdatering av alle data hvert sekund er brukt som utgangspunkt. Data som skal overføres består av GPS-posisjon og diverse sensordata i form av strøm-, spenning- og temperatur-målinger fra brenselcelle. Status fra motorkontrolleren skal også være med. GPS-posisjonen er tenkt overført som en standard tekststreng fra NMEA-0183 formatet (National Marine Electronics Association).

NMEA-0183 standarden [1] har en tekststreng som heter RMC (Recommended Minimum Sentence C), som inneholder klokkeslett, dato, fart, posisjon, kurs og en sjekksum. Lengden er 70 tegn, altså 70 byte. Sensordata kan komprimeres og trenger ikke sendes som tekst, men å sende alt som en tekststreng er en enkel løsning både for sender og mottakersiden.

Et overslag på 50 måleverdier med 10 byte per verdi er brukt som utgangspunkt. 50 måleverdier kan virke mye, men det kan være interessant å måle individuelle cellespenninger i brenselcellen, den består av 46 celler. Tilsammen blir det 570 byte som må overføres hvert sekund, som igjen er 4560 bit/s. Dette er akkurat litt mer enn 4300 bit/s, som er en standardhastighet for rs232 seriell kommunikasjon. Neste standard hastighet er 9600 bit/s, da er det også muligheter for å utvide med mer data senere. Denne hastigheten bør derfor være et minimumskrav for den trådløse linken.

2.2 Kommunikasjon i bilen

For at telemetri-modulen skal få tilgang til måledata fra bilen er det nødvendig med et grensesnitt mellom telemetri-modulen og resten av elektronikken i bilen. Systemet som ble brukt i 2008 og 2009 er ikke designet med tanke på å kunne hente ut informasjon fra systemet i særlig grad. Brenselcellestyringen har noe loggemulighet, men er ikke koblet til motorstyringen på noen måte. Cruise-controllstyringen sitter i rattet og snakker kun med motorstyringen. Knapper for lys og brenselcellestyring er koblet direkte til det de styrer. Dette gjør det nærmest umulig å hente ut og eventuelt endre systemparametere fra en telemetri-modul.

På grunn av problemer med å koble telemetri-modulen til systemet i den nåværende bilen ble det bestemt at resten av systemet i bilen også må byttes ut. Dette arbeidet skal bli gjort av undertegnede utover våren i forbindelse med master-oppgaven. Likevel er det nødvendig å vurdere hvordan resten av systemet skal bli for å lage ferdig telemetri-modulen med et grensesnitt mot resten av systemet.

2.2.1 Brukervennlighet og modularitet

I og med at hele det elektroniske systemet skal designes på nytt er det ønskelig å gjøre en god jobb med brukervennligheten. Dette vil både lette arbeid og bruk av systemet for årets team, og gjøre det enklere å bruke for kommende eco-marathon-team. Et krav til telemetri-modulen i denne sammenhengen er at den skal kunne kobles inn og ut av systemet uten at resten av systemet trenger å rekonfigureres. Dette bygger også på tankegangen til designstudentene om å gjøre bilen modulbasert og enkelt rekonfigurerbar. For å gjøre inn og utkobling enkelt bør telemetri-modulen kun kobles til resten av systemet med en ledning som både bærer strøm og kommunikasjon. Det er også ønskelig om modulen starter av seg

selv og gjør sin jobb uten at noen knapper eller lignende må konfigureres og settes opp på forhånd.

2.2.2 Feilsikker kommunikasjon

I og med at bilen kan være et elektrisk støyende miljø er det viktig å kunne overføre informasjon mellom moduler i bilen på en feilsikker måte. Et krav blir dermed at kommunikasjonen mellom moduler skal baseres på en standard som tar høyde for feil og utfører feilretting/deteksjon. En standard som er vanlig for biler er ønskelig i denne sammenheng, siden bilen skal være en tilnærming til en ordentlig bil.

2.3 Logging

Trådløs kommunikasjon er i utgangspunktet ikke feilsikkert, det er veldig mange faktorer som kan gjøre at forbindelsen brytes eller faller ut midlertidig. Dette må det tas høyde for i telemetri-modulen. Å måtte kjøre et nytt løp, eller en ny test fordi måledata er gått tapt på grunn av en ustabil trådløs link er veldig unødvendig og må kunne forhindres. Det er derfor et krav at alle måledata skal logges i telemetri-modulen for å kunne hentes ut ved et senere tidspunkt. Størrelsen på denne loggen kan fort bli noen megabyte hvis det skal kjøres i flere timer (se 2.1.1).

2.4 Effektforbruk

Siden hele prosjektet og konkurransen går ut på energivennlighet og lite energiforbruk er dette også et viktig krav til elektronikken. Selv om elektronikken kan sees på som en del av sikkerhetssystemet og i følge reglene da kan drives av separate batterier, er det ønskelig at den bruker lite strøm. Det er vanskelig å stille noe fast krav til dette, men unødvendig sløsing skal unngåes. Dette kan foreksempel bety å bruke LED-belysning (Light Emitting Diode), benytte effektive spenningsomformere og å sette modulen i standby hvis bilen står stille.

2.5 GPS

For å ha tilgang til bilens posisjon og fart og knytte dette opp mot måledata i loggen og telemetri-dataene, er det nyttig med en innebygget GPS i telemetri-modulen. Dette kan også gjøre modulen veldig nyttig som en alenestående sporingsmodul. Dersom mobilnettet brukes for opplasting av data vil en kunne spore telemetri-modulens posisjon over internett

fra hvor som helst i verden. GPS vil også gi tilgang til klokkeslett og dato, dette kan brukes som en nøyaktig tidsreferanse for loggdata.

2.6 Oppsummering av spesifikasjonskrav

Kravene i tillegg til trådløsforbindelsen er listet opp i tabell 2.

1	Modularitet, enkelt å koble modulen inn og ut.
2	Kommunikasjon mellom moduler basert på en automotive standard.
3	Logge alle data i tilfelle trådløs kommunikasjon svikter.
4	Energieffektiv.
5	Innebygget GPS.
6	Mulighet for å bruke modulen som alenestående posisjonslogger eller liknende.
7	“Plug and play” -funksjonalitet.

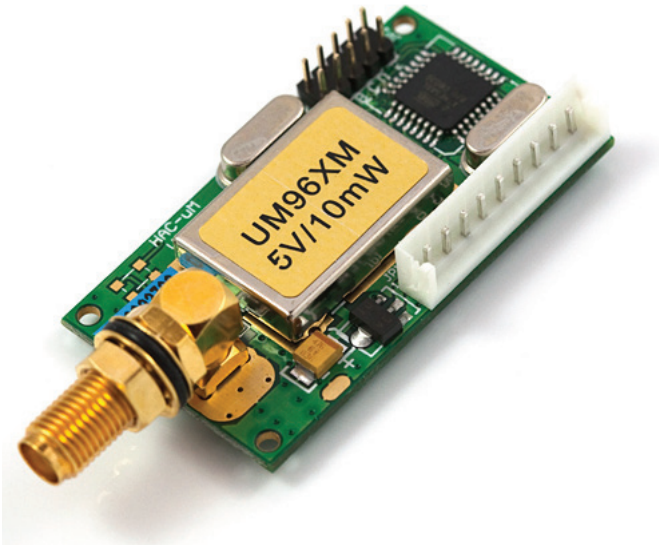
Tabell 2: Krav til telemetri-modulen i tillegg til den trådløse forbindelse

3 Designvalg

3.1 Trådløs kommunikasjon

Det er en rekke mulige veier å gå når det gjelder selve radiolinken. Av aktuelle radioforbindelser for telemetri-modulen er fire muligheter vurdert mot hverandre. En seriell FM-radiolink, ZigBee, Wi-Fi og GSM/GPRS/3g (Global System for Mobile communications)/(General Packet Radio Service)/Tredje generasjons mobilnett). Som vurderingskriterier er blant annet hastighet, rekkevidde, lisens på frekvensbruk, utvidelsesmuligheter og brukervennlighet vurdert. Et annet problem som også er vurdert er den trådløse kommunikasjonens evne til å fungere fra et kjøretøy i bevegelse. Det er ikke alle standarder som er laget for bruk i kjøretøy. Minimum båndbredde på kommunikasjonen er satt til 9600 bit/s, som funnet i seksjon 2.1.1.

3.1.1 Spesiallaget RF-modul



Figur 1: 9600 bit/s seriell radiolinkmodul

Det finnes veldig mange spesiallagde radiomoduler som tar inn serielle data og sender det ut på en gitt frekvens med foreksempel frekvensmodulasjon. Hastigheten er ofte rundt 9600 bit/s. En slik løsning kan sannsynligvis fungere for kjøretøy i bevegelse siden en vanlig FM-radio fungerer fint i biler. Forholdene kan imidlertid endre seg i løpet av en runde på banen. Det er også fare for interferens hvis åpne frekvensbånd benyttes.

Et eksempel på en slik modul er UM96XM (figur 1). Det ligger begrensninger på tillatt utsendt effekt og dermed vil rekkevidden være begrenset. En slik løsning krever sannsynligvis

godkjennelse for bruk fra teletilsynet i både Tyskland og Norge siden den skal brukes begge steder. Dette har blitt undersøkt tidligere i forbindelse med eco-marathon prosjektet [2]. Løsningen ble da forkastet nettopp på grunn av at det er vanskelig å få godkjent bruk med nok utsendt effekt til at rekkevidden blir god nok.

3.1.2 ZigBee

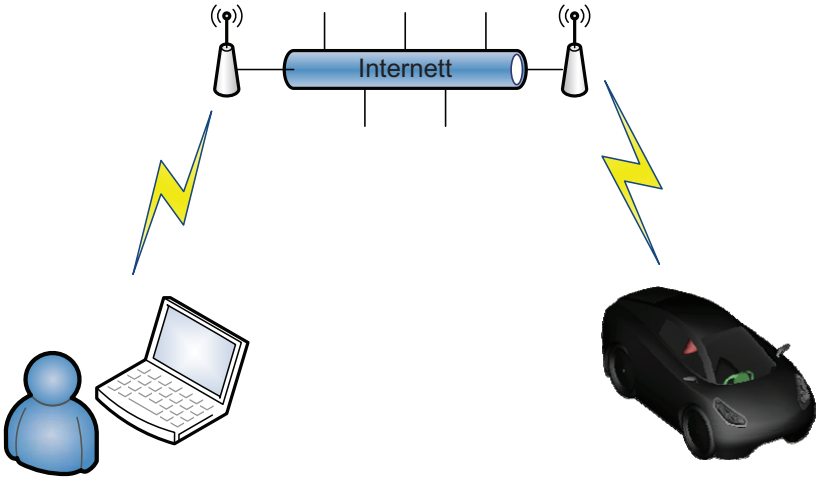
ZigBee er en egen standard for laveffekt radiosamband i sensornettverk. Standard hastighet for ZigBee-nettverk er 250 kbit/s, men det finnes også versjoner som opererer på 9600 bit/s for å få lenger rekkevidde. Begge deler holder for telemetri-applikasjonen. Fordelen med ZigBee er at modulene opererer i det lisensfrie 2.4 GHz-båndet. Ulempene er maks tillatt utstrålt effekt som er begrenset til noen få milliwatt. En eventuell løsning med ZigBee måtte derfor hatt antenner med høy forsterkning, eventuelt retningsbestemt antenne, men det er upraktisk siden bilen beveger seg. ZigBee er heller ikke en standard som er laget for bruk i kjøretøy, det betyr at selv om det sannsynligvis ville fungert, så kan uforutsette problemer oppstå. ZigBee ble forkastet på grunn av liten rekkevidde og uforutsette problemer som kan dukke opp ved bruk i bil.

3.1.3 Wi-Fi

Wi-Fi, eller IEEE 802.11 er den vanligste trådløse standarden for pc'er og støtter veldig høye overføringshastigheter. Opp mot 54 Mbit/s er vanlig, og den nye IEEE 802.11n varianten går helt opp til 200 Mbit/s [3]. Rekkevidden er også her et problem, og som for ZigBee måtte antenner med stor forsterkning blitt brukt, selv da er det godt mulig det ikke hadde fungert. Standarden er heller ikke laget for kjøretøy i bevegelse. Det vanligste bruksområdet er kommunikasjon mellom en stasjonær basestasjon og pc'er som for det meste står stille, signalkvaliteten blir forringet ved bevegelse [4].

3.1.4 GSM/GPRS/3G

Den siste løsningen som er vurdert er datakommunikasjon over mobilnettet (figur 2). Mobiltelefoni ble brukt til kommunikasjon med sjåføren av eco-marathon teamet i både 2008 og 2009. Det var mange grunner til dette, mest viktig for valget av mobiltelefon var at lisens for frekvenser ikke er nødvendig og rekkevidden er heller ikke et problem. Siden det er data som skal overføres er sannsynligvis GPRS eller 3g den beste løsningen. Dette er pakkesvitsjet kommunikasjon over mobilnettet [5]. Fordelen er at man vanligvis bare betaler for overført data, og ikke oppkoblingstiden. Maksimal hastighet for GPRS ligger på rundt 40 kbit/s [6], som er mer enn nok for telemetriapplikasjonen. Reell hastighet vil sannsynligvis ligge et stykke under maksimal hastighet.



Figur 2: Telemetri-modul og basestasjon

En løsning basert på 3g ble ikke vurdert videre fordi det er mer komplekst og dyrere i bruk. En fordel med GPRS er at kommunikasjonen foregår over internett, dette gjør det mulig for flere å følge med fra forskjellige pc'er, og også sitte hvor som helst hvor det er internett. Ulempen er at det må være internett tilgjengelig ved banen, men det er igjen mulig å bruke internett gjennom en mobiltelefon med GPRS/3g. Siden alle de andre radiolink-variantene hadde usikkerhetsmomenter knyttet til rekkevidde og kanskje problemer rundt lisenskrav, ble GPRS valgt som kommunikasjonslink (tabell 3).

