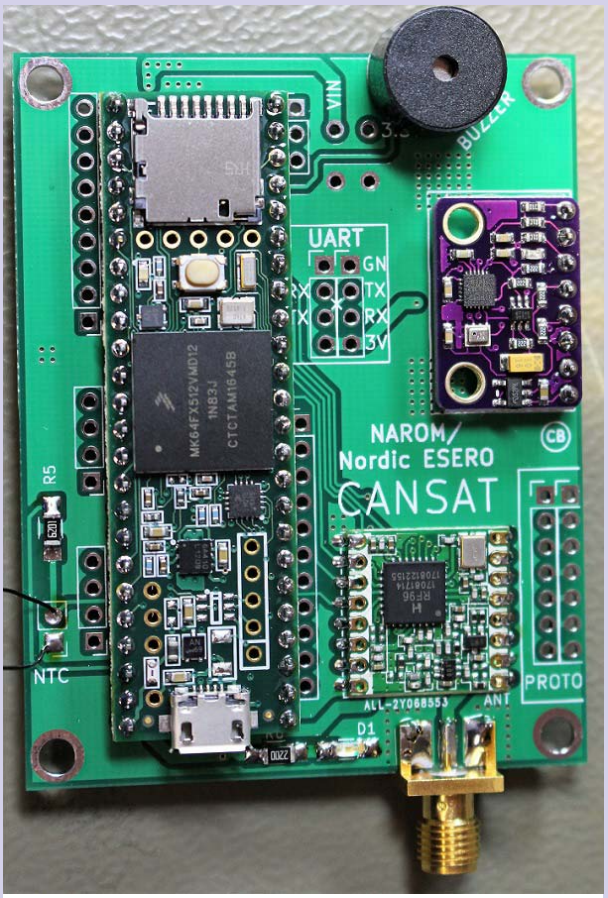
Prøvetrykk 5.3

Christoffer Stausland
Bente Jensen
Nils Kr. Rossing

CanSat - Teensy 3.5
En veiledning



NTNU
Trondheim

Institutt for fysikk

Skolelaboratoriet
for matematikk, naturfag og teknologi

November 2018

CanSat – Teensy 3.5
En veiledning

CanSat – Teensy 3.5, En veiledning

Trondheim 2018

Bidragsytere:
Christoffer Stausland, NAROM, (christoffer@narom.no) utviklet byggesettet med programvare og biblioteker
Bente Jensen, NAROM, (bente@narom.no) har laget nettsidene og beskrevet byggesettet
Nils Kr. Rossing, *(nils.rossing@ntnu.no)* Skolelaboratoriet ved NTNU

Layout og redigering: Nils Kr. Rossing, Skolelaboratoriet ved NTNU

Bilder: Bente Jensen, NAROM
Forsidebilde: Teency 3.5 (Foto, Nils Kr. Rossing)
Baksidebilde: Teency 3.5 montert i brakett (Foto, Nils Kr. Rossing)

Faglige spørsmål rettes til:
Skolelaboratoriet for matematikk naturfag og teknologi, NTNU
v/Nils Kr. Rossing, 73 55 11 91
nils.rossing@ntnu.no

Realfagbygget, Høgskoleringen 5
7491 Trondheim

Skolelaboratoriet
Telefon: 73 55 11 43
http://www.ntnu.no/skolelab/

Prøvetrykk 5.3, Rev 5.32 – 13.11.18

Siste utgave av boka kan lastes ned fra:
http://www.ntnu.no/skolelab/sl-bla-bokserie

CanSat – Teensy 3.5
En veiledning

Christoffer Stausland, Bente Jensen, NAROM
Nils Kr. Rossing, Skolelaboratoriet NTNU

Skolelaboratoriet for matematikk, naturfag og teknologi, NTNU

Forord

Hensikten med dette heftet er å samle erfaringer fra utprøving av CanSat-kittet versjon 7.0 utviklet ved NAROM av Christoffer Stausland og Bente Jensen vinteren og våren 2017/18 og som bl.a. brukes under kurset i Trondheim høsten 2018. Heftet er oppsummering primært av mitt studie av det nye settet sommeren 2018 og som danner grunnlaget for mine bidrag under kursene i Trondheim høsten 2018.

At både Christoffer og Bente står som forfattere er naturlig siden det er det teamet ved NAROM som har utviklet settet. Mitt bidrag har vært å sette meg inn i virkemåte og bruk og ev. kommentere det som er gjort. Og om mulig stille mer eller mindre intelligente spørsmål under veis.

Det er forøvrig ikke meningen at heftet skal erstatte den offisiell CanSat håndboka som man finner her: https://www.narom.no/undervisningsressurser/the-cansat-book/v2018/, men i beste fall være et supplement.

Dersom det er deler av dette heftet som inneholder feil, så er det mitt fulle ansvar.

Nils Kr. Rossing
Skolelaboratoriet ved NTNU
November 2018

Innhold

Revisjonsrapport

Revisjon 5.0	*15.07.18 - Første utgave av rev. 5.0 juli 2018 er nesten helt nyskrevet, men det er hentet inn noen deler fra rev. 4.2 som var basert på Arduino UNO og ver. 4.0 av CanSat-kortet*
Revisjon 5.1	*09.07.18 - Inkludert GPS og behandling av data hentet fra Rev. 4.2. Kapittelet om GPS er oppdatert mht. Teensy, mens kapittelet om behandling av data er ren kopi fra 4.2.*
Revisjon 5.2	*17.09.18 Rettet en del mindre feil. Rettet opp tilkobling til lader og batteritilkobling til CanSat-kortet.*

1 Innledning

Heftet er en samling og systematisering av grunnleggende erfaringer gjort under arbeidet med å prøve ut grunnmodulen av CanSat byggesettet versjon 7.0 utviklet av Christoffer Sausland og Bente Jensen ved NAROM. Byggesettet er utviklet og lagt opp ved NAROM, Andøya for bruk på lærerkurs og til undervisningsformål i ungdomsskolen og videregående skole.

Heftet gjennomgår hårdvaren for den nye versjonen av CanSat-kortet i tillegg til de sensorene som inngår på kortet. Heftet inneholder en oppsummering av funksjoner som spesielt er knyttet til Teensy 3.5, GY91 og RFM96.