	Spesial-modul	Wi-Fi	ZigBee	GPRS
Båndbredde	-	+	+	+
Rekkevidde	-	-	-	+
Lisenskrav	-	+	+	+
Bruksområder	-	+	+	+
Bruk i bevegelse	+	-	-	+

Tabell 3: Vurderingstabell for trådløs kommunikasjon

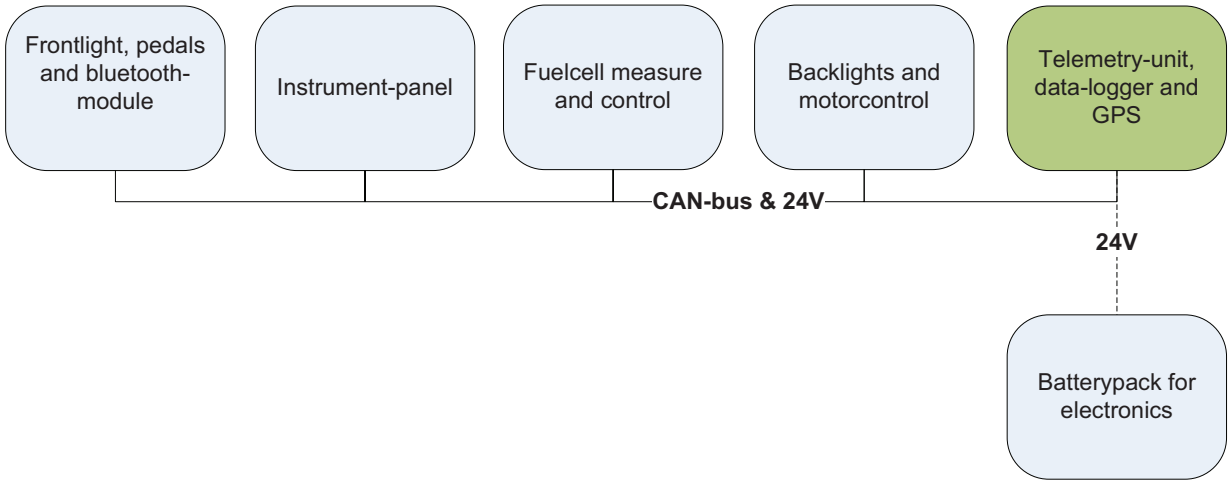
3.2 Kommunikasjon i bilen

For å kommunisere med resten av enhetene i bilen må en kommunikasjonsstandard velges. Teamet i 2009 brukte seriell rs232-basert kommunikasjon med motorstyringen, men ellers var det lite kommunikasjon mellom modulene. Ettersom en av teamets målsettinger for 2010 er å lage en bil med gjennomførte designløsninger, er det også ønskelig at elektronikken er på høyde med løsninger som brukes i ordentlige biler. Tre typer nettverksbusser er vanlig i moderne biler, LIN-bus (Local Interconnect Network), CAN-bus (Controller Area Network) og FlexRay [7]. Alle tre bussene er multidrop-busser, det betyr at alle enheter som snakker

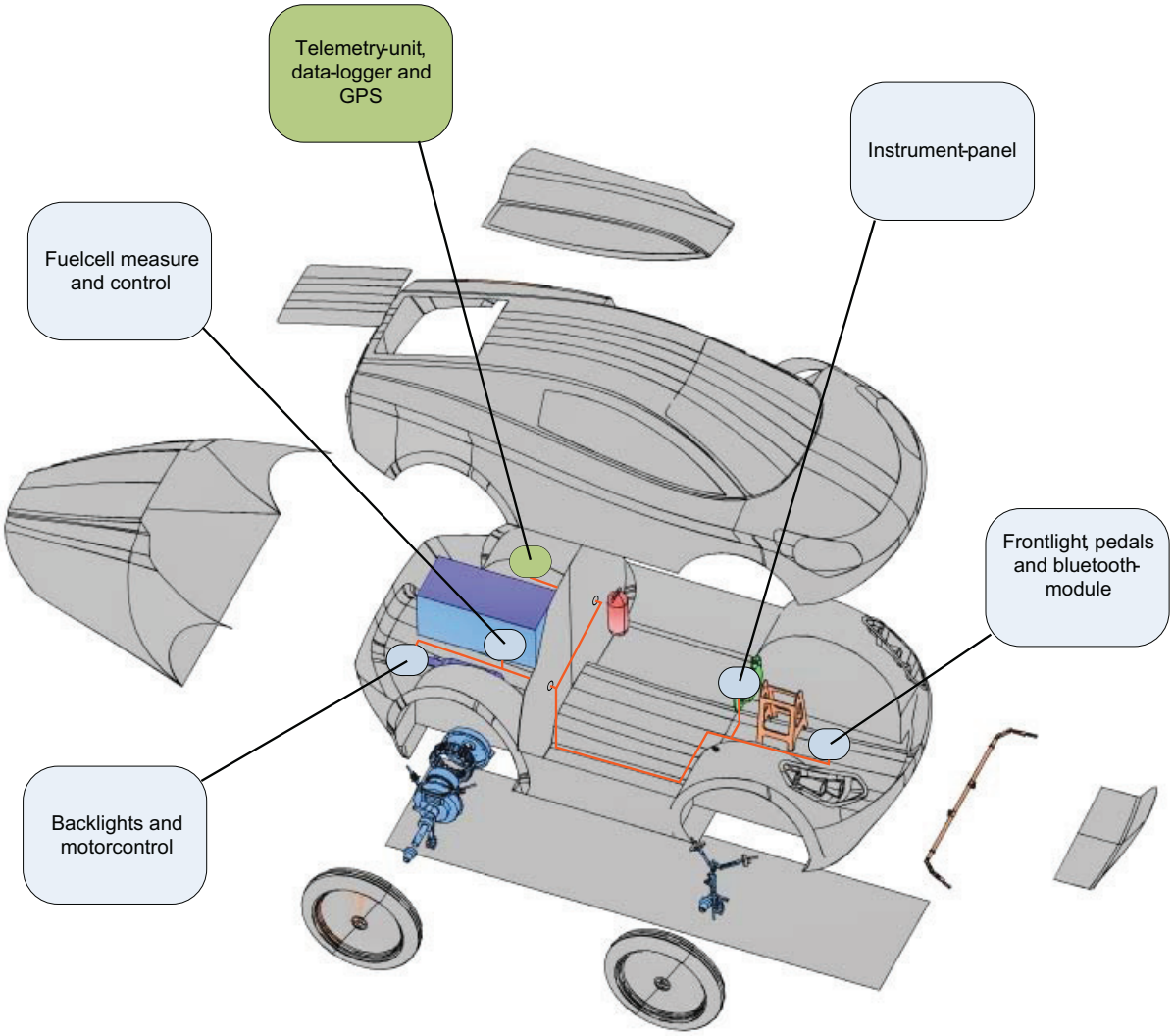
sammen er koblet til de samme ledningene. Dette minimerer antall meter ledning som må til for å koble alle modulene sammen.

Valget av kommunikasjonsstandard falt på CAN-bus siden multimaster er viktig for modulariteten, det betyr at alle moduler kan initiere kommunikasjon selv, og sende beskjeder til alle andre moduler som er koblet til. CAN-bus har også en rekke fordeler med feildeteksjon og høyere overføringshastighet i forhold til LIN eller rs232. FlexRay er for avansert igjen, og har unødvendig stor overføringshastighet (tabell 4).

På grunn av CAN-bus sin mulighet for at enhver tilkoblet node kan sende beskjeder på bussen kan moduler i bilen fritt kobles til og fra. Telemetri-modulen kan dermed kobles til og fra etter behov og plukke opp alle meldinger den trenger for logg og telemetri. Modulen kan i tillegg sende meldinger ut på bussen ved to-veis kommunikasjon. Et foreløpig utkast til et slikt system er vist skjematisk i figur 3 og et eksempel på fysisk plassering er vist i figur 4. Beskrivelse av de forskjellige kommunikasjonsbussene følger fra 3.2.1.



Figur 3: Skematisk oppsett for elektronikken i bilen



Figur 4: Forslag til fysisk plassering av elektronikkmoduler i bilen

Bus	LIN	CAN	FlexRay
Hastighet	40 kbit/s	1 Mbit/s	10 Mbit/s
Kostnad	$	$$	$$$
Ledninger	1	2	2-4
Typisk anvendelse	Speil, setejustering, soltak, annet tilbehør	Motor, gir-utveksling, ABS	Drive-by-wire, aktiv demping, adaptiv cruise-control, infotainment

Tabell 4: Tre alternative kommunikasjonsbusser for biler [8])

3.2.1 LIN-bus

LIN-bus er ment som en subnettverksbus for enkle sensorer og aktuatorer i biler. Soltak, regnsensor og ventilasjonstyring er eksempler på bruksområder. LIN-bus er en single-master bus, det betyr at det kun er en enhet på bussen som kan initiere kommunikasjon på bussen. Hastigheten er begrenset til 40 kbit/s. Det er tilstrekkelig med kun en ledning i tillegg til jord. Det går også an å legge kommunikasjonen over 12 V-lederen.

3.2.2 CAN-bus

CAN-bus er en multimaster bus, som er basert på meldings-id'er framfor adressering av en bestemt mottaker. Bussen benytter bitvis arbitrering, som har den effekten at meldinger med lav id får høyere prioritet enn meldinger med høyere id. Bussen støtter hastigheter opp mot 1 Mbit/s og benytter to ledninger med balansert differensiell overføring. Dette gir høy støyimmunitet og tillater høyere hastighet enn LIN-bus.

3.2.3 FlexRay

FlexRay er en ny kommunikasjonsstandard som har enda høyere hastighet enn CAN-bus. Den benytter også to ledere, er differensiell, og støtter hastigheter opp mot 10 Mbit/s. FlexRay benytter TDMA (Time Division Multiple Access) for å holde orden på hvem som sender og mottar. Dette gjør at en enhet er garantert å få overført data med regelmessige, gitte mellomrom. Dette er særlig gunstig i reguleringssløyfer, der tidskrav er viktig. FlexRay benyttes foreksempel i det aktive dempersystemet i nye BMW X5 [9].

Figur 5: Telit-modul med GSM/GPRS og GPS (Sparkfun [10])

3.3 Mobiltelefonmodul

For å få til GPRS-kommunikasjon er det nødvendig med mye elektronikk og prosesseringskraft. Dette fordi både radiokommunikasjonen og TCP/IP-protokollen (Transmission Control Protocol/Internet Protocol) utføres av forholdsvis avanserte algoritmer som krever en del prosesseringskraft og minne. Alle moderne mobiltelefoner med støtte for websurfing og lignende har allerede alt dette innebygget, men det er ikke nødvendigvis like lett å få sendt egne data over en mobiltelefon sin GPRS-tilkobling.

En fullstendig konfigurerbar mobiltelefon-modul med mulighet for å kjøre egen kode hadde vært en optimal løsning. Etter litt leting på nettet og tips fra veileder falt valget på en Telit-modul [11]. Telit er et italiensk selskap som lager moduler for trådløs kommunikasjon til innbygging i annen elektronikk. De fleste modulene baserer seg på GSM/GPRS og har innebygget støtte for standard AT-kommandoer [12]. Modulene kan kjøre enkle Python-skript for å gjøre oppsett og konfigurasjon i en selvstendig enhet mulig [13]. Siden telemetrimodulen også skal ha GPS, falt valget på en GM862-GPS (figur 5).

Modulen har innebygget simkortholder og GPS-mottaker. To antennekontakter, en for GPS, og en for GSM er tilgjengelig langs kanten på enheten. Resten av tilkoblingene er tilgjengelig i en liten kontakt på undersiden. For å gjøre utvikling og testing enklere var et utviklings-kit nødvendig. Valget falt på en USB-variant (Universal Serial Bus) fra www.sparkfun.com (figur 6).

Figur 6: Utviklingskit for Telit-modul (Sparkfun [10])

3.3.1 Utviklingskort

Kortet fra sparkfun består av en USB-kontakt, ekstern strømtilkobling, spenningsregulator og en USB til UART krets (Universal Asynchronous Receiver/Transmitter). Kortet kan trekke strøm rett fra USB-kontakten, den eksterne strømtilkoblingen er derfor overflødig når en pc er tilgjengelig. USB til seriell kretsen [14] gjør at kortet dukker opp som en virtuell serieport på pc'en når det kobles til og driveren er installert. Dette er meget praktisk da man slipper spenningskonvertering fra rs232-nivåer på 7 V-15 V ned til 3.3 V [15]. Mange nye pc'er kommer også helt uten vanlig serieport, og derfor er det praktisk med en USB-tilkobling.

3.4 Logging

Ettersom sanntidsoverføring av informasjon fra bilen er avhengig av en fungerende GPRS-tilkobling og internettforbindelse bør det tas høyde for at denne kommunikasjonen kan svikte. Derfor er det, som nevnt i kravspesifikasjonen, viktig at telemetri-modulen kan lagre en logg med alt som skjer, som så kan granskes i ettertid. Dette vil gi telemetri-modulen akkurat samme funksjonalitet som systemer brukt til logging tidligere.

For å loggføre data er det mest aktuelt med enten EEPROM- (Electrically Erasable Programmable Read-Only Memory) eller flash-basert minne. Dette er minnetyper som kan

skrives og slettes flere ganger og beholder alle data ved strømbrudd. Ettersom datamengdene under et løp kan komme opp i flere megabyte er flash-minne mest aktuelt. Forskjellen på EEPROM og flash-minne er størrelsen på område som må slettes før en skriveoperasjon kan foretas. EEPROM kan slettes og skrives bit for bit. Flash-baserte enheter derimot må som oftest slette mange kilobyte av gangen, derav navnet "flash". Dette gir flash mye høyere lagringstetthet fordi transistorer for sletting benyttes på et større volum av minneceller [16].

Små enheter som minnekort på flere gigabyte kan fremstilles med flash-teknologien. Det finnes også egne kretser med flash-minne, men den billigste løsningen er å bruke et standard minnekort for digitalkameraer og lignende. Det mest aktuelle alternativet, på grunn av enkelt grensesnitt er SD-kort (Secure Digital [17]), eller microSD kort som tar mindre fysisk plass. SD-kort benytter seg av SPI (Serial Peripheral Interface) til dataoverføring, det passer dermed bra for mikrokontrollere. Spenningsnivået for I/O-linjer og forsyning for SD-kort og microSD-kort er 2.7 V - 3.6 V.

SD-kort en veldig vanlig minnekortstandard for digitalkameraer og lignende forbrukerutstyr. Dette gjør det enkelt å hente data ut ved hjelp av en vanlig pc og minnekortleser. Dette forutsetter imidlertid at et filsystem som FAT (File Access Table [18]) implementeres på mikrokontrolleren. SD-kort baserer seg på flash-minne og har en innebygget kontroller som tar seg av lese og skriveoperasjoner på blokknivå, dette betyr for eksempel at en slipper å slette en blokk før en skriveoperasjon utføres.

3.5 Mikrokontroller

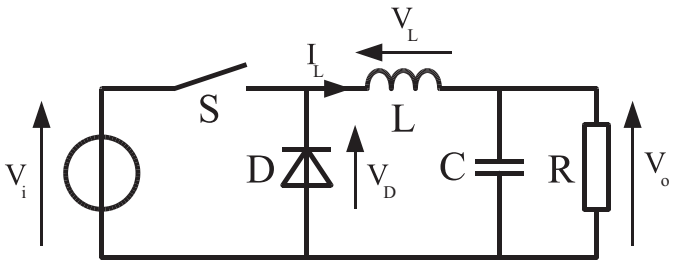
For å implementere et system som utfører både loggføring til minnekort, kommunikasjon med en trådløs modul og kommunikasjon over CAN-bussen er det nødvendig med en mikrokontroller. Fortrinnsvis en med innebygget CAN-funksjonalitet. Det er ikke særlig høye krav til ytelse, en 8-bit mikrokontroller er derfor tilstrekkelig. Atmel sine mikrokontrollere med innebygget CAN-bus ble et naturlig valg siden de er lett tilgjengelige via ITK's komponentlager (Institutt for Teknisk Kybernetikk).

AT90CAN-serien er mikrokontrollere som kommer i 64-pins pakker og har dermed mange I/O-linjer (Input/Output) og mange muligheter for utvidelse via både SPI- og I2C-busser (Inter-Integrated Circuit [19]) [20]. AT90CAN-serien kan kjøre på spenninger fra 2.7 V og helt opp til 5 V. På grunn av spenningsnivå på I/O-linjene til både minnekort og Telit-modul ble 3.3 V valgt som driftspenning for mikrokontrolleren. Dette begrenser øvre klokkefrekvens til 10 MHz [20].

3.6 Strømforsyning

Elektronikken ble i 2009 forsynt med strøm fra to 12 V-batterier i serie. Dette gjorde det mulig å hente ut både 12 V og 24 V fra batteriene. Det elektriske hornet som er påbudt å ha i bilen går på 12 V. Trykksensorer og solenoid-ventiler på hydrogentilførselen går på 24 V, derfor var det nødvendig med både 12 V og 24 V i fjor. Ettersom det fortsatt er uklart hva som skal gjenbrukes på systemet videre, er det ønskelig å holde alle muligheter åpne. Elektronikken bør derfor kunne kjøres på både 12 V og 24 V. Fordelen med høy spenning er at strømmen ikke trenger å være så høy ved samme overførte effekt. Dette gjør at ledninger i bilen kan være tynnere.

3.6.1 DC-DC omformer



Figur 7: Skematisk tegning av DC-DC buck-omformer

Ettersom telemetri-modulen også skal fungere som en selvstendig utviklingsmodul for Telit-kretsen bør den kunne drives fra kun USB-tilkoblingen uten 24 V tilgjengelig. USB-kontakten kan gi 500 mA ved 5 V. Dette gjør at utgangsspenningen fra spenningsomformeren som tar inn 24 V også bør gi 5 V ut. For å konvertere 24 V ned til 5 V må man bruke en såkalt DC-DC buck-omformer (Direct Current) (figur 7). Dette er en type spenningsomformer som konverterer en likespenning til en annen likespenning uten vesentlige effekttap.

Konverteringen gjøres ved at omformeren svitsjer inngangsspenningen på og av med varierende på-tid avhengig av nivået på inngangsspenning i forhold til utgangsspenning. En spole brukes til å mellomlagre energien. Spolen holder strømmen ved utgangen høy når svitsjen er av og tar opp energi som lagres i magnetfeltet til spolen når svitsjen er på. Ulempen med svitsjene regulatorer er at de sender ut en god del elektrisk støy. Støyen kan begrenses ved fornuftig design av kretsen og adekvat støyfiltrering med kondensatorer. For oppsummering av valgte DC-DC-omformerparametere se tabell 5.

	Min	Typ	Maks
Inngang	6 V	24 V	30 V
Utgang	4.9 V	5.0 V	5.1 V
Strøm	0 A	200 mA	2.5 A

Tabell 5: Valg av spenning og strøm for DC-DC-omformer

3.6.2 Lineær regulator

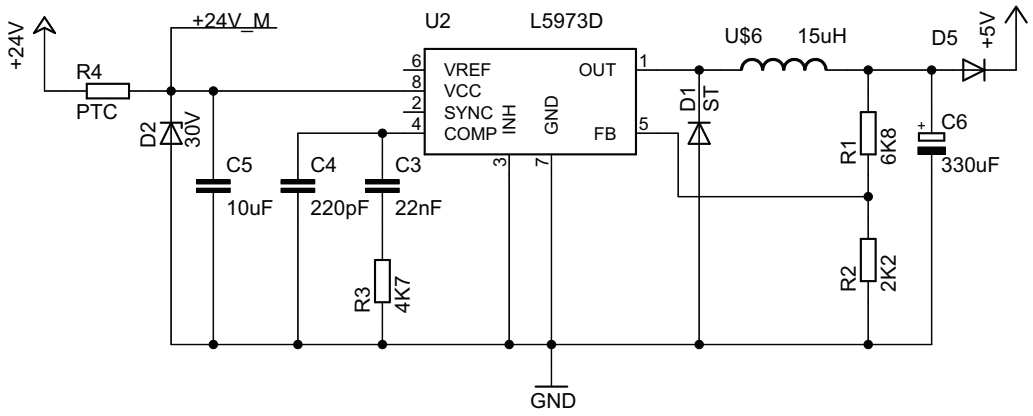
Alternativet til en svitsjene DC-DC omformer er en lineær regulator. Dette er i sin enkleste form en transistor mellom inngang og utgang som styres av en spenningsreferanse og komparator. Lineære regulatorer kan brukes når det ikke er for stor spenningsforskjell eller strøm som trekkes. En lineær regulator må selv dissipere differansen i effekt som spenningsforskjellen mellom inngang og utgang gir ved en gitt strøm. Dette gjør at de er lite energieffektive.

Lineære regulatorer er gode med tanke på støy, og brukes derfor ofte for støysensitive kretser. For telemetri-modulen kan det være gunstig å bruke både lineær regulator og svitsjene regulator. DC-DC omformeren kan ta 24 V ned til 5 V, og en lineær regulator kan ta 5 V ned til for eksempel 3.3 V og samtidig undertrykke støy fra DC-DC omformeren.

4 Detaljspesifikasjon

For å gjøre telemetri-modulen brukervennlig og robust må den bygges inn i en boks. Tilkoblingsmulighetene må være enkle, og det må være vanskelig/umulig å koble noe feil. Designet må ta høyde for overspenning og polaritetsfeil på inngangsspenningen. Første idé var å bygge videre på utviklingskortet fra Sparkfun, men det viste seg vanskelig å finne noen boks som passet til dette kortet. Det andre alternativet var å lage et kretskort for Telit-modulen med alt på det samme kretskort. Etter å ha funnet en passende boks å ha systemet i ble den siste løsningen valgt. Kretskortet kunne da spesiallages til boksen.

4.1 DC-DC omformer



Figur 8: Kretsløsning for DC-DC omformer

Utgangsspenningen er fastsatt til 5 V, siden det er det samme som USB-bussen leverer. Det er ønskelig at inngangsspenningen kan varieres fra litt over 5 V opp til høyeste batterispenning, og gjerne helt opp til over 30 V (tabell 5). Dette vil gi god margin med tanke på at batteri-spenningen fra foreksempel to bly-batterier i serie kan være ganske høy. Et 12 V bly-batteri kan ha en spenning på opp i mot 13 V-14 V, to i serie vil dermed gi 26 V-28 V.

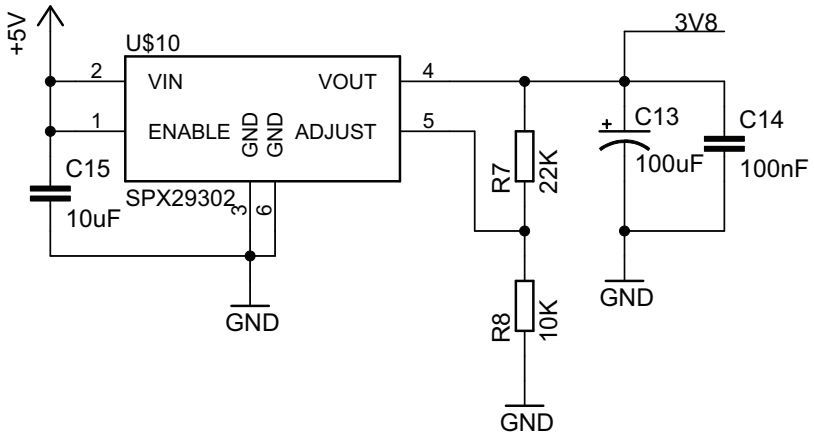
Kretsløsningen som ble valgt er lånt fra databladet for Telit-modulen for å være sikker på at den kan lever nok strøm. Telit-modulen kan trekke opp til 2.5 A under sending [11]. For å beskytte mot overspenning og omvendt polaritet er det plassert en 30 V zener-diode over inngangen (figur 8). Strømbegrensning ved overspenning og kortslutning er løst ved å plassere en PTC-motstand (Positive Temperature Coefficient) i serie med den positive polen ved inngangen.

Ved for stor strøm vil PTC-motstanden bli varm og varmen gjør at motstanden igjen øker på grunn av den positive temperaturkoeffisienten. Dette vil fungere som en sikring ved at

hele inngangsspenningen vil ligge over motstanden når en stor strøm har gått gjennom den. Normal operasjon opprettes igjen ved å skru av spenningstilførselen og vente til motstanden er kald. En slik beskyttelse er veldig enkel å implementere og er samtidig enklere i bruk enn en vanlig sikring som må byttes hvis den har blitt trigget. En diode er plassert i serie med utgangen for å forhindre at det går en strøm inn i DC-DC-svitsjekretsen når kun USB-strøm er koblet til.

4.2 Telit spenningsregulator

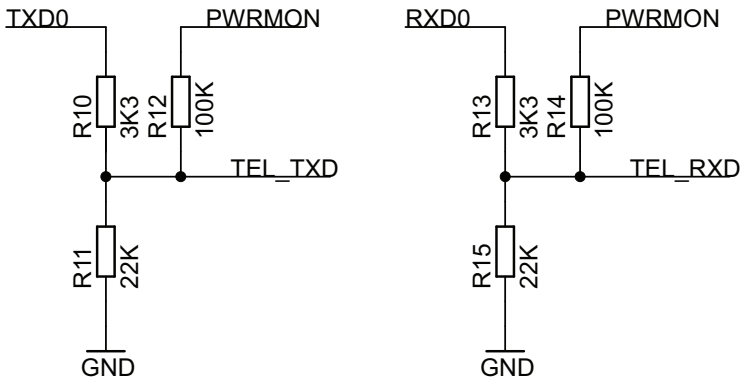
Telit-modulen krever en spenning på mellom 3.4 V og 4.2 V. Den enkleste løsningen var å kopiere kretsløsningen med en lineærregulator fra utviklingskortet. Denne løsningen var allerede testet og fungerte med 5V fra USB-porten. En lineær regulator vil også minske støy fra 5 V-bussen. Spenningen ut fra regulatoren bestemmes av spenningsdelingen ved utgangen. Den ble valgt til 3.9 V, samme som på utviklingskortet.



Figur 9: Kretsløsning for lineærregulator til Telit-modul

4.2.1 Interface mot Telit-modulen

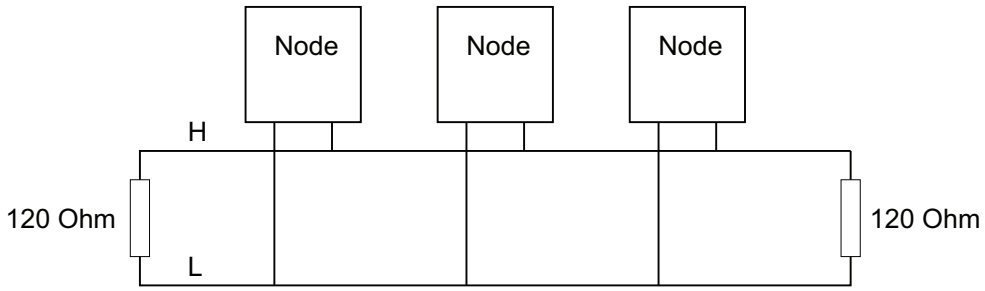
Telit-modulen har litt spesielle spesielle krav til signalnivåer på I/O linjer. Høyt nivå skal ligge på 2.8 V selv om spenningsforsyningen ligger på mellom 3.4 V og 4.2 V. Av alternative muligheter for kommunikasjon med en mikrokontroller er både UART, SPI og I2C mulig. Ettersom det kan bli ønskelig å sende AT-kommandoer til Telit-modulen fra mikrokontrolleren må UART benyttes. Dette gjør at spenningsnivåtilpassing blir enkelt, UART benytter en linje til sending, og en til mottak. Dermed kan en enkel spenningsdeler



Figur 10: Spenningsdeling for signalnivåkonvertering

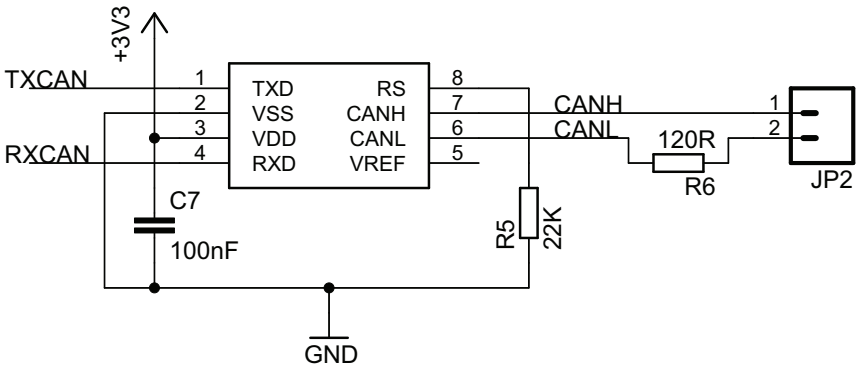
på Tx-linjen benyttes (figur 10). Ettersom mikrokontrolleren kjører på 3.3 V vil 2.8 V på Rx-linja bli tolket som en høy verdi [20].

4.3 CAN-bus



Figur 11: CAN-bus terminering med 120 Ω motstander

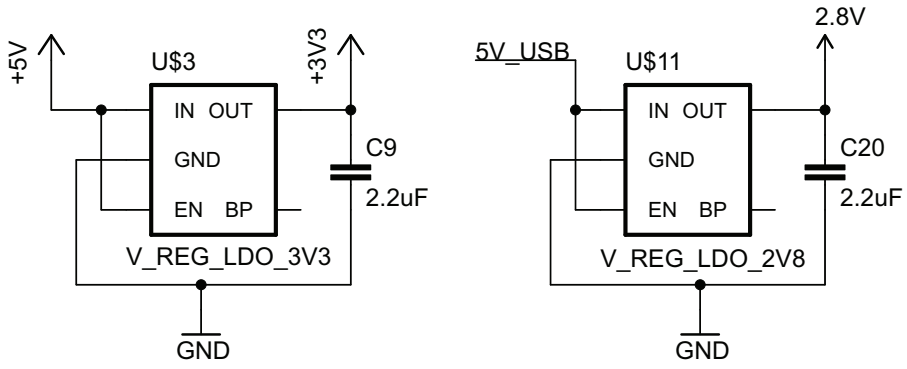
En CAN-node består vanligvis av en CAN-kontroller og en differensiell transceiver [7]. CAN-bussen består av to ledninger, CAN-H og CAN-L. Ved en endenode i nettverket skal bussen termineres med en 120 Ω motstand (figur 11). Dette er spesielt viktig ved lange avstander og høy overføringshastighet. Termineringen minimerer refleksjon av signalet. Mikrokontrolleren som er benyttet har en innebygget CAN-kontroller, derfor er bare en transceiver nødvendig som ekstra komponent. En veldig vanlig CAN-bus-transceiver er MCP2551 [21]. Denne kunne blitt brukt, men krever 5 V for å fungere. Mikrokontrolleren kjører på 3.3 V på grunn av microSD kort og kommunikasjon med Telit-modulen. Derfor ble en 3.3 V CAN-transceiver valgt i stedet [22]. Denne er også kompatibel med 5 V transceivere på andre noder.



Figur 12: Kretsløsning for CAN-bus transceiver

4.4 Lineær spenningsregulator

En AT90CAN mikrokontroller trekker rundt 10 mA [20], SD-kort, lysdioder og CAN-transciever vil maksimalt kunne trekke litt over 100 mA på grunn av at SD-kortet har dette som maks strømtrekk ved en sletteoperasjon [23]. En lineær regulator på rundt 150 mA som tar spenningen ned fra 5 V til 3.3 V vil fungere bra. Dette er enkle kretser som ofte ikke trenger mer enn en avkoblingskondensator for å fungere (figur 13).

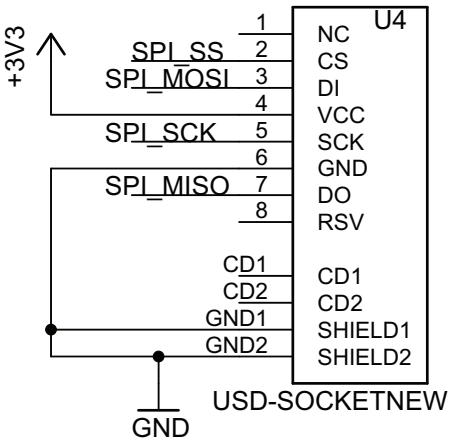


Figur 13: Kretsløsning for spenningsregulatorer til mikrokontroller og signalreferanse for Tx/Rx-linjene

4.5 Micro-SD kort

For å lagre logg-data er et microSD kort benyttet. MicroSD ble valgt fordi det tar vesentlig mindre plass enn et vanlig SD-kort. MicroSD-kortet trenger 2.7 V - 3.6 V og kan derfor kobles til samme spenning som AT90CAN og CAN-transceiveren. SPI-linjene kan også

kobles direkte til mikrokontrolleren.