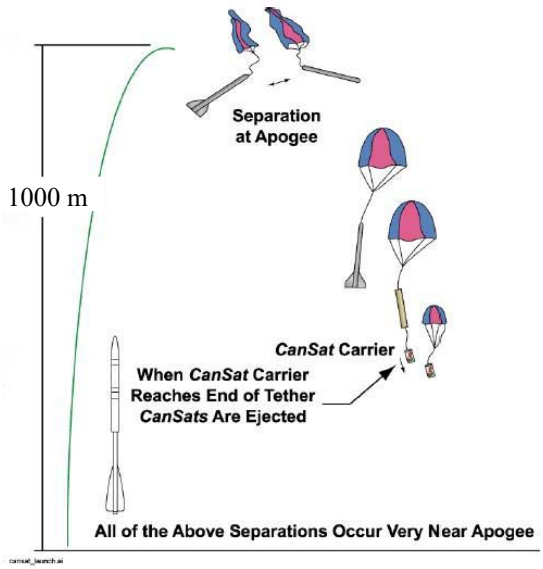
1.1 Grunnmodulen – CanSat (ver. 7.0)

Fra og med 2018 leveres en helt versjon av CanSat kittet (ver. 7.0) bygd på Teensy 3.5 med en ARM-prosessor og SD-kort leser/skriver. Slik byggesettet leveres inneholder det:

1. Byggesettet:
   - 2 stk. Teensy 3.5 (Microkontroller med SD-kort)
   - 2 stk. RFM96 (Tranceiver 433 MHz, 100 mW)
   - 1 stk. GY-91 (3-akse akselerometer, 3-akse gyrometer, 3-akse magnetometer, trykk- og temp.)
   - 1 stk. NTCLE300E3103SB (10 kΩ)
   - 1 stk. AT1224TWT (Buzzer, 5V)
   - 1 stk. BD33KA5FP (Spenningsregulator) (mottakeren trenger ikke regulator)
   - 2 stk. Lysdiode, 1206, SMD
   - 2 stk. CanSat-kort
   - 2 stk. 30 cm koaksialkabel m/SMA kontakt (antenne)
   - 2 stk. 20 cm jumpere for testing, hann
   - 1 stk. 15 cm ledning for batteritilkobling, hann
   - 2 stk. 1µF, SMD
   - 2 stk. 10 µF, SMD
   - 6 stk. 4,7 kΩ, 1206, SMD
   - 1 stk. 470 Ω, 1206, SMD (Seriemotstand lysdiode)
   - 1 stk. 10 kΩ, 1206, SMD (Seriemotstand NTC)
   - 1 stk. lader for Lipo batteri m/kabel og tilkobling til USB mikro
   - 1 stk. Lipo batteri 3,7 V, 1000 mAh
2. Utstyr for programmering
   - USB-kabel (USB A —> USM mikro)
   - Programvare fra nettet, Arduino m/Teensyuino
3. Fallskjerm
   - Fallskjerm
   - Øyeskrue m/mutter (M5)
4. Tilleggsutstyr
   - Brakett for montering (lånes eller lages)
   - Brusboks 33 ml

1.2 Konsept og spesifikasjoner

Ved hjelp av settet skal det bygges en liten sonde som skal skytes ut fra en rakett eller slippes fra ballong ev. drone, fra 300 – 1000 meters høyde. Sonden skal være på størrelse med en Cola-boks og ha en maksimal vekt på 350 g. Sonden utstyres med en sensormodul (IMU) som leser av sensorer og overfører måledata til en radiosender som telemetrerer dataene til en mottaker på bakken (bakkestasjon). Sendingen foregår i ISM-båndet dvs. omkring 434 MHz (ca. 70 cm bølgelengde). Idet sonden skytes ut/slippes, faller sonden kontrollert mot bakken i fallskjerm. Sensorer samler inn måledata under fallet som kontinuerlig overføres til bakkestasjonen.



Minimumsspesifikasjoner

Det stilles følgende krav til sonden:

1. Sonden skal minimum måle trykk og temperatur hvert 3. sekund.
2. Sonden skal overføre data til bakkestasjonen under fallet.
3. Sonden skal bygges slik at den får plass i en 330 ml standard Cola-boks, eller slik at ingen deler av sonden stikker ut over omfanget til en slik boks.
4. Boksen skal veie mindre enn hva en full Cola-boks veier, dvs. mindre enn 350 g.
5. Sonde skal drives med batteri eller solcellepaneler.
6. Antennen skal være fleksibel og ikke stikke ut mer enn ca. 10 cm når den er stuvet sammen ved oppskyting.
7. En fallskjerm skal være forsvarlig festet til den ene enden av boksen/elektronikken.

Krav til bakkestasjon

Bakkestasjonen skal...

1. ... kunne motta på senderfrekvensen til senderen i sonden (ca. 434 MHz)...
2. ... og kan om nødvendig, suppleres med en rettningsantenne som kan følge sonden i fallet.
3. ... motta alle data fra sonden under fallet og lagre i PCen for senere analyse og presentasjon

Krav til bærerakett

Bæreraketten skal...

1. ... løfte sonden opp til toppunktet for ferden hvor en eller flere (to) sonder skytes ut
2. ... en forsinket ladning inne i raketten skyter ut sondene slik at den kan falle fritt mot bakken
3. ... minimum løfte sondene til ca. 800 meter

Ballongslipp

Som et alternativ til oppskyting kan en slippe sonden fra en værballong. Primært fra en ballong som er festet til en snor til bakken. En slik løsning vil være billigere og lar seg lettere gjennomføre på egen skole enn en rakettoppskyting. Ulempen er at slippet blir mer avhengig av vind- og værforholdene og vil sannsynligvis måtte skje fra noe lavere høyde.

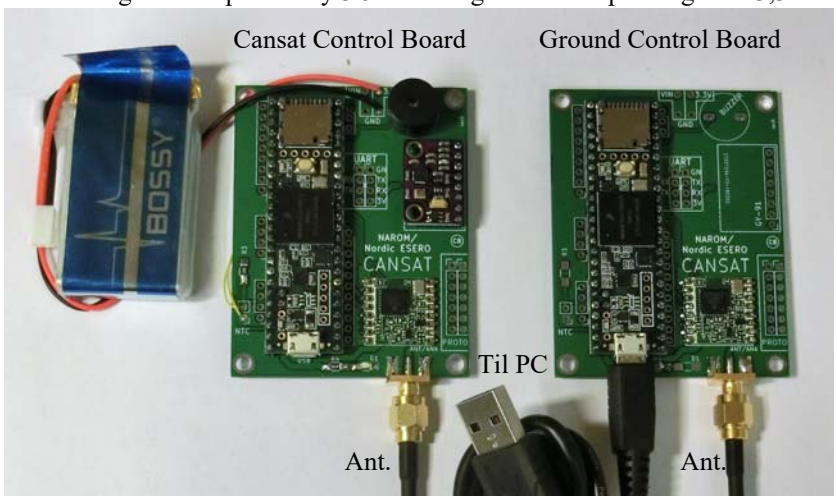
NB! Både oppskyting og slipp fra ballong må avtales med de lokale myndigheter, ev. nærliggende flyplasser.