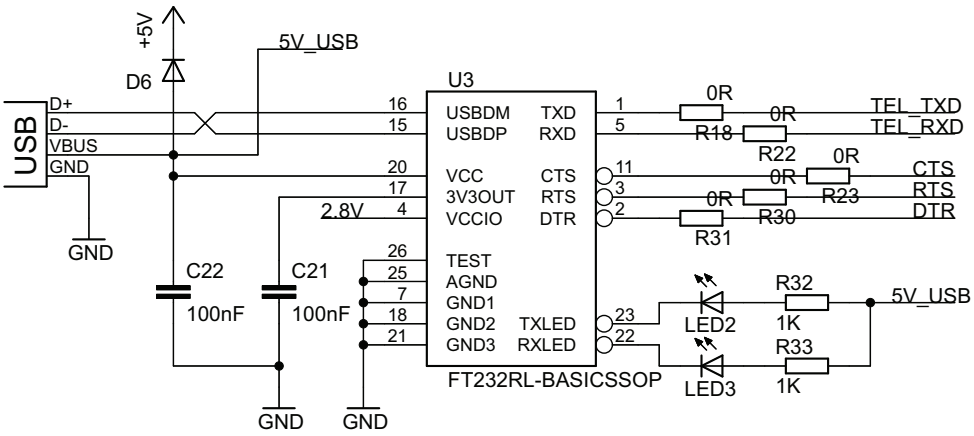
Figur 14: MicroSD-socket skjemategning

Figur 15: MicroSD-kort

4.6 USB tilkobling

Det er ønskelig med enkel tilkoblingsmulighet til pc, samme kretsløsning som ble brukt på utviklingskortet for Telit-modulen er valgt. Dette gjør at Telit-modulen kan få AT-kommandoer fra pc'en på akkurat samme måte som da utviklingskortet ble brukt. Dette

vil gjøre det enkelt å utvikle programvare senere. FT232R [14] er brukt for å få en virtuell serieport på pc'en via USB. Rx og Tx linjene fra FTDI-brikken er koblet rett til Telit-modulen siden I/O spenningen er satt til 2.8 V med den separate lineærregulatoren. Telit-modulen har automatisk innstilling av hastighet på serieporten, dette gjør at det meste fra 9600 bit/s opp til 115.2 kbit/s fungerer uten problemer.



Figur 16: USB til seriell kretsløsning

4.7 Spenningsmålinger

Siden telemetri-modulen kan bli forsynt med strøm fra både USB-kontakten og bilens nettverkskontakt er det nyttig å kunne måle både 5 V-bussen og 24 V forsyningsspenningen. Dette kan enkelt gjøres ved å bruke den interne analog til digital konverteren i mikrokontrolleren. Denne har et inngangsområde tilsvarende forsyningsspenningen til mikrokontrolleren. Både 5 V og 24 V må derfor senkes til under 3.3 V med en spenningsdeler. Motstandsverdiene som er valgt vil gi lite strømtrekk gjennom spenningsdelingen.

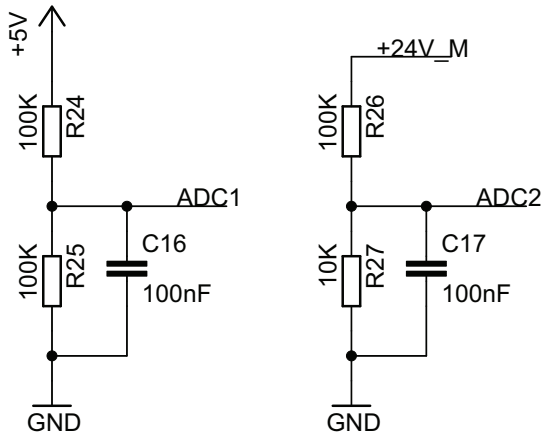
Ved å måle om 24 V spenning er tilstede kan mikrokontrolleren oppføre seg forskjellig, avhengig av om den er koblet til i bilen eller om USB er koblet til. En kan også finne ut om batterispenningen begynner å bli for lav. 5 V målingen kan brukes til å sjekke om DC-DC omformeren fungerer som den skal. Det er uansett bedre å legge opp til en måling for mye enn en for lite.

Hvis det er ønskelig kan 3.3 V spenningen måles internt i mikrokontrolleren. Dette gjøres ved å måle den interne 1.1 V båndgap-referansen med ADC'en (Analog to Digital Converter) og regne seg tilbake til hva ADC'en sin referansespenning er (2). Dette er en enkel utregning, adc er tallverdien fra analog til digital-konverteren og 2^n er oppløsningen til ad-konverteren:

$$V_m = \frac{\text{adc}}{2^n} \cdot V_{\text{ref}} \tag{1}$$

Her er V_m spenningen adc'en måler som skal være 1.1 V, og V_{ref} er 3.3 V referansen en ønsker å finne. Ved å snu på ligningen gir det:

$$V_{\text{ref}} = \frac{2^n}{\text{adc}} \cdot V_m \tag{2}$$



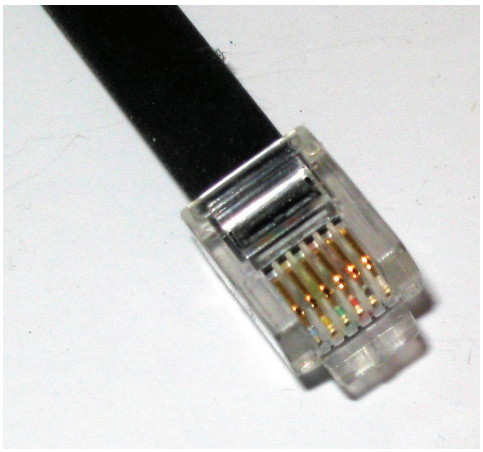
Figur 17: Spenningsdeling for å måle forsyningsspenning

4.8 Kobling mellom moduler i bilen

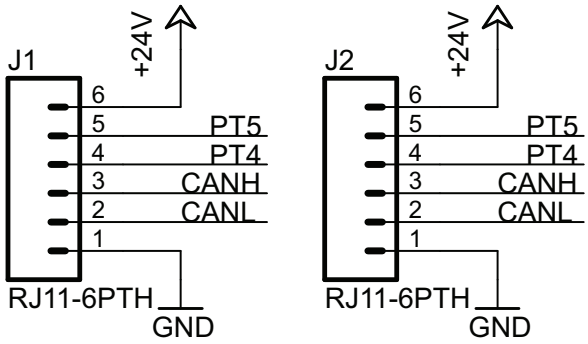
Et hovedkriterie for det nye systemet i bilen er å minske antall ledninger, og gjøre det enkelt å plugge ting inn og ut. I 2009 ble vanlige serieport-ledninger med DB9-plugger til pc'er benyttet i bilen. Dette er også en vanlig kontakttype for CAN-bus. Ulempene med denne typen plugg er både størrelse, vekt, og den er vanskelig å montere selv på en ledning. Dette gjorde at ferdige ledninger ble benyttet i bilen tidligere, de er ofte altfor lange, og mye ekstra ledning ligger spredt rundt i bilen.

Til det nye konseptet ble RJ-11-kontakter (Registeret Jack) valgt. Dette er samme type plugger som vanlige nettverksledninger benytter, men med kun 6 ledere i motsetning til nettverksplugger som har 8. Dette gir en litt smalere kontakt og vil fysisk hindre feilkobling med vanlig nettverksutstyr. RJ-11 betegnelsen er egentlig en beskrivelse av hvordan kontakten er koblet opp, selve kontakttypen heter Modular-jack. RJ-11 er brukt videre i dette dokumentet for å betegne at det er snakk om en modular-jack-kontakt med 6 ledere. Hver modul i systemet vil få to RJ-11 kontakter slik at T-ledd på ledningene blir unødvendig, signal og strøm passerer bare rett gjennom modulen og videre til neste modul.

To av lederne blir spenningsforsyning og jord, to går med til CAN-bussen, og to ledninger er ekstra for eventuelle utvidelser senere. RJ-11 kontaktene har mothake på pluggen, som gjør at den ikke kan falle ut av seg selv. Dette er en viktig egenskap i en bil hvor ting kan riste ganske mye. Pluggene er lett å montere på ledningen med en spesial-tang, og ledningen er enkel å håndtere. Maksimal strøm i ledning og kontakt er oppgitt til 1 A per leder. Dette gir 24 W ved 24 V på ledningen. 24 W er mer enn nok for elektronikken i modulene som kobles til, men kjørelys og horn bør få egen forsyningsledning da hornet alene kan trekke 50 W [2].



Figur 18: Modular-jack kontakt



Figur 19: Pinout for RJ11-konnektorene

4.9 Antenner

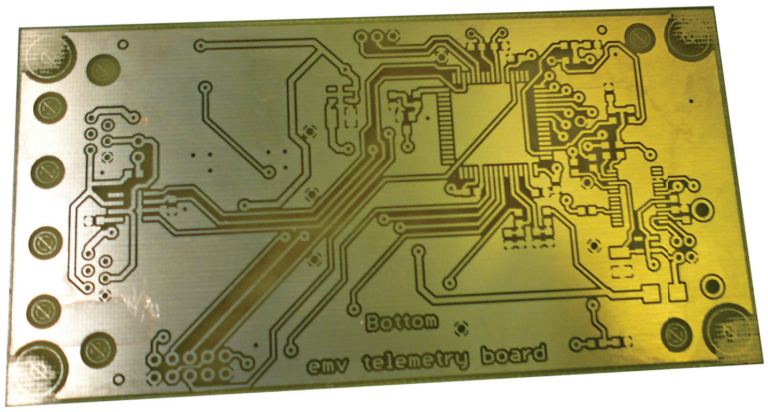
Antennene som telemetri-modulen trenger er en aktiv GPS-antenne og en passiv rundstråle mobil-antenne. Begge antennene fulgte med utviklingskortet til Telit-modulen. GPS-antennen ble modifisert for å få plass inne i boksen ved å korte inn ledningen og ta bort

plastikkdekslet rundt antennen. Mobiltelefonantennen ble i utgangspunktet forsøkt plassert inne i boksen, dette viste seg å bli en problematisk plassering, problemet er beskrevet i seksjon 6.

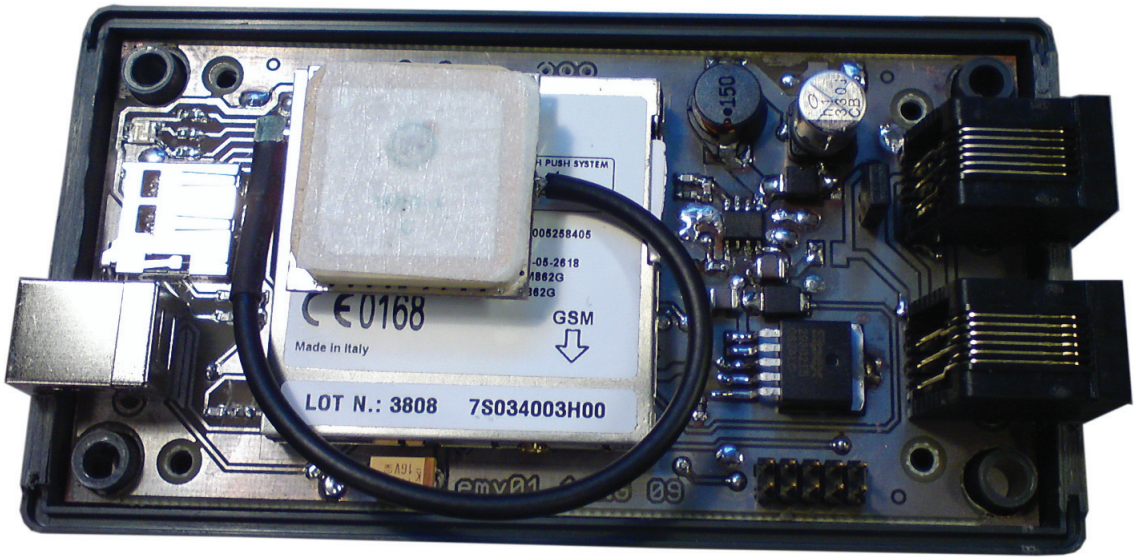
4.10 Kretskort og boks

Kretskortet ble spesialtilpasset til en elektronikkboks levert av komponentlageret ved Institutt for teknisk kybernetikk. Et problem med boksen er at eco-marathon-reglene spesifiserer et krav til elektronikkbokser, de må være gjennomsiktige eller ha et vindu. Dette er planlagt løst ved å legge inn et vindu av plexiglass i lokket på boksen. Regelen er sannsynligvis laget for at batterier ikke skal kunne gjemmes unna i bokser rundt om i kjøretøyet.

Det var egentlig planlagt å få kretskortet bestilt fra fabrikk, men prototypen fungerte så bra at denne planen ikke ble gjennomført. Figur 20 viser kretskortet rett etter fortinning. På figur 21 er komponenter loddet på og kortet plassert i den ene halvdelen av boksen.



Figur 20: Kretskort rett etter etsing og fortinning



Figur 21: Ferdig kretskort, montert i boks med GPS-antenne, men uten mobilantenne montert

5 Software design

Det er i utgangspunktet tre moduler som kjører sin egen kode i tilknytning til telemetri-modulen. Telit-modulen har mulighet for å kjøre Python-kode og AT90CAN-mikrokontrolleren kjører sitt eget program som er skrevet i C. Pc'en tilknyttet internett må ha mulighet til å opprette en TCP/IP server, sende, motta og visualisere data.

5.1 Dataflyt

Slik systemet er tenkt å fungere vil mikrokontrolleren lytte etter CAN-meldinger, logge data til SD-kortet og sende en lang statusbeskjed til Telit-modulen. Telit-modulen vil så legge til GPS-posisjonen og eventuelt andre data om mobildekning, etc. og sende dette over GPRS-forbindelsen. Når dette blir mottatt på server-siden, vil serveren respondere med en styreordbeskjed som kan inneholde alt fra motorpådrag til "skru på hovedlys"-beskjeder. Dette blir så tatt imot av Telit-modulen som sorterer ut beskjeder til seg selv, og sender resten videre til mikrokontrolleren.