Slipp fra drone

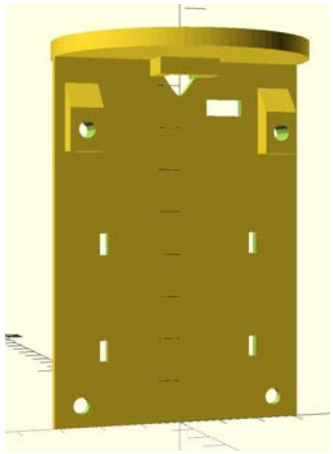
Dette er også et godt alternativ, men krever personell med godkjent løyve til å fly droner. Dessuten skal nærmeste flyplass varsles om den forestående aktiviteten i luftrommet.

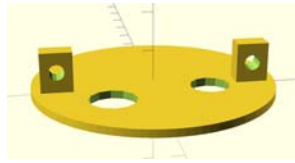
1.3 Montering av CanSat'en

Den ferdige CanSat'en består av to nesten identiske kort. Ett kort som skal monteres i brusboksen (Can), (CanSat Control Board – CCB) som er utstyret med sensorer og buzzer, og ett kort som skal kobles opp mot PC'en som et bakkekontrollkort (Ground Control Board – GCB). GCB er ikke utstyrt med sensorer eller buzzer og er derfor noe enklere. GCB trenger heller ikke spenningsregulator da regulatoren på Teensy 3.5 kortet regulerer ned spenningen til 3,3 V..

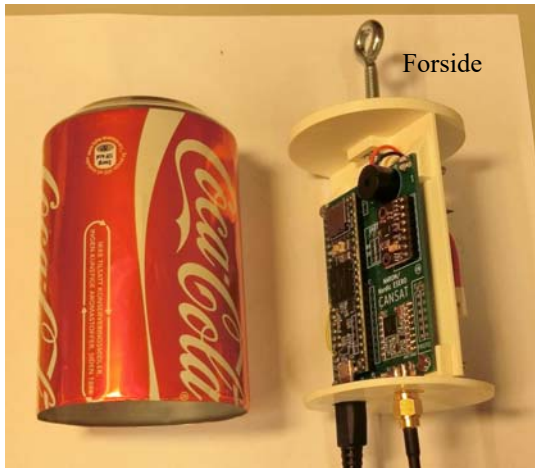


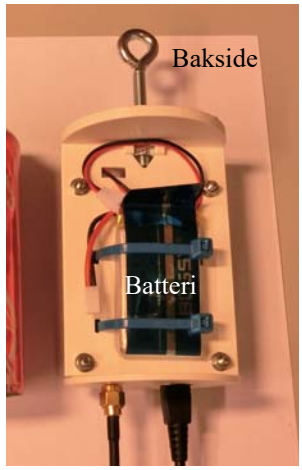
CanSat-kortet som skal monteres i brusboksen må først monteres på en passende brakett som enten kan være 3D-printet i PLA eller ABS, eller laget i aluminium eller stål. En 3D-printet utgave er lett å lage og å kopiere, men er ikke så robust som en festeanordning i metall. Her har vi valgt å lage en testutgave i PLA i to deler for å kunne bruke en enkel 3D-printer uten eget "stillas-materiale".





Figuren under viser den ferdig monterte CanSat'en festet til braketten, men uten brusboksen som skal omhylle CanSat med brakett. Antennen stikker ut gjennom et hull bunnplata, mens en USB-mikro-kabel kan kobles til gjennom et annet hulle.





Bildet over viser den monterte CanSat'en sett fra for- og baksiden.

2 Montering og installasjon

Kapittelet gir en oversikt over komponentene og en veiledning for montering av byggesettet.

2.1 Komponentliste

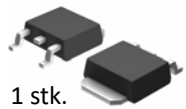
Komponentliste for NAROM 2018 CanSat kit:

Komponent	Antall	Akronym	Kommentarer
Buzzer Magnetisk, 3 - 7 V, 2.4 kHz, 87dB @ 5V	1	Buzzer	AT-1224-TWT-5V-2-R Digikey: https://www.digikey.no/product-detail/en/pui-audio-inc/AT-1224-TWT-5V-2-R/668-1470-ND Datablad:
Resistor 4.7kΩ 1206 SMD	6	R1, R2, R3, R5, R6	2xR1, 2xR2, R5, R6 – 4,7kΩ Foreslår R3 - 10kΩ
Resistor 4.7kΩ 1206 SMD	1	R3	Foreslår R3 - 10kΩ
Resistor 1206 470 Ω motstand for LED	1 (2)	R4	Typisk 220 - 470 Ω
Radiomodul RFM96	2		433 MHz, 100mW Datablad: http://www.hoperf.com/upload/rf/RFM95_96_97_98W.pdf
Sensor GY-91	1		
Processor board Teensy 3.5	2	GY91	
NTC-sensor, 10 kΩ NTCLE201E3103SB	1	NTC	Digikey: https://www.digikey.no/products/en?keywords=NTCLE300E3103SB Datablad: https://www.vishay.com/docs/29051/ntclesb.pdf
Spenningsregulator IC REG 3.3V - BD33KA5FP Pakke TO252-3 (SMD)	1	V1	Alternativ spenningsregulator for V_in høyere enn 5,5V (f.eks. et 9V batteri) Regulatoren synes å måtte ha en V_{in} på min. 4,8 V dersom den skal levere V_o = 3,3 V Datablad: http://rohmfs.rohm.com/en/products/databook/datasheet/ic/power/linear_regulator/bdxxka5-e.pdf
1206 10µF kondensator	2	C1, C3	
1206 1µF kondensator	2	C2, C4	
Stiftlist 36-pin	2 (3)		Digikey: https://www.digikey.no/product-detail/en/sullins-connector-solutions/PREC040SAAN-RC/S1012EC-40-ND/2774814
Lysdiode 1206 LED	1 (2)	D1	
SMA 50 Ohm kontakt	2		Digikey: https://www.digikey.no/product-detail/en/cinch-connectivity-solutions-johnson/142-0701-801/J502-ND/35280