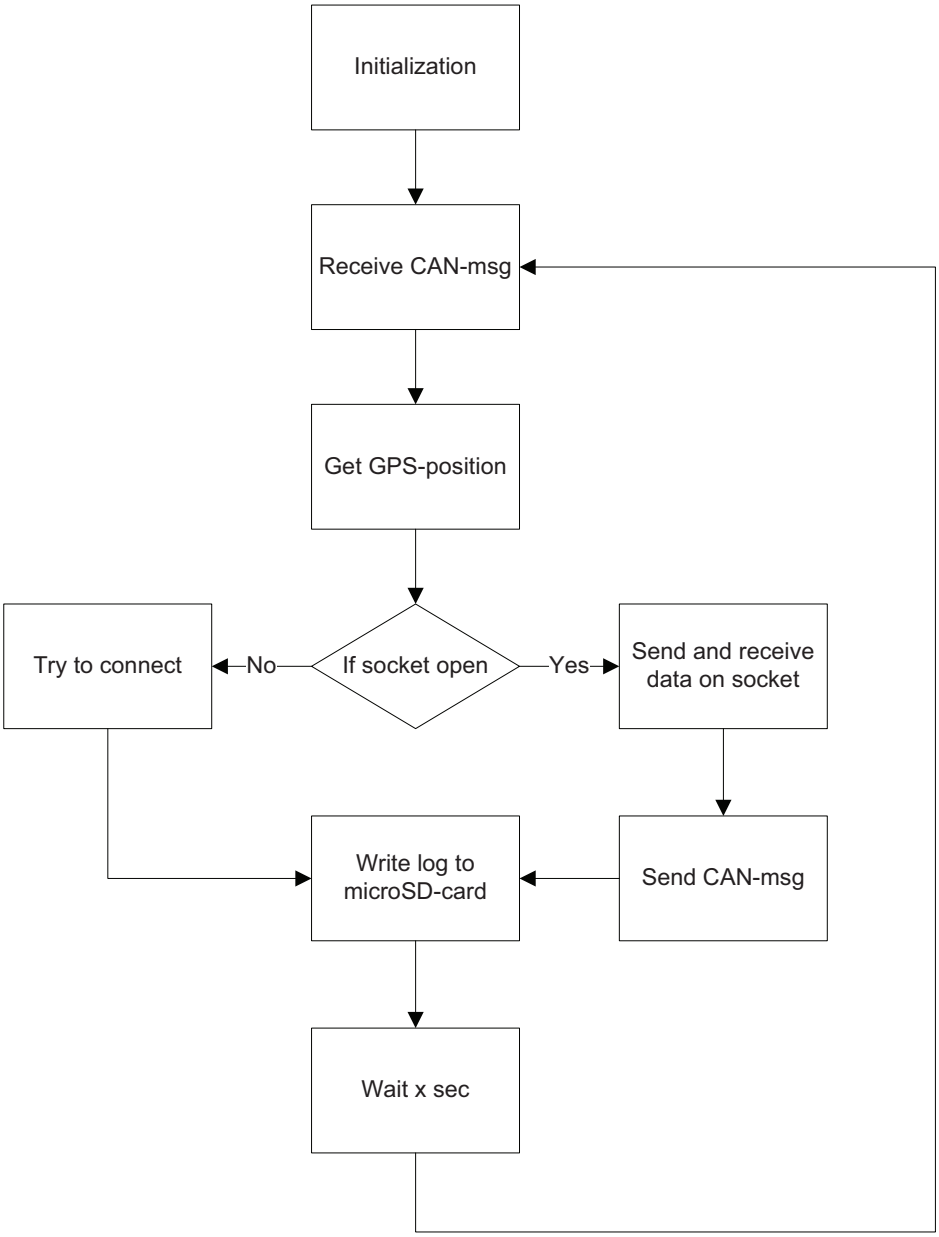
Mikrokontrolleren sender så ut de aktuelle CAN-meldingene, og eventuelt logger til SD-kortet hva som er mottatt og sendt. Dette gjentar seg med et fast tidsintervall på for eksempel ett sekund. Periode-tiden kan også justeres avhengig av om bilen er i bevegelse, brenselcellen er på, eller lignende aktivitetsparametere. Det er nødvendig at telemetri-modulen først sender data til serveren for så å motta data fra serveren. Dette er fordi serveren ikke vet om IP-addressen til Telit-modulen og det er lettere å administrere en TCP-socket på Telit-modulen som kun mottar data på bestemte tidspunkt. Siden IP-adresser kan endre seg innimellom er det planlagt å benytte et domenenavn istedet for IP-adressen.

5.2 AT90CAN C-kode

C-koden består en av en hovedløkke som sekvensielt mottar nye CAN-meldinger, sender data til Telit-modulen, mottar data fra Telit-modulen, sender CAN-meldinger og skriver til loggen. For å være sikker på å ikke gå glipp av CAN-meldinger eller UART-beskjeder fra Telit-modulen er det viktig med interrupt-baserte drivere for disse enhetene. Det er også lagt opp til en egen debug-tilkobling over den andre UART-modulen i AT90CAN-kontrolleren som ikke trenger å være interruptbasert.

For å utføre spenningsmålingene er ADC'en satt opp i single-conversion mode, og foretar 32 målinger som midles hver gang målemetoden kalles. Det er også lagt opp til pulsbredde-modulering av hver av fargene i statuslysdioden. Dette muliggjør mange forskjellige statusmeldinger fra samme lysdiode. Et flytdiagram over koden i mikrokontrolleren slik den er tenkt å fungere er vist i figur 22. Her er det lagt opp til at Atmel-kontrolleren

selv henter GPS-data fra Telit-modulen.



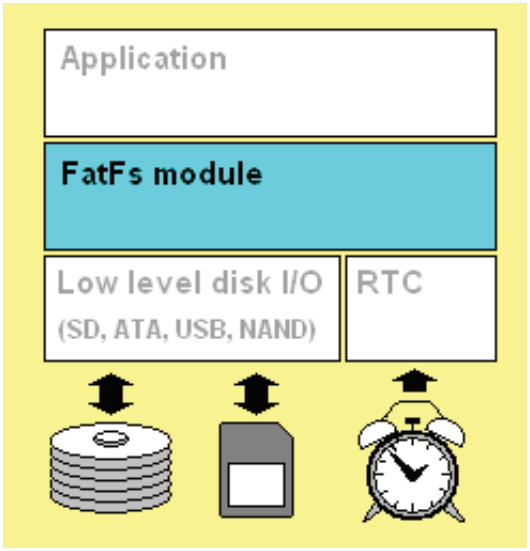
Figur 22: Enkelt flytdiagram for koden i mikrokontrolleren

5.2.1 MicroSD-kort

For å kommunisere med microSD-kortet skal SD-kortets SPI-modus benyttes. Et problem er at microSD-kort ikke nødvendigvis må støtte SPI-kommunikasjon. Det var derfor nødvendig

å undersøke litt rundt på nettet for å finne et passende minnekort. Et Kingmax-kort ble funnet med SPI-grensesnitt [23]. Et ferdig bibliotek for FAT-filsystemet skal brukes for å lese og skrive filer til minnekortet. Dette vil gjøre at minnekortet enkelt kan tas ut av telemetri-modulen og puttes i en vanlig pc, og deretter dukke opp som en ekstern disk.

Loggen som skal føres skal ligge i tekstfiler, der en ny loggfil blir opprettet hver gang mikrokontrolleren restartes. Tid og dato for loggen kan hentes direkte fra GPS-dataene. Som FAT-bibliotek ble et som heter FAT-Fs valgt [24], dette er en filsystemmodul som ligger mellom lavnivå-SPI-driveren og høynivå C-funksjoner som fopen, fwrite og fclose (figur 23).



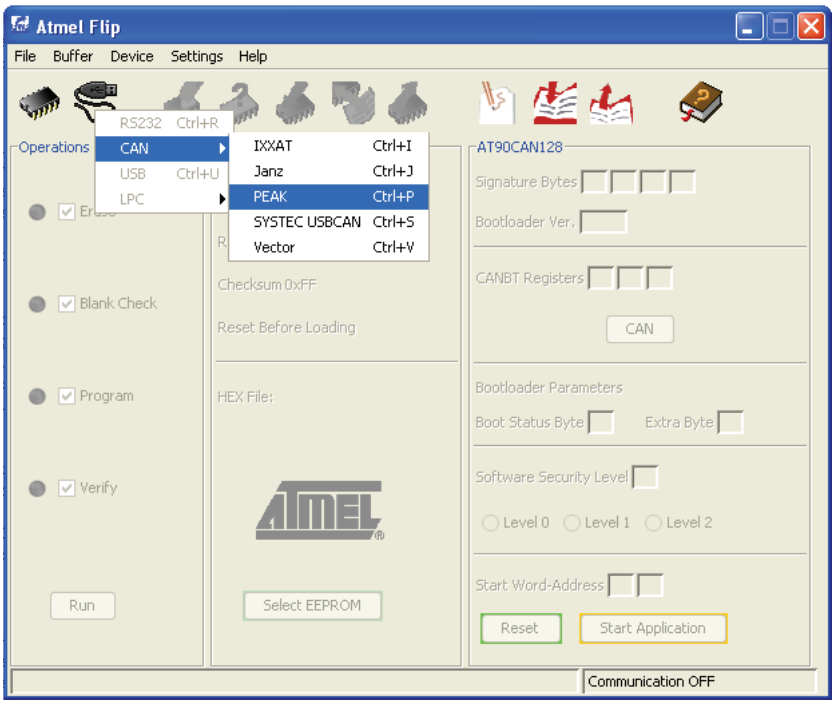
Figur 23: FAT-modulen

5.2.2 Bootloader

For å endre funksjonen til en mikrokontroller må ny kode lastes inn i flash-minnet. Under utvikling av koden gjøres dette ofte med en egen JTAG-enhet (Joint Test Action Group) [25]. Men når modulen er plassert i bilen vil dette være upraktisk. Ved å legge en såkalt bootloader inn i mikrokontrolleren er det mulig å få mikrokontrolleren til å programmere seg selv med kode som kan mottas fra for eksempel CAN-bussen. Dette gjør at hver modul i den ferdige bilen kan omprogrammeres med en pc som er koblet til CAN-bussen. Dette vil lette utviklingsarbeidet da ny kode kan testes uten å ta moduler ut av bilen for omprogrammering.

Atmel har allerede en Windows-applikasjon kalt FLIP (Flexible in system programmer) som støtter omprogrammering over for eksempel CAN-bus. Utfyllende kodeeksempler for bootloaderkoden er også tilgjengelig. En bootloader skrevet for IAR-kompilatoren og en skrevet for AVR-GCC (AVR-GNU Compiler Collection) er tilgjengelig via Atmel sin nettside,

AVR-GCC versjonen er benyttet. USB til CAN-bus adapteren som ble testet er levert av PEAK-systems.



Figur 24: Skjermbilde av Atmel sin FLIP bootloaderapplikasjon med PEAK sin USB-CAN adapter som valg

5.3 Python

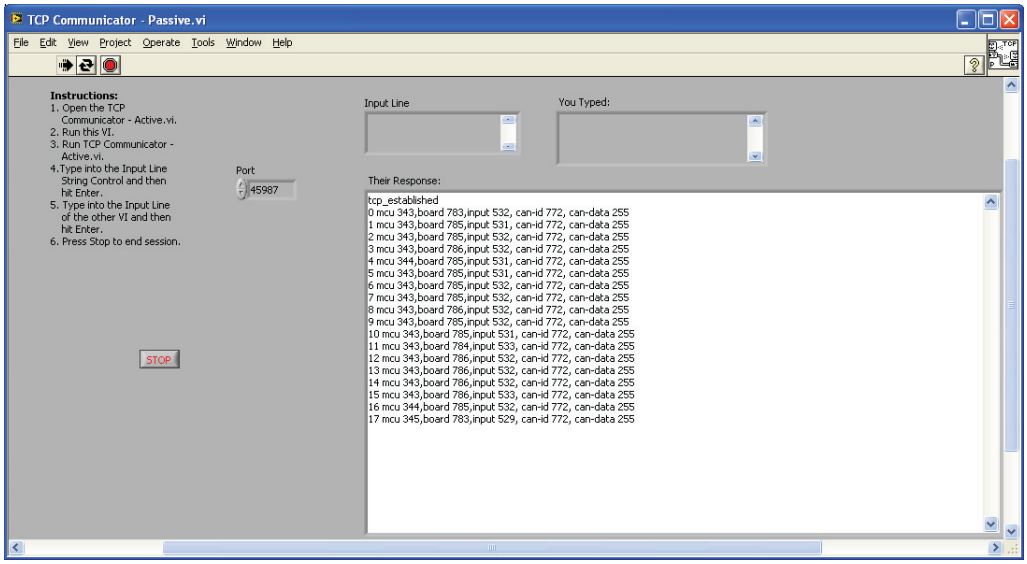
Telit-modulen skal i utgangspunktet kjøre sin egen Python-kode som både håndterer GPRS-forbindelsen, tar inn GPS-posisjon, samt mottar og sender data til serveren. Denne koden er lagt opp på samme måte som koden i mikrokontrolleren med en løkke som går evig. Men det er dessverre ikke noen interruptmuligheter på Telit-modulen, og hvilke Python-moduler som kan importeres er ganske begrenset.

Alternativet til Python er å la mikrokontrolleren ta seg av all konfigurasjon ved hjelp av AT-kommandoer. Dette kan bli en minst like god løsning og gjør koden for telemetri-modulen lettere å administrere. Kun ett program skrevet i C for mikrokontrolleren er nødvendig. Begge prinsippene er testet og resultater presentert i seksjon 6.

5.4 Labview

Som utgangspunkt for serverprogrammet ble National Instruments sitt grafiske programmeringsmiljø Labview valgt som plattform på grunn av enkelt oppsett av grafisk brukergrensesnitt. Labview sitt brukergrensesnitt er tilpasset prosesstyring, som er akkurat den rollen programmet skal fylle til slutt. En fullgod Labview-applikasjon skal utvikles utover våren når det er mer klart akkurat hva behovet er.

Det som er testet i forbindelse med denne oppgaven er mottak og sending av tekststrenger via en TCP/IP-forbindelse til Telit-modulen. Så lenge dette fungerer er det enkelt å splitte opp tekststrengen og visualisere dataene. Koden for TCP/IP-forbindelsen ble lånt fra et Labview-eksempel. Den ble utvidet til å kunne kjøre kontinuerlig og lytte etter en enhet som prøver å opprette en forbindelse (figur 25).



Figur 25: Passiv TCP/IP kommunikasjonsapplikasjon i Labview

6 Test og resultater

Etter at kretskortet ble laget og komponentene loddet på, ble det testet grundig. Det var viktig å verifisere at alle kretsløsninger fungerte som planlagt. Ved avvik fra forventet oppførsel kan i beste fall komponentverdier endres, eller små ledninger brukes for å rette opp feilkoblinger. I verste fall må et nytt kort lages. Etter at komponentverdier og koblinger har blitt sjekket opp mot kretsskjema kan kortet testes med spenning påsatt. En smart måte å starte på er å koble utgangen fra spenningsregulatorer bort fra resten av kretsen, og teste at den regulerte utgangen faktisk har riktig spenning.

6.1 DC-DC omformer

Under designfasen ble DC-DC omformerdesignet hentet fra databladet til Telit-modulen. Dette designet var nesten helt likt som forslaget i databladet til selve DC-DC-omformerkretsen [26], kun noen litt avvikende komponentverdier. Ettersom de ideelle komponentverdiene er avhengig av reelt strømtrekk og inngangsspenning som ennå ikke var fastlagt, var planen å finne ideelle komponentverdier etter at kretskortdesignet ble verifisert. Når kretskortet er ferdig produsert kan fortsatt komponentverdiene endres.

6.1.1 Test av DC-DC omformer

DC-DC omformeren ble koblet fra resten av kretsen. Forventet oppførsel ved påsatt spenning er at utgangen skal følge inngangen under 5 V, men holde seg på 5 V når spenningen stiger over dette. Ved første forsøk ble ved en feiltagelse spenning påsatt med feil polaritet, men det eneste som skjedde var at labstrømforsyningen begrenset strømmen som gikk gjennom zenerdioden over inngangen. Spenningen over inngangen gikk aldri under diodefallet til zenerdioden i lederetning på -0.6 V.

Ved riktig polaritet fungerte DC-DC omformeren som forventet. Støyamplituden på utgangen lå på 100 mV ved 0.1 A belastning og 25 V inngangsspenning (figur 30). Etter å ha regnet ut optimale verdier kan det vurderes om komponentverdier bør endres.

6.1.2 Valg av spole

Spolen er kretselementet som lagrer energien i DC-DC omformeren. Når bryteren i DC-DC kretsen er slått på vil strømmen i spolen øke. Etter at bryteren er slått av vil energien lagret i magnetfeltet som strømmen i spolen satt opp, føre til at det fortsatt vil gå en strøm i spolen. Ut fra (4) kan en se at en spole ikke kan ha momentane sprang i strøm. Energien

lagret i spolen er gitt ved:

$$E = \frac{1}{2}LI^2(\text{Joule}) \tag{3}$$

- L er spolens induktans [Henry]
- I er strømmen i spolen

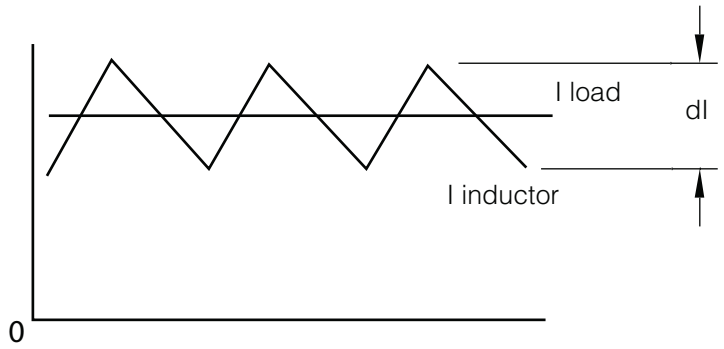
Spenningen over spolen er til enhver tid gitt ved:

$$V = L\frac{\mathrm{d}i}{\mathrm{d}t} \tag{4}$$

Ved høye svitsjefrekvenser kan en se på verdiene som følger:

- V er spenningen over spolen
- di er ripplestrømmen i spolen
- dt er tiden svitsjen i regulatoren er slått på

Dette er illustrert i figur 26.