SMA kabel for ¼ bølgelengde antenne	(2 x) ½		Digikey: https://www.digikey.no/product-detail/en/cinch-connectivity-solutions-johnson/415-0029-024/J3724-ND/457095 Ebay: https://www.ebay.com/itm/New-12-4-SMA-male-Plug-to-SMA-female-Jack-RF-Coaxial-Coax-pigtail-Cable-SY/252308155047?hash=item3abebceaa7:g:bGwAAOSwwpdW2LeR
Micro SD kort	1		2 Gbyte (billigste type)
3,7V 1000 mAh Lipo Battery	1		
Batteriledninger for to kort	2 (1)		Det trengs vel strengt tatt ikke ledning til mottakerkortet
Batterilader med mikro USB-plugg	1		
Fallskjerm	1		
Feste for fallskjerm (Øyebolt M5)	1		
Testledning ca. 30 cm	2		Banggood: https://www.banggood.com/40pcs-20cm-Male-To-Male-Color-Breadboard-Cable-Jumper-Cable-Dupont-Wire-p-70127.html?rmmds=search&cur_warehouse=CN
USB A til mikro - kabel	1		Digikey: https://www.digikey.no/product-detail/en/phihong-usa/IP-USB1(C10)S/993-1294-ND/4935860
Mekanikk - 3D printet eller aluminium	1		
CanSat-kort	2		NAROM

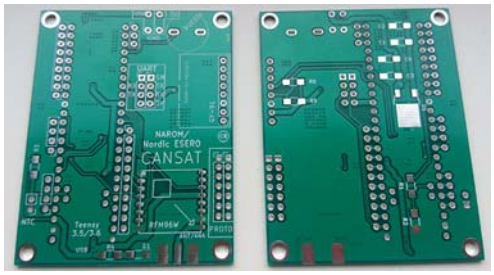
Bildet under viser en oversikt over de ulike komponentene.

2 stk. Prosessorkort - Teensy 3.5-1 med SD-kort holder



1 stk. Spennings-regulator

1 stk Buzzer



2 stk. CanSat-kort, en for sender en for mottaker

2 stk. Radiomoduler

1 stk. NTC-motstand

9 stk. motstander
6·4,7 kΩ, 2·470 Ω
1·10 kΩ

4 stk. kondensatorer
2·10 µF, 2·1 µF

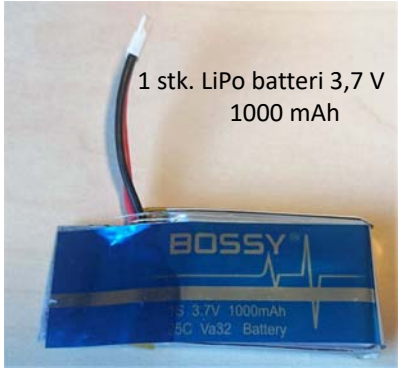
2 stk. stiftlist a 36 stifter

2 stk. Lysdiode

2 stk. antennekabler

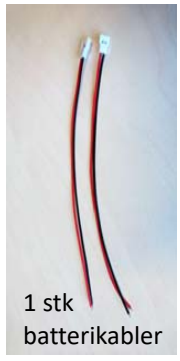
2 stk. SMA-kontakt for ant.

1 stk. GY-91 Sensorkort



1 stk. LiPo batteri 3,7 V 1000 mAh

1 stk. batterilader for USB-tilkobling



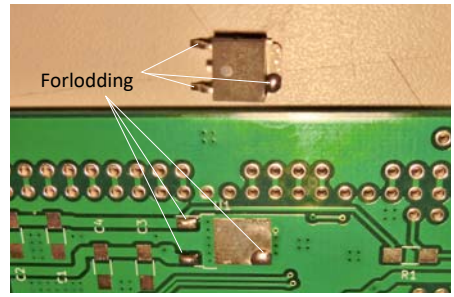
1 stk batterikabler

2 stk Jumpere

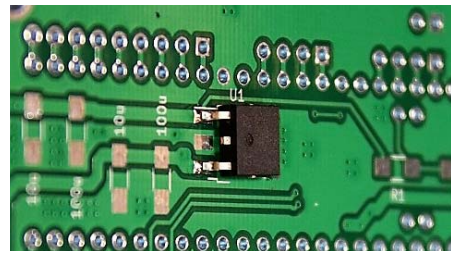
2.2 Monteringsanvisning

Monteringasanvisningen er hentet fra NAROM sine nettsider, men supplert med noen nye bilder[1].

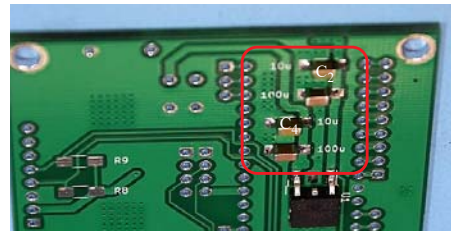
1. ***Lodding av overflatemonterte komponenter (SMD)***
   Lodding av overflatemonterte komponenter kan være krevende siden de er små. Det kan være lurt å legge på litt loddetinn på en av pad'ene før komponenten plasseres på kortet. Deretter plasseres komponenten slik at terminalen som skal loddes berører loddingen som så varmes opp samtidig som komponenten presses ned med en pinsett. Deretter loddes den andre terminalen.



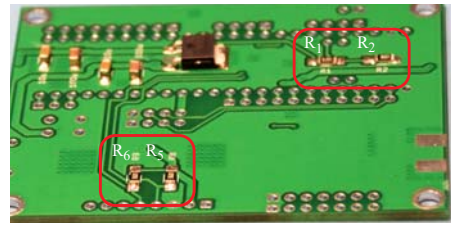
2. ***Spenningsregulator***
   Spenningsregulatoren monteres på kortets loddeside (baksiden) som vist på bildet til høyre. Posisjon *U1*. Husk at komponenten har tre terminaler, to foran og en bak. Det kan være viktig at komponentens underside er presset ned mot kortet for å bedre kjøling. Påfør gjerne litt loddetinn på et av beina før den plasseres på kortet.



3. ***Kondensatorer***
   Fire kondensatorer monteres på kortets loddeside (baksiden) som vist på bildet til høyre. Posisjonene C_1 og C_3 – 10 µF, og C_2 og C_4 – 1 µF. Komponentene loddes til koblingspunktene (pad'ene).