Figur 26: Rippelstrøm i spolen

De parametere som er gitt:

	navn	verdi
Svitsjefrekvens:		250 kHz
Inngangspenning:	V_i	25 V
Maks rippelstrøm:	ΔI	1 A
Utgangsspenning:	V_o	5 V

Tabell 6: Tallverdier og navn på DC-DC omformerparametere

Her er rimelige verdier for inngangsspenning og rippelstrøm valgt. Rippelstrømmen er valgt utfra et ønske om rask transientrespons, ved å tillate høye strømmer i spolen kan regulatoren respondere raskere på lastendringer [27]. Arbeidssyklusen kan deretter regnes ut som følger [28]:

$$D = \frac{V_o}{V_i}$$
$$D = \frac{5}{25} = 0.2$$

Dette betyr at svitsjen vil være påslått 20% av tiden. Spenningen over spolen er videre gitt ved:

$$V_L = V_i - V_o$$

Den høyeste spenningen over spolen oppstår når svitsjen er påslått. Gitt tallverdier i tabell 6 blir $V_L = 20$ V. Spoleverdien kan nå regnes ut ved å snu på ligning 4:

$$L = V\frac{\mathrm{d}t}{\mathrm{d}i} \tag{5}$$

Hvor dt er gitt ved:

$$\mathrm{d}t = \frac{\text{arbeidssykel}}{\text{svitsjefrekvens}} \tag{6}$$

Innsatt tallverdier gir dette:

$$L = 20\ \mathrm{V} \cdot \frac{\frac{0{,}2}{250 \cdot 10^3\ \mathrm{kHz}}}{1\ \mathrm{A}} = 16\ \mu\mathrm{H} \tag{7}$$

Dermed var ikke den valgte verdien på 15 µH urimelig. Det kan allikevel være at 1 A som rippelstrøm er litt i høyeste laget. Høy rippelstrøm kan føre til høy spenningsrippel på utgangen [27].

6.1.3 Valg av utgangskondensator

Etter at spolen og dermed rippelstrømmen er bestemt kan parametere for utgangskondensatoren bestemmes. Det er tre viktige momenter å tenke på når det gjelder utgangskondensatoren:

- Kapasitans
- Ekvivalent serie motstand (ESR)

- Maksimal rippelstrøm

Kapasitansen er viktig for å holde utgangsspenningen stabil selv om strømmen i spolen varierer mye. En plutselig utkobling av lasten er en situasjon som fører til en stor endringer i strømmen på utgangen. Utgangskondensatoren må dermed klare å ta imot strøm fra spolen, og samtidig holde spenningen på utgangen under maksimalt tillatte spenning. Dette tilfellet er brukt for å regne ut mulig kapasitansverdi for utgangskondensatoren. Ved maksimal last vil en strøm I gå gjennom spolen, ved plutselig utkobling må all lagret energi i spolen bli overført til kondensatoren. Ut fra energibetraktninger på spole og kondensator får en:

$$\frac{1}{2}C_{\text{out}}V_{\text{etter}}^2 = \frac{1}{2}C_{\text{out}}V_{\text{før}}^2 + \frac{1}{2}LI_{\text{max}}^2 \tag{8}$$

hvor

$$\begin{aligned} I_{\text{max}} &= I_{\text{last}} + \frac{I_{\text{ripple}}}{2} \\ V_{\text{etter}} &= V_{\text{før}} + \Delta V \\ V_{\text{før}} &= V_{\text{out}} \end{aligned}$$

dette gir videre:

$$C_{\text{out}} = \frac{L(I_{\text{last}} + \frac{I_{\text{ripple}}}{2})^2}{(\Delta V + V_{\text{out}})^2 - V_{\text{out}}^2} \tag{9}$$

En grei verdi for rippelspenningen kan foreksempel være et ønske om 1% spenningsavvik som gir 1% av 5 V = 50 mV, innsatt tallverdier:

$$\begin{aligned} I_{\text{last}}&: \quad 2.5 \text{ A} \\ I_{\text{ripple}}&: \quad 1 \text{ A} \\ L&: \quad 15 \text{ µH} \\ V_{\text{out}}&: \quad 5 \text{ V} \\ \Delta V&: \quad 50 \text{ mV} \end{aligned}$$

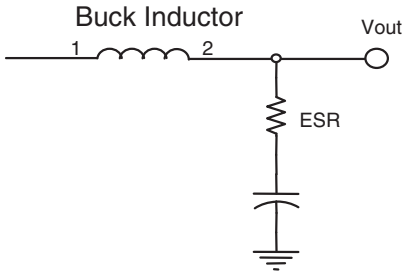
gir dette en kapasitans på C_{out} =269 µF, som en sikkerhetsmargin for kondensatorverdi og endrede egenskaper ved temperaturforandringer bør komponentverdien velges ca. 20% høyere [27]. Nærmeste standardverdi er da:

- C_{out} =269 µF

Dette resultatet er egentlig ikke helt reelt, for å kunne oppfylle ønsket om 50 mV spenningsripple på utgangen er det viktig at kondensatorens ekvivalente seriemotstand ikke er for stor. Enhver kondensator kan modelleres som en ideell kondensator med en motstand i serie (figur 27).

Spenningsrippelen som følge av denne motstanden blir:

$$V_{\text{ESR-rippel}} = \Delta I \cdot C_{\text{ESR}} \tag{10}$$



Figur 27: Ekvivalent seriemotstand

Hvor ΔI er rippelstrømmen i spolen. Denne rippelen vil komme i tillegg til rippel som følge av lastendring. Dette fører til at endelig kapasitans på kondensatoren bør være høyere enn ligning 9 tilsier. Ligning 10 gir ved innsetting:

$$C_{\text{ESR}} = \frac{V_{\text{ESR-rippel}}}{1\text{ A}} \tag{11}$$

C_{ESR} bør altså ligge under 50 mΩ. Rippelen blir nå summen: $V_{\text{last-ripple}} + V_{\text{ESR}}$ =100 mV. For å oppnå 50 mV total rippel bør både kapasitansen økes og seriemotstanden minkes. Ved å sette et krav på 25 mV rippel i både ligning 9 og 10 vil en kondensator på 560 µF med 10-20 mΩ ESR oppfylle kravet. Dette er høye krav til kondensatoren, ved å godta mer enn 1% rippel vil en billigere og fysisk mindre kondensator kunne benyttes.

6.1.4 Inngangskondensator

Inngangskondensatoren er viktig for å ikke lage for mye rippel på spenningsforsyningen. Verdien som velges er avhengig av impedansen til forsyningen. I dette tilfellet er det et batteri. I følge [27] er 10-20 µF vanlige verdier. Et viktig poeng her er at typen kondensator bør være elektrolytt eller keramisk, dette er fordi tantalum-kondensatorer har kortslutning som typisk feiltilstand, og det er uheldig å kortslutte inngangen.

6.1.5 DC-DC omformer konklusjon

Stor nok utgangskondensator med 50 mΩ ekvivalent seriemotstand var ikke mulig å oppdrive på det tidspunktet kretsen ble laget, så en variant med opp mot 1 Ω seriemotstand er benyttet. Dette er nok hovedårsaken til at målt rippel er så mye som 100 mV allerede ved 100 mA belastning. Det er mulig å få tak i kondensatorer med ned mot 10 mΩ ESR, dette hadde vært et bedre valg. Utgangen går uansett gjennom en lineærregulatorer som vil dempe rippelen før den når mikrokontroller og Telit-modul.

6.2 Linære spenningsomformere

De lineære spenningsforsyningene ble koblet til 5 V og utgangsspenningen ble målt. Alle tre fungerte som forventet og holdt riktig spenning på utgangen. De er ikke testet med annen belastning enn det som er montert på kortet.

6.3 Software testing

Koden er delt opp i moduler og hver modul er testet separat.

6.3.1 Hello world

For å teste om mikrokontrolleren fungerer ble et enkelt program skrevet for å skru lysdioden av og på. Dette ble så lastet opp ved hjelp av JTAG-tilkoblingen [25].

6.3.2 CAN-driver

En interruptbasert CAN-driver for AT90CAN er lånt fra et tidligere prosjekt. Siden AT90CAN ikke har noen loopback-funksjonalitet må CAN-driveren testes med en annen node koblet til CAN-bussen. Til dette formålet ble en USB-CAN adapter benyttet. Dette er samme adapter som fungerer som bootloader-adapter til Atmel sin bootloader-applikasjon [29]. Testingen gikk ut på å sende meldinger ut på CAN-bussen og motta meldinger som så sendes til debug-uart'en.

6.3.3 UART-driver

For å teste uart-drivern ble data sendt til mikrokontrolleren samtidig som kontrolleren er opptatt med noe annet. Alle dataene ble da lagret i ringbufferet i interrupt-rutinen, og kunne leses ut senere.

6.3.4 SPI og microSD-kort

Først ble kortet testet ved å sende noen SPI-kommandoer og se om responsen var riktig og som oppgitt i databladet, dette fungerte. Det ble også forsøkt å skrive noen sektorer og lese de samme sektorene for å se om lesing og skriving fungerte. Deretter ble et rammeverk for FAT-filsystemet implementert.

6.3.5 FAT-filsystem

Filsystemet ble testet ved å først lage et korrupt minnekort ved å skrive 0xAA til hele kortet. Deretter ble det formatert med FAT-fs sin mkfs-funksjon. En fil ble så opprettet, skrevet til og lukket. Kortet ble deretter lest i en vanlig pc. Minnekortet dukket opp som en ekstern disk og inneholdt filen som ble laget. Dette fungerte dermed akkurat som tiltenkt. Opprettet- og endret-dato på filen fungerte også som det skulle.

```
FATFS FileSystemObject;
FIL logFile;
f_mkfs(0,0,0);
f_mount(0, &FileSystemObject);
f_open(&logFile, filename, FA_READ | FA_WRITE);
f_write(&logFile,output_string,strlen(output_string),&byteswritten);
f_close(&logFile);
```

6.4 Testing av Telit-modul

For å verifisere at den valgte Telit-modulen fungerte som forventet, ble nødvendig funksjonalitet testet først på utviklingskortet. All funksjonalitet som har med GPRS og nettverkstilkobling kan testes ved å sende AT-kommandoer til modulen via serieporten. For å laste opp Python-kode ble et eget terminal-program fra Telit benyttet. Telit-modulen har innebygget funksjonalitet som gjør TCP/IP tilkoblinger enkle å administrere, når koblingen først er satt opp kan tekst som blir sendt og mottatt over serieporten automatisk bli sendt over TCP/IP-linken. GPS-funksjonalitet og Python-bibliotek for dette ble også testet.

6.4.1 AT-kommandoer

AT-kommandoer var opprinnelig kalt Hayes kommando-sett, etter at det ble funnet opp av Hayes Microcomputer Products i 1977 [30], AT står for attention. Kommandosettet har overlevd til den dag i dag, men det har fått en rekke modifikasjoner underveis. Det er et modifisert kommandosett som brukes på Telit-modulen. Det er lagt til nye standardkommandoer for GSM-telefoni. Eksempler fra GSM-AT-kommandosettet:

- `AT+CPIN=1234`

  Legger inn pin-kode

- `AT+COPS=?`

  Lister opp tilgjengelige nettverk

- `ATD <nummer>;`

  Ringer opp nummeret

Første test var å legge inn pinkoden, og prøve å ringe et nummer etter at modulen var registrert på nettverket (figur 28). Simkortet som er brukt er et vanlig kontantkortabonnement.

```
AT
OK
AT+CPIN?
+CPIN: READY

OK
AT+COPS?
+COPS: 0,0,"N 05"

OK
AT#MONI
#MONI: N 05 BSIC:06 RxQual:0 LAC:0410 Id:0FDE ARFCN:982 PWR:-85dbm TA:2

OK
ATD 98674209;
OK

BUSY
AT
OK
_
```

Figur 28: Test av noen AT-kommandoer

6.4.2 Easy gprs

For å koble til internett over GPRS har Telit laget et enkelt kommandogrensesnitt med egne AT-kommandoer. Dette grensesnittet kaller de "Easy GPRS" [6]. Det er i praksis kun tre AT-kommandoer som må sendes til Telit-modulen for at den skal bli koblet opp til internett over GPRS og få en IP-adresse. En fjerde kommando er så nødvendig for å sette opp en TCP-kobling til en ekstern server. Følgende kommandoer setter opp TCP/IP-stakken og oppretter en forbindelse over GPRS med www.google.com:

- `AT+CGDCONT = 1,"IP","internet.gprs","0.0.0.0",0,0`

  Setter opp en PDP-kontekst (Packet Data Protocol) med ID=1.

- `AT#SCFG=1,1,300,90,600,50`

  Setter opp en tilkoblingsmal med ID=1 tilknyttet PDP-kontekst #1. Denne malen bestemmer tillatte pakkestørrelser og tidsavbrudd for alle socket-tilkoblinger som opprettes gjennom PDP-kontekst #1.

- `AT#SGACT=1,1,"username","password"`

  Denne kommandoen aktiverer kontekst #1, og bruker username og password hvis det

er nødvendig. Etter dette punktet har modulen en unik IP-adresse på internett hvis ingen av kommandoene feiler.

- `AT#SD=1,0,80,"www.google.com",0,0`

  Bruker tilkoblingsmal #1 og setter opp en TCP forbindelse til www.google.com på port 80.

Nå vil alt som skrives i terminalen bli sendt over over TCP-forbindelsen til www.google.com. Foreksempel kan man laste ned søkesiden til google ved å sende:

```
GET / HTTP/1.1<cr><lf>
Host: www.google.com<cr><lf>
Connection: keep-alive<cr><lf>
<cr><lf>
```

der

```
<cr><lf>
```

betyr linjeskift (carriage return og linefeed).

Etter at HTTP-forespørselen (Hyper Text Transfer Protocol) er sendt vil terminalen bli fylt av HTML-kode (Hyper Text Markup Language) fra google. For å komme tilbake til kommando-grensesnittet, og få skrevet flere AT-kommandoer venter man minst ett sekund og sender:

```
+++
```

og venter i minst et sekund til. Ventetiden er lagt inn i tilfelle avbruddsekvensen forekommer i dataene som skal sendes. I tillegg til enkelt oppsett av TCP/UDP (User Datagram Protocol) forbindelser er det lagt inn en brukervennlig FTP-klient (File Transfer Protocol) og SMTP-klient (Simple Mail Transfer Protocol) for filoverføring og epost.

6.4.3 Python

Telit-modulen har mulighet for å kjøre Python-skript direkte, det betyr at den interne prosessoren i modulen kjører en Python-interpreter. Python-interpreteren støtter ikke siste versjon av Python, og har kun støtte for et lite subset av moduler (biblioteker i Python [31]. Under testing av Python-skript-mulighetene var det viktig å finne ut hvor god ytelse det var mulig å oppnå. Hvis ytelsen er akseptabel kan Telit-modulen stå for noen av oppgavene til telemetri-modulen alene. Når det gjelder størrelse på skript har modulen 2MB brukeraksesserbart flashminne [31].