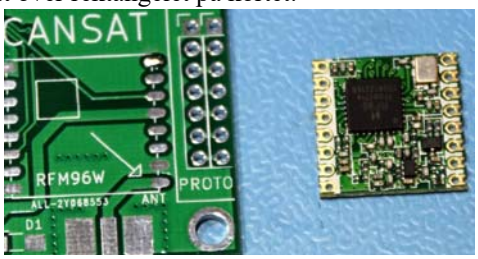


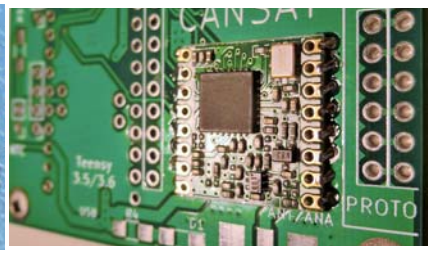
4. ***Fire motstander***
   Monter fire motstander med verdi 4,7 kΩ på loddesiden (baksiden) som vist på bildet til høyre. Posisjonene R_1 og R_2 – 4,7 kΩ, og R_5 og R_6 – 4,7 kΩ. Komponenten loddes til koblingspunktene.



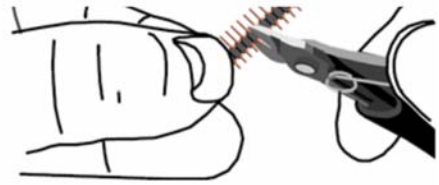
1. https://www.narom.no/undervisningsressurser/the-cansat-book/v2018/constructing-the-narom-2018-can-sat-shield/

5. ***Radiomodulen***
   Monter radiomodulen (RFM-96-433) på komponentsiden av kortet (forsiden). Vær sikker på at modulen er montert riktig vei. Den svarte integrerte kretsen hos radiomodulen skal plasseres rett over rektangelet på kortet.

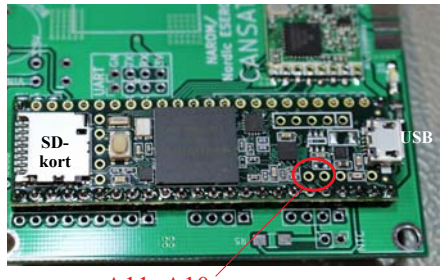




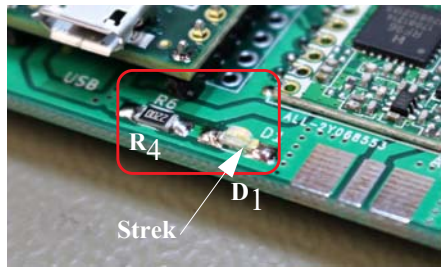
6. ***Stiftlisten***
   Del opp stiftlisten i henholdsvis 2 x 24 pin, 1 x 8 pin og 1 x 2 pin ved hjelp av en sideavbiter. De to 24 pins stiftlistene og den med 2 pinner skal brukes til å montere Teensy 3.5. Den med 8 pinner monteres på GY91.

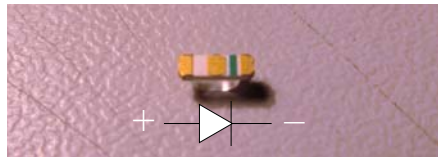


7. ***Teensy 3.5***
   Stikk stiftlistene ned i hullradene der Teensy'en skal monteres og plasser Teensy'en på toppen av stiftlistene. ***MERK spesielt de to stiftene som skal plasseres slik at de treffer analog inngangene A10 og A11.*** Pass på at stiftlistene treffer hullene og at USB-kontakten er på rett side som vist på figuren til høyre. Lodd så først stiftlistene til Teensy'en, deretter snus kortet og lodd stiftlistene til kontaktpunktene på loddesiden av kortet.

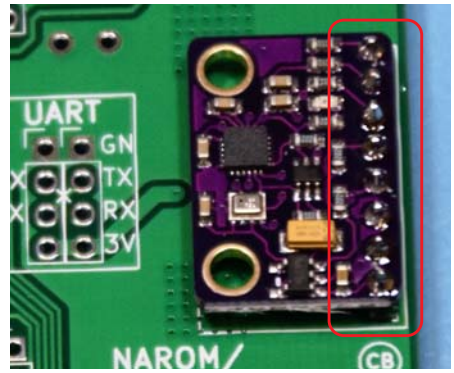


8. ***Lysdioden***
   Monter lysdioden (D_1) og seriemotstanden (R_4 – 470Ω) som vist på bildet til høyre. Påse at lysdioden monteres riktig vei. På undersiden har dioden er en liten blågrønn strek, denne skal monteres slik at den er til høyre som vist på figuren. Dette er diodens katode, dvs. den siden som skal plasseres nærmest jord (GND)





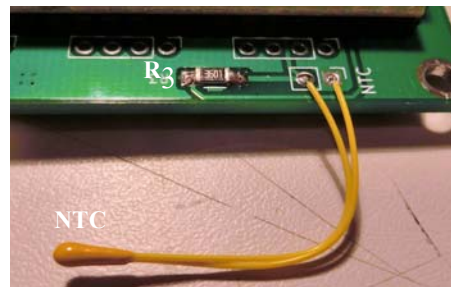
9. ***Sensorkort - GY-91***



Monter sensorkortet ved hjelp av 8 pins stiftlist. Sett stiftlisten ned i CanSat-kortet. Lodd så fast en av pinnene til kortet og sjekk at stiftlista står vinkelrett på kortet. Om den er skjev, løsne loddingen og rett opp. Lodd fast resten av pinnene når lista står rett.

Plasser deretter sensorkortet på stiftlisten. Lodd fast en av kontaktpunktene på sensorkortet til en av pinnene til stiftlisten. Undersøk om sensorkortet er parallelt med kortet. Løsne loddingen og rett opp dersom kortet er skjevt. Når det er rett, lodd fast resten av pinnene. Det er viktig at kortet er montert parallelt med CanSat-kortet slik at x-, y- og z-retningene for akselerometeret, magnetometeret og gyrometeret blir riktig orientert i forhold til CanSat-kortet.

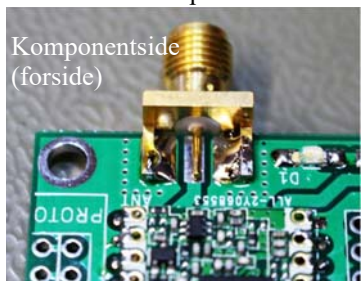
10. ***NTC-motstand***

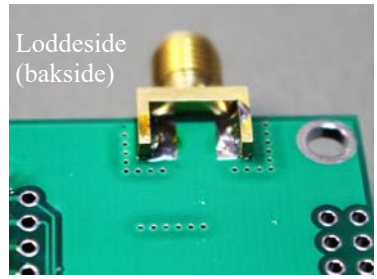


Monter NTC-motstanden (R_{NTC}= 10 kΩ) og seriemotstanden (R_3 = 4,7 kΩ/10 kΩ) som vist på bildet til høyre. Bruk gjerne forlodding av en av padene.

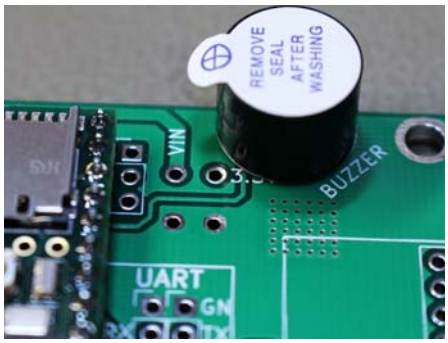
11. ***SMA-kontakt for antennen***

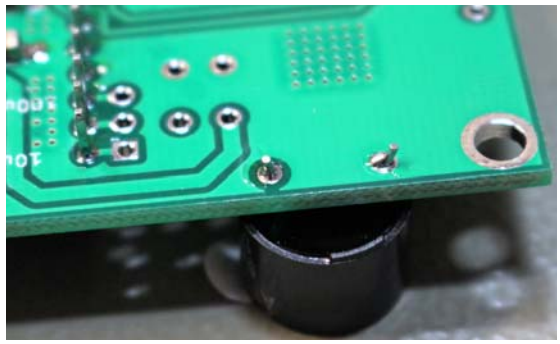
Monter SMA-kontakten som vist på bildet til høyre. Kontakten må loddes på for- og baksiden. Merk at senterlederen, som er antenneledningen, skal loddes til komponentsiden (forsiden). For å få til dette er det viktig at kontakten monteres rett vei som vist på bildet under.



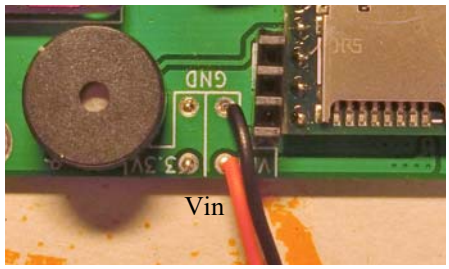


12. ***Buzzer***
    Monter buzzeren. Pass på at den monteres rett vei med + på riktig side. Dersom klistrelappen er falt av så står det også en + på siden av buzzeren sammen med teksten.

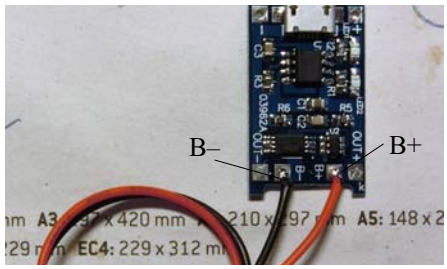




13. ***Batteriledning***
    Monter batterikontakten til CanSat-kortet. Pass på at polariteten blir riktig. Pluss på kontakten skal kobles til Vin, og minus til GND.



14. ***Batterilader***
    Byggesettet inneholder en liten ladekrets for batteriet. Denne skal gjøre det mulig å lade batteriet fra USB-porten på PC'en. En batterikontakt loddes til kretsen som vist på figuren til høyre. Rød ledning kobles til B+ og sort ledning til B–.



2.3 Antennen

Antennen må være en myk antenne som kan kveiles opp sammen med CanSat'en inne i raketten slik at den ikke hindrer den i å bli skutt ut av raketten..

Vi lager derfor en enkel trådantenne som monteres til en SMA-kontakt. Normalt vil en slik antenne være 1/4–bølgelengde og laget av isolert ledning. Tykkelsen er relativt ukritisk, det viktigste er at den er robust og fleksibel. Antennelengden kan lett beregnes fra ligning (2.1) når frekvensen, *f*, og lyshastigheten, *c*, er kjent:

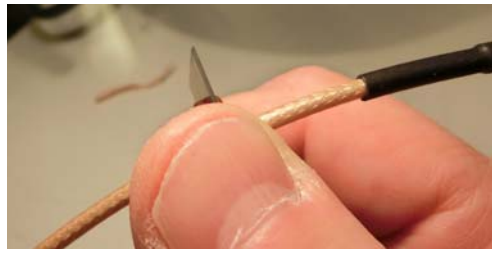
$$L = \frac{c}{4f} = \frac{3\times10^{8}}{4\cdot 434\times10^{6}}[\text{m}] = 0{,}173\ \text{m} \qquad (2.1)$$

Frekvensen 434 MHz gir dermed en antennelengde på 17,3 cm. Lengen er ikke svært kritisk.

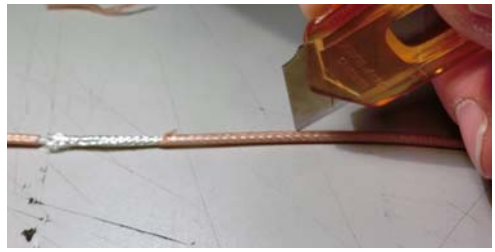
I vårt tilfelle lages antennen av en koaksialkabel som har en senterleder som er adskilt fra en omkringliggende flettet strømpe av fortinnet kobber med en isolerende strømpe i klar plast. Lengden av antenna regnes fra det stedet på kabelen der kobberstrømpa er fjernet og fram til enden av senterlederen.

Figurene under viser hvordan plastbeskyttelsen og metallstrømpa kan fjernes.

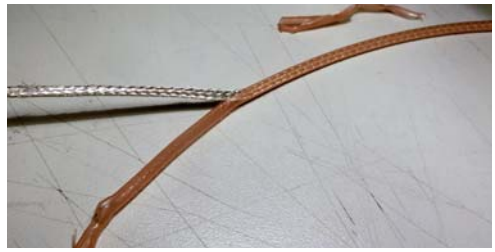
15. ***Mål opp og ut i plastbelegget***
    Mål opp 173 mm langs kabelen og marker begynnelse og ende. Start gjerne noen cm fra kontakten. Bruk en skarp kniv og skjer gjennom plastbelegget ved begynnelsen og hele veien rundt slik at belegget i prinsippet kan dras av. Dette er imidlertid ikke så enkelt siden den sitter godt fast i metallstrømpa, så derfor ..