Først ble et enkelt "hello world"-skript testet:

```
import SER
SER.set_speed('115200','8N1')
SER.send('Hello world!' + '\n\r')
```

Her importeres serieportmodulen, så settes hastighet, databit og stopbit. Hello world sendes ut på serieporten etterfulgt av linjeskift. For å kjøre Python-skriptet etter at det er lastet opp med terminalprogrammet, sender man AT-kommandoene

- `AT#ESCRIPT="<scriptname>`

  Aktiverer det skriptet som skal kjøres (det er mulig å legge inn flere skript med forskjellig filnavn).

- `AT#EXECSCR`

  Kjører det skriptet som ble aktivert sist. Når denne kommandoen er kjørt mister man AT-kommandogrensesnittet til skriptet har kjørt ferdig.

For å teste hastigheten ble en enkel løkke som kun inkrementerte en teller, kjørt mange ganger og kjøretiden ble målt i skriptet. Det viste seg at løkken kun kjørte gjennom ca. 1000 ganger i sekundet, noe som er veldig sakte hvis skriptet skal brukes til noe annet enn enkelt oppsett av modulen via AT-kommandoer.

6.4.4 Python debugging

Debugging er veldig praktisk å ha hvis man skal utvikle større applikasjoner. Problemet med GM862-GPS er at den ene serieporten er brukt opp av GPS-modulen. I Telit-moduler uten GPS brukes denne til å sende Python-interpreteren sine debugmeldinger. Dette problemet ble løst ved å omdefinere Python sin write-funksjon og sette sys.stderr lik den omdefinerte klassen [13].

```
class SerWriter:
    def write(self,s):
        SER.send(s+'\r')
sys.stdout = sys.stderr = SerWriter()
```

Nå vil alle debugmeldinger komme sammen med teksten fra SER.send- og print-funksjonene.

6.4.5 GPS

Den innebygde GPS-mottakeren i Telit-modulen baserer seg på en SiRFstar 3 GPS-brikke som har veldig god følsomhet og kort TTFF (Time To First Fix) [32].

GPS-modulen kan aksesseres på tre måter;

- NMEA-0183 data hentes rett ut fra den andre serieporten på Telit-modulen.
- Egne Telit AT-kommandoer.
- Python-modulen "GPS".

Det betyr at Telit-modulen enten kan hente GPS-data rett fra Python-skriptet, eller sende GPS-data ut til eksterne enheter via en ekstra seriell port eller AT-kommandoer direkte. Både AT-kommandoer og Python-modulen ble testet og fungerte veldig bra. Det var mulig å få stabil posisjonering innendørs opptil fire meter fra nærmeste vindu. TTFF innendørs er 3-5 minutter. Utendørs er ikke testet, men sannsynligvis enda raskere.

6.4.6 Problemer med Python

Løsningen med både Python-kode på Telit-modulen og C-kode på mikrokontrolleren ble etterhvert forkastet da Telit-modulen viste seg å være ustabil selv med enkle Python-kodesnutter. Modulen ble enten hengende uten å gi noen respons, eller den hadde restartet etter å ha stått på i noen timer selv med kun en enkel while-løkke. AT-kommandoer måtte istedet sendes fra mikrokontrolleren til Telit-modulen. Det er laget en egen C-modul for denne jobben, som tar seg av både sending av kommandoer, venting på mottak, og sjekk av den returnerte verdien.

Testingen foregikk ved at kommandoer ble sendt og resultatet sjekket opp mot det som er forventet oppførsel. Forventet oppførsel ble testet ved å sende AT-kommandoer manuelt. Et viktig aspekt ved AT-kommandoene er timingen, mikrokontrolleren må vente en liten stund før svaret kan leses ut. Denne venteperioden kan være alt fra noen millisekunder, til et par sekunder, og er individuell for hver AT-kommando. Venteperiodene måtte bestemmes ved prøving og feiling.

6.5 Antenneplassering

I utgangspunktet var det tenkt at både GPS-antennen og mobiltelefonantennen skulle være inne i boksen. Dette var ønsket for å beskytte begge antennene, og gjøre telemetri-modulen så kompakt som mulig. Ved å ha antennene internt slipper man en å være redd for løse

koaxial-antenneledninger som ofte er litt skjøre. Antenner montert på utsiden av boksen er utsatt for bøying og påkjenninger ved håndtering.

6.5.1 GPS-antenne

GPS-antennen ble modifisert med kortere ledning og limt fast på Telit-modulen, slik at den er rettet mot himmelen når boksen er montert på et horisontalt underlag. Denne plasseringen av GPS-antennen fungerte utmerket, og signalstyrken var like god som ved forsøk med en ekstern antenne (figur 21).

6.5.2 Mobiltelefonantenne

Mobiltelefonantennen ble forsøkt montert parallelt med og i 1.5 cm avstand fra kretskortet. Dette skapte dessverre problemer for Telit-modulen. Ved registrering på et mobilnett, som skjer rett etter oppstart av Telit-modulen, bruker modulen maksimal sende-effekt for å være sikker på å bli hørt av en basestasjon [11]. Ved montering av antennen rett over kretskortet gikk Telit-modulen i reset nesten hver gang den prøvde å registrere seg på nettet.

Problemet ble funnet å være spenningen som ved registrering sank ned til under 3.2 V som er nedre grense for Telit-modulen, se oscilloskopbilde 31. Kommer spenningen under 3.2 V går modulen i reset. Problemet ble forsøkt løst ved å montere en større kondensator på forsyningen, og øke spenningen fra DC-DC omformeren litt, slik at fallet over diode og lineærregulator ved større strømstyrker ikke skulle føre til spenningsfall. Dette bedret problemet litt, men det var fortsatt et sporadisk problem.

Siden kretsløsningen nå var overdimensjonert i forhold til databladet, ble det forsøkt å montere antennen lenger vekk fra kretskortet, og vinkelrett på, dette viste seg å være løsningen på problemet. Sannsynligvis var det ugunstig med et jordplan så nærme antennen, som førte til et større effektforbruk i RF-forsterkeren (Radio Frequency) enn normalt. Dette problemet kan ha vært årsaken til at Python oppførte seg ustabilt, men dette ble ikke undersøkt videre. Bilde 29 viser modulen med mobilantenne plassert på utsiden og USB-kontakt for serieportkommunikasjon.

6.6 Sluttprodukt

Modulens funksjonalitet ble testet ved å sende en enkel CAN-melding til modulen og observere at informasjonen kom fram til Labview-programmet på pc'en. På samme måte ble en melding sendt fra pc'en, og mottatt som CAN-melding på CAN-bussen. Ettersom resten av systemet i bilen ikke er ferdig designet, er protokollen for overføring av data ikke

Figur 29: Telemetri-modulen i lukket boks med mobilantenne og USB-kontakt

detaljspesifisert ennå, men det blir sannsynligvis en tekst-streng som blir sendt fram og tilbake. Dermed kan dataene leses ut direkte fra TCP/IP-forbindelsen, og mikrokontrolleren slipper å bruke tid på å konvertere tekststrengen fra GPS'en. Dette vil også gjøre protokollen lett å debugge ved at det kun er leselig ascii-kode som benyttes. Dette er samme prinsipp som ligger bak de vanligste protokollene på internett, HTTP og SMTP for eksempel.

7 Diskusjon

Etter at kretskortet var laget dukket det opp et par små problemer med både hardware og software. Det er ikke noen alvorlige feil som går ut over basis-funksjonaliteten, men det er ting som bør rettes opp hvis et nytt kretskort skal lages.

7.1 Hardware

I utgangspunktet fungerte det meste av hardwaren for telemetri-modulen ved første forsøk. Grundig planlegging av kretsløsningene, og tiden brukt på å lage et gjennomtenkt kretskortutlegg var vel anvendt tid. Allikevel ble det funnet noen problemer ved kretsløsningen som bør løses i en eventuell neste versjon.

7.1.1 Redesign av UART-løsningen

Rx og Tx UART-linjene til Telit-modulen og mikrokontrolleren gjør det mulig å kommunisere med Telit-modulen fra både USB-porten via FTDI-brikken [14], og fra mikrokontrolleren til Telit-modulen. Problemet er at Tx-linjen fra FTDI-brikken går sammen med Tx-linjen på mikrokontrolleren. Så USB-porten kan ikke brukes til å snakke med mikrokontrolleren. Det hadde vært praktisk med kun en USB-port for overføring av data fra minnekortet og samtidig debug-muligheter. En mulig løsning er å ha noen jumpere som kan flyttes for å bytte om Rx og Tx-signalene.

7.1.2 På og av-slåing av Telit-modul

Et uventet problem med Telit-modulen er måten den slås av og på. Det er en power-on pinne som skal holdes lav i minst et sekund, for å svitsje enheten enten på eller av, avhengig av tidligere tilstand. Problemet er at mikrokontrolleren ikke vet om modulen er på eller av. Dette ble løst i software ved å sende en AT-kommando, og anta at modulen er av hvis den ikke svarer. En bedre løsning kan være å sjekke power-monitor-signalet fra modulen som går høyt når den er slått på.

7.1.3 Feedback-banen i DC-DC omformerdesignet

Slik kretsutlegget er lagt opp nå går feedback-banen fra utgangen av DC-DC omformeren nesten rett under spolen. Dette er et av de mest støyende områdene på kretskortet. Dette ble også observert med et oscilloskop, selv om proben er over 3 cm unna spolen får man

tydelig innslag på skopet. Feedback-banen er bare noen millimeter unna. Inngangen på skopet er mye mer høyimpedant enn feedback-banen, men det er allikevel en fordel å holde feedback-banen så langt unna som mulig.

7.1.4 Klokkefrekvens

Det viste seg at en klokkefrekvens på 10 MHz ikke var kompatibel med CAN-bus hastigheter over 125 kbit/s. Dette ble løst ved å endre klokkekrystallet til 8 MHz. 8 MHz passer ikke like bra sammen med UART-hastigheter over 38.4 kbit/s [20], men 38.4 kbit/s holder til kommunikasjonen med Telit-modulen og debug-meldinger.

7.1.5 Ustabilitet

Av uforutsette hardware-problemer er det i hovedsak antenne-plasseringen som lagde mest problemer. Det tok litt tid å finne ut at det faktisk var selve plasseringen som gjorde at Telit-modulen krasjet ved oppstart.

7.2 Software

7.2.1 Python ustabilitet

Python-kode viste tegn til ustabilitet ved at modulen stadig krasjet etter noen timer. Dette ble testet ved å la en enkel løkke kjøre i noen timer og kun sende og motta AT-OK kommandoer. Etter noen timer så hadde Telit-modulen som oftest skrudd seg av eller restartet. Det samme ble testet fra mikrokontrolleren uten noen problemer. Dermed ble det konkludert med ustabilitet ved kjøring av Python-kode. Problemet oppstod ved forskjellig tidspunkt hver gang, det ble ikke funnet noen systematikk i det. Om dette gjelder alle Telit-moduler er lite sannsynlig, da det ikke var mulig å finne andre med lignende problemer på supportforum på nettet. Det kan være andre problemer enn selve koden som gjorde at modulen restartet, det ble ikke tid til å undersøke dette nærmere.

7.2.2 AT-kommandoer og timing-problemer

Et problem som oppstod ved kommunikasjon med Telit-modulen fra mikrokontrolleren var tiden det tok å vente på svar fra hver AT-kommando. Noen kommandoer tar opptil flere sekunder å utføre. For eksempel å opprette TCP-forbindelsen, eller bytte mellom en aktiv og suspendert TCP-socket. Dette fører til at en oppdateringshastighet på 1 Hz blir umulig å opprettholde hvis GPS-posisjonen må hentes med AT-kommandoer.

For å suspendere TCP-forbindelsen tar det minst 2 sekunder (se 6.4.2), så må GPS-posisjon hentes, og deretter kan forbindelsen aktiveres igjen. Realistisk oppdateringshastighet kan kanskje bli hvert 3. sekund, foreløpig er den på omtrent 7 sekunder. Løsningen på dette problemet kan være å hente GPS-posisjonen fra NMEA-Tx-linjen på Telit-modulen. Dermed kan TCP-socketen være aktiv hele tiden. Denne løsningen krever et nytt kretskortutlegg.

8 Konklusjon

I løpet av prosjektet har det blitt utviklet en telemetri-modul for Shell Eco-Marathon-kjøretøyet. Modulens delsystemer har blitt testet hver for seg, og modulens helhetlige funksjon er også testet og funnet å fungere som tiltenkt. Telemetri-modulen er i stand til å gjennomføre to-veis kommunikasjon med andre elektronikkmoduler i bilen og en internettilknyttet pc.

Modulen holder en TCP/IP-forbindelse over GRPS åpen med en Labview-applikasjon på en vilkårlig pc tilkoblet internett. Kretskortet som er utviklet og laget har oppført seg veldig stabilt. Det har ikke bydd på problemer etter at årsaken til ustabilitet, på grunn av antenneplasseringen, ble funnet og løst. For å kunne bruke den i en reell applikasjon må applikasjonskode skrives på både server- og klient-side. Alle drivere for GPRS-modul, microSD-kort, FAT-filsystem og CAN-bus er testet og er lett å ta i bruk. Et enkelt program i Labview som tar imot tekststrenger over en TCP/IP-link er benyttet for testing av GPRS-linken.

Telemetri-modulen er første modul i et helhetlig elektronikk-system for den kommende eco-marathon bilen, prinsipper for hele systemet er utviklet, og kommunikasjonsbuss med tilhørende fysisk sammenkobling mellom moduler er fastlagt. En bootloader for en Atmel AT90CAN-mikrokontroller, som kan laste opp kode over CAN-bussen, er testet og skal benyttes i det endelige systemet.

Oppgaven har omfattet et veldig bredt spekter av utfordringer, fra elektronikk og kretskort-design, til softwareutvikling i C, Python og Labview. Arbeidet med systemet har gitt innblikk i TCP/IP-kommunikasjon og GPRS-dataoverføring. En av de største overraskelsene var mengden tid det tar å lage et fysisk produkt. Alt fra å finne tilgjengelige elektronikkomponenter, til uforutsette overraskelser med ustabil hardware har vært tidkrevende. Dette har gjort at tiden ble for knapp til utvikling av ferdig software for telemetri-modulen.

Foreløpig er software skrevet med "proof of concept" som målsetting. Forbedringer som må utføres til våren er visualisering av data i Labview, og prøve å øke oppdateringshastigheten. Telemetri-modulen er ment som en del av instrumenteringen til neste års eco-marathon, dermed har en del av denne oppgaven innebært konseptutvikling for systemet som skal implementeres i bilen senere. Dette har bydd på tverrfaglige utfordringer, og samarbeid med team-medlemmer fra andre sivilingeniør-linjer ved NTNU.

Referanser

[1] R. Langley, GPS world **6**, 54 (1995).

[2] J. M. H. Bakken, Styresystem for fremdrift av shell-ecomarathon-kjøretøy, Master's thesis, NTNU, 2009.

[3] J. Wilson, Technology , 1 (2004).

[4] A. Aguiar and J. Klaue, Bi-directional WLAN channel measurements in different mobility scenarios, in *Proceedings of the Vehicular Technology Conference (VTC Spring)*, Citeseer, 2004.

[5] R. Kalden *et al.*, IEEE Personal Communications **7**, 8 (2000).

[6] Telit, *Easy GPRS User Guide*, 2009, `http://www.telit.com/`.

[7] D. Paret, *Multiplexed Networks for Embedded Systems: CAN, LIN, FlexRay, Safe-by-Wire...* (John Wiley and Sons Ltd, 2007).

[8] Flexray communication bus, 2009, `ftp://ftp.ni.com/pub/devzone/pdf/tut_3352.pdf`.

[9] A. Schedl, Goals and Architecture of FlexRay at BMW, in *slides presented at the Vector FlexRay Symposium*, 2007.

[10] Sparkfun, 2009, `http://www.sparkfun.com`.

[11] Telit, *GM862-GPS Datasheet*, 2009, `http://www.telit.com/`.

[12] Telit, *AT Commands Reference Guide*, 2009, `http://www.telit.com/`.

[13] G. Van Rossum and F. Drake Jr, *Python reference manual* (iUniverse, 2000).

[14] FTDI Ltd, *FT232R Datasheet*, 2009, `http://www.ftdichip.com/`.

[15] C. Strangio, CAMI Research Inc., Lexington, Massachusetts **2005** (2003), `http://www.applied-imagination.com/aidocs/RS232_standard.html`.

[16] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, Proceedings of the IEEE **91**, 489 (2003).

[17] SD Card Association, *SD Specifications*, 2006, `http://www.sdcard.org/`.

[18] W. Heybruck, H. StorageTechnologies, and N. Charlotte, Hitachi White Paper, Hitachi Global Storage Technologies (2005).

[19] D. Kalinsky and R. Kalinsky, Embedded Systems Programming , 55 (2002).

[20] Atmel, *AT90CAN Datasheet*, 2008, `http://www.atmel.com/`.

[21] Microchip, Mcp2551 transceiver, 2004.

[22] T. Instruments, Sn65hvd230 3.3v can-transceiver.

[23] Kingmax, *microSD Datasheet*, 2007, `http://www.kingmax.com`.

[24] Fat file system module, 2009, `http://elm-chan.org/fsw/ff/00index_e.html`.

[25] Atmel, *AVR JTAG ICE User Guide*, 2006, `http://www.atmel.com/dyn/resources/prod_documents/doc2562.pdf`.

[26] ST Semiconductor, *L5973 Datasheet*, 2008, `http://www.st.com/`.

[27] D. Schelle and J. Castorena, Power Electronics Technology **32**, 46 (2006).

[28] C. industries, *Inductor Selection for Switching Regulators*, 2008, `http://www.cooperbussmann.com/`.

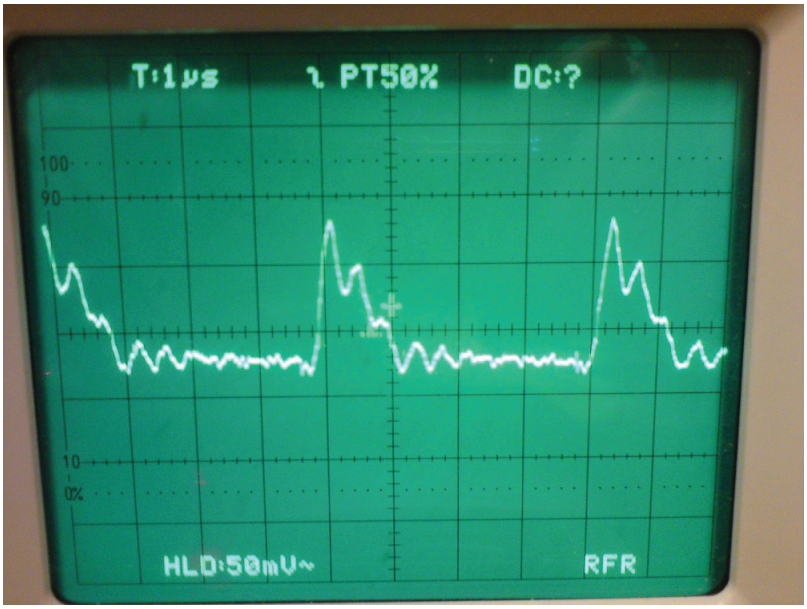
[29] Atmel, *Flip3 Bootloader Datasheet*, 2009, `http://www.atmel.com/`.

[30] S. Hazari, TechTrends **35**, 25 (1990).

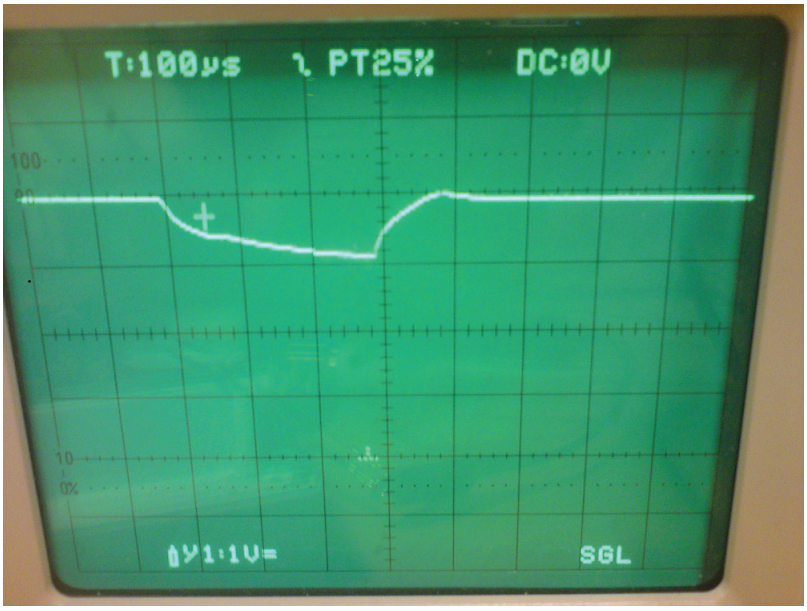
[31] Telit, *Easy Script Python*, 2009, `http://www.telit.com/`.

[32] SiRF Technology, Inc., *SiRFstarIII GPS Solution*, 2008, `http://www.sirf.com/products/GSC3LPProductInsert.pdf`.

A Oscilloskopmålinger



Figur 30: Ripple på utgangen av DC-DC konverter ved 100 mA laststrøm



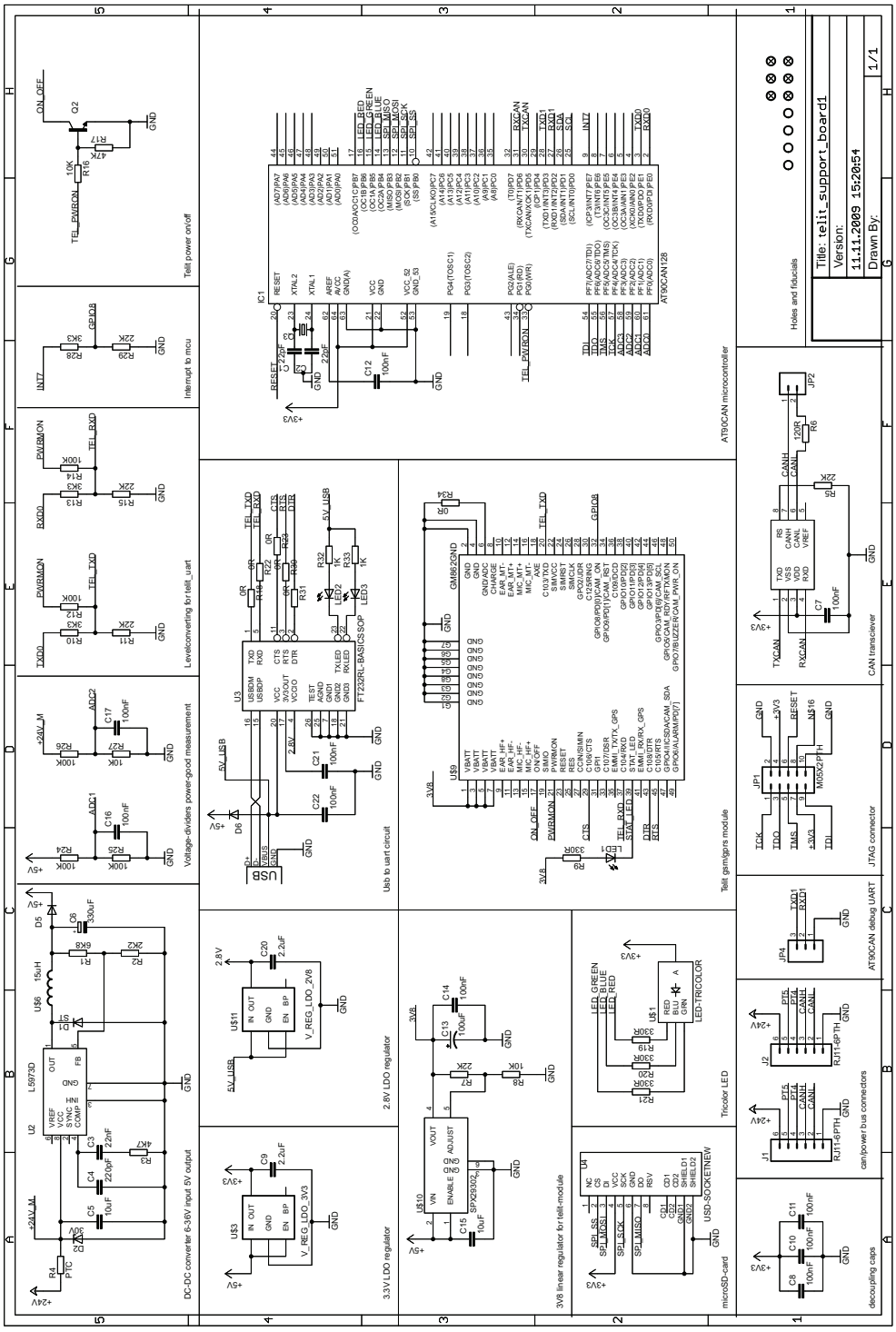
Figur 31: Spenningsdipp som førte til reset av Telit-modulen

B Innhold på CD

Den vedlagte cd'en inneholder aktuelle artikler, datablader, skjemategninger, kretskortdesign og software som er benyttet i løpet av prosjektoppgaven.

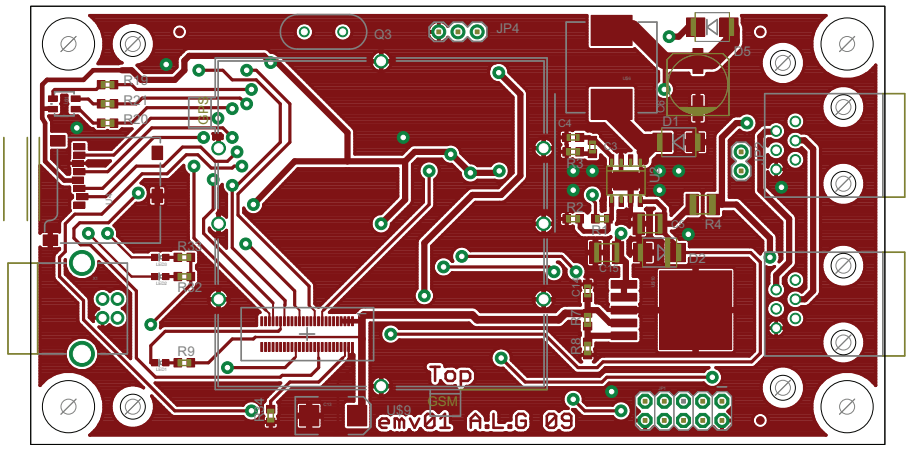
- Telemetrymodule
  - Articles
    - CAN
    - FAT16_32_introduction
    - FlexRay
    - GPS
    - Power
    - SPI_I2C
    - Wireless
  - Datasheets
    - Atmel_mcu
    - Enclosure
    - CAN-bus
    - GPS
    - JTAG
    - microSD
    - Power
    - Telit
    - USB_UART
  - Hardware
    - Libraries
    - telemetry_board_v01.sch
    - telemetry_board_v01.brd
    - gerberfiles_production_ready.zip
  - Rapport
  - Software
    - Labview
    - Atmel

C Komplett skjemategning

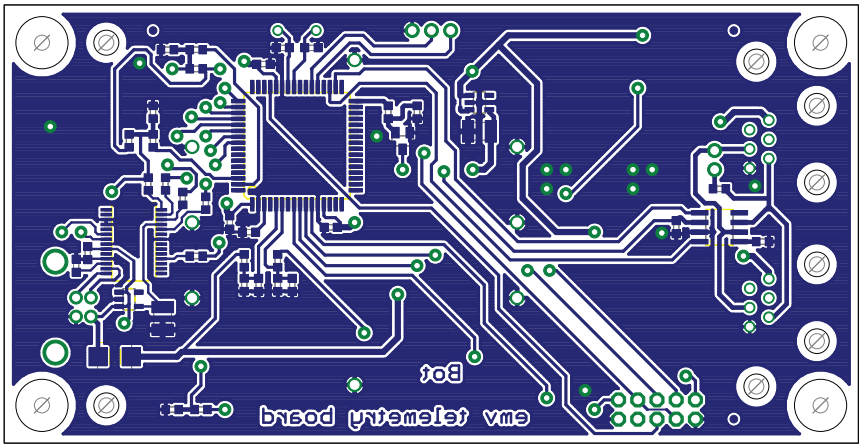


Figur 32: Komplett skjemategning.

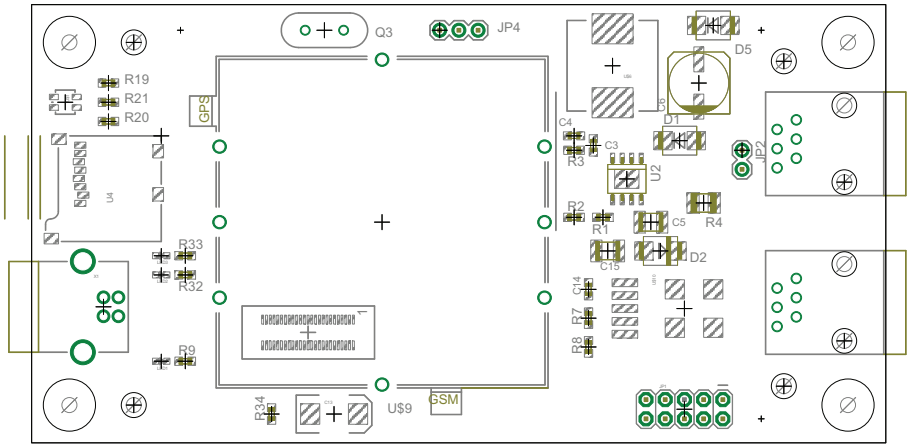
D Kretsutlegg 1:1



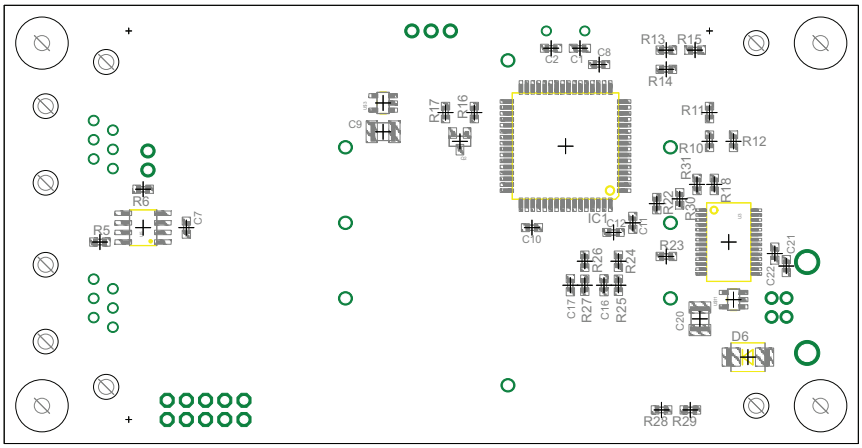
Figur 33: Oversiden av kretsutlegget.



Figur 34: Undersiden av kretsutlegget, speilvendt.



Figur 35: Komponentplassering på oversiden.



Figur 36: Komponentplassering på undersiden.