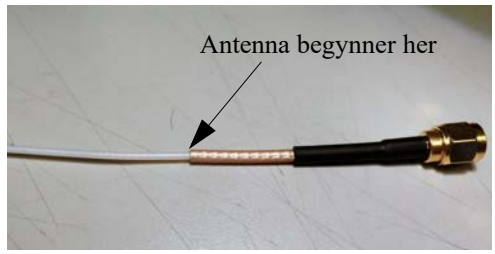
16. ***Skjær plastbelegget på langs***
    ... bruk kniven til å skjære langs plastbelegget slik at du kommer ned til metallstrømpa. Det gjør ikke noe at kniven kutter kordeler i metallstrømpa da den skal fjernes likevel, men unngå å skjær inn i den innerste blanke plastisolasjonen rundt senterlederen.



17. ***Dra av belegget***
    Dra av plastbelegget slik at strømpa blir blottlagt. Skjær og dra av helt til strømpa er fjernet.

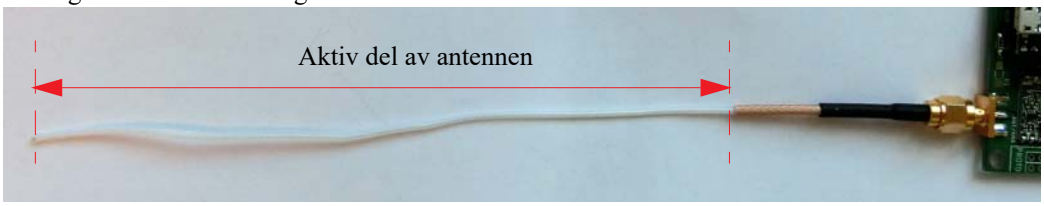


18. ***Fjern metallstrømpa***
    Etter at plastbelegget er fjernet er det lett å fjerne metallstrømpa. Bruk en sideavbiter og klipp metallstrømpa løs ved begynnelsen av antenna om den ikke alt er kuttet av. Plastbelegget som dekker den innerste senterlederen skal være intakt. Mål opp 173 mm og kapp antennekabelen i enden.



19. ***Aktiv antennelengde***

Figuren under viser lengden av den aktive delen av antenna.



3 Kretsbeskrivelse

I dette kapittelet skal vi se på de enkelte delene av grunnmodulen[2].

Siden 2012 har man benyttet Arduino UNO med et eget CanSat-shield-kort utviklet ved universitetet i Århus som grunnstamme i CanSat'en som har vært benyttet ved NAROMs kurs. Denne varianten har hatt både fordeler og ulemper.

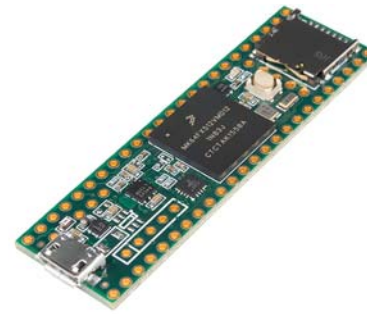
Fordelene har vært at Arduino-konseptet begynner å bli kjent i skolen, dessuten finnes det et stort utvalg tilleggsutstyr og sensorer tilpasset Arduino-familien.

Ulempen ved bruk av Arduino UNO i CanSat har primært vært knyttet til klokkehastighet og manglende kompatibilitet med 3,3 V sensorer. Selv om Arduino UNO leverer 3,3 V så skaper det noen ganger litt ekstra arbeid for å tilpasse signalspenningene fra sensorene til 5V inngangene hos Arduino UNO.

Vinteren 2017/18 ble derfor utviklet en ny variant ved NAROM basert på Teensy 3.5.

3.1 CanSat-kortet (ver. 7.0) og Teensy 3.5

3.1.1 Spesifikasjon for Teensy 3.5



Teensy er et mikrokontroller kort med innebygget lager for programmer og data. I tillegg er kortet utstyrt med en SD-kort skriver/leser som gjør det mulig å lagre store mengder data. Som for Arduino så kan kortet kommunisere og programmeres direkte via en USB mikro kabel som kan plugges rett inn i kortet

Kortet er bygget opp omkring en 32 bits, 120 MHz ARM-prosessor. Kortet har 58 generelle digitale inn/utganger, hvorav 42 er tilgjengelige på pinner eller pader bak på kortet. Alle kan brukes med interrupt og alle tåler 5 V, selv om de er beregnet på 0 – 3,3 V. Dessuten kan 20 av dem brukes til pulsbreddemodulasjon. Teensy har videre 25 analoge innganger (to ADC) med 13 bits oppløsning (0 – 8191). I tillegg har den to analoge utganger (DAC) med 12 bits oppløsning (0 – 4095).

For kommunikasjon med sensorkort har den 3 x I^2C serie busser og 3 x SPI serielinjer,

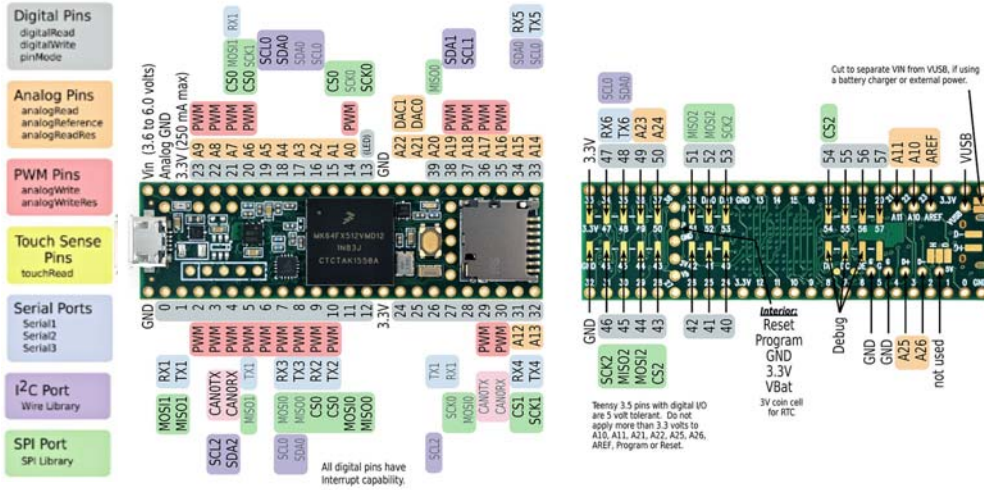
For den spesielt interesserte henvises til følgende spesifikasjon[3]:

- 120MHz ARM Cortex-M4 with Floating Point Unit
- 512K Flash, 192K RAM, 4K EEPROM
- Microcontroller Chip MK64FX512VMD12
- 1 CAN Bus Port
- 16 General Purpose DMA Channels
- 5V Tolerance on All Digital I/O Pins

2. Beskrivelsen er delvis hentet fra https://www.narom.no/undervisningsressurser/the-cansat-book/v2018/
3. https://www.pjrc.com/teensy/techspecs.html

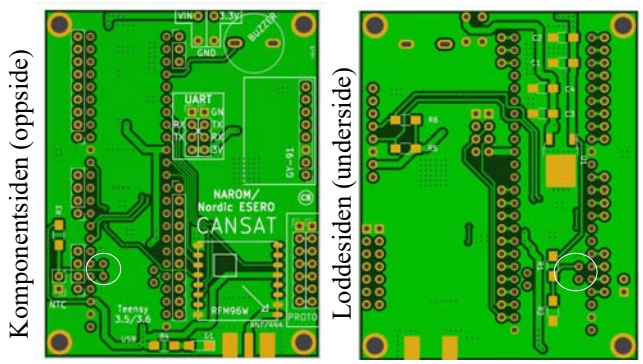
- 58 I/O Pins (40+2 breadboard friendly)
- 25 Analog Inputs to 2 ADCs with 13-bit resolution
- 2 Analog Outputs (DACs) with 12-bit resolution
- 20 PWM Outputs (Teensy 3.6 has 22 PWM)
- USB Full Speed (12Mbit/sec) Port
- Ethernet mac, capable of full 100Mbit/sec speed
- Native (4-bit SDIO) micro SD card port
- I2S Audio Port, 4-Channel Digital Audio Input & Output
- 14 Hardware Timers
- Cryptographic Acceleration Unit
- Random Number Generator
- CRC Computation Unit
- 6 Serial Ports (2 with FIFO and Fast Baud Rates)
- 3 SPI Ports (1 with FIFO)
- 3 I2C Ports
- Real-Time Clock
- 62.3mm x 18.0mm x 4.2mm (2.5in x 0.7in x 0.2in)

Figuren under viser pinningen på selve Teensy-kortet versjon 3.5. Som vi ser har hver pinne flere funksjoner som styres av programmet. Vi legger også merke til at enkelte pinner kan være både analoge og digitale, inn- eller utganger.

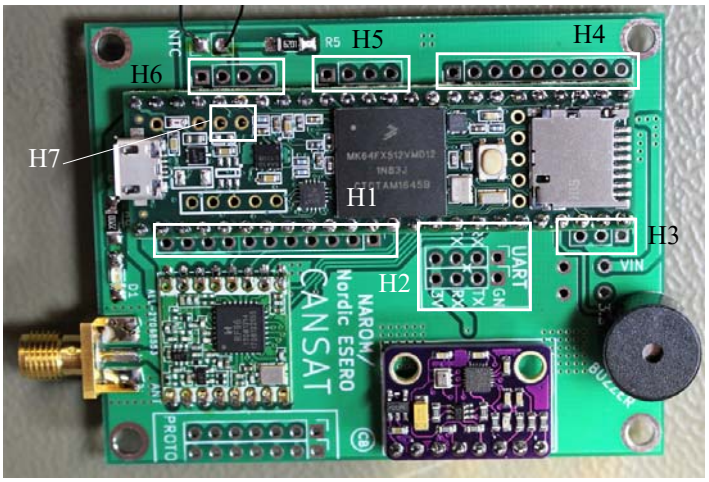


3.1.2 Layout CanSat-kortet

Figuren til høyre viser layout for CanSat-kortet (ver. 7.0). Legg spesielt merke til at to av pinnene innenfor den ytre raden er tatt i bruk til NTC (A10) og batterisjekk (A11) (ringet inn).



Ikke alle pinnene er tatt ut på komponentsiden (oppsiden) av CanSat-kortet.



La oss se hvilke pinner som er gjort tilgjengelige på CanSat-kortet.

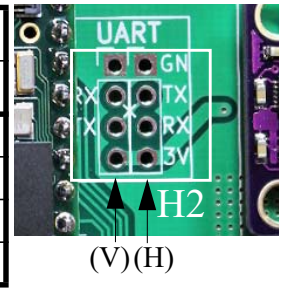
Oversikt over pinning

H1 omfatter hovedsakelig digitale inn- og utganger og serie kommunikasjon

Header 1 (H1) og 2 (H2)					
#	DIO	PWM	Serieport	I^2C	Kommentar
2	0	-	RX1	-	H2 - RX 1 (til høyre) H2 - RX 1 (til venstre)
3	1	-	TX1	-	H2 - TX1 (til høyre) H2 - TX1 (til venstre)
4	2	PWM	-	-	
5	3	PWM	-	SCL2	
6	4	PWM	-	SDA2	
7	5	PWM	-	-	
8	6	PWM	-	-	
9	7	PWM	RX3	SCL0	
10	8	PWM	TX3	SDA0	
11	9	PWM	RX2	-	
12	10	PWM	TX2	-	

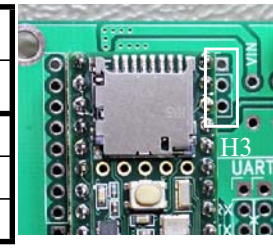
H2 er beregnet for serie-kommunikasjon (UART). De to kontaktene (4 pin) er koblet til de samme portene, men Rx og Tx er speilet i forhold til hverandre i venstre (V) og høyre (H) kolonne.-

Header 2 (H2)					
#	DIO	PWR	Serieport (V)	Serieport (H)	Kommentar
1		GND	-	-	
2	0/1	-	RX1	/TX1	
3	1/0	-	TX1	/RX1	
15		3,3V	-		



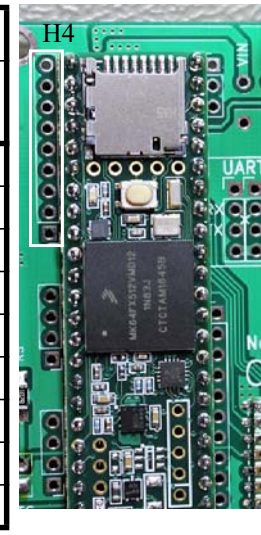
H3 omfatter hovedsakelig analoge innganger, digitale inn- og utganger og serie-kommunikasjon

Header 3 (H3)					
#	DIO	PWM	Analog	Serieport	Kommentar
22	30	PWM	-	-	
23	31	-	A12	RX4	
24	32	-	A13	TX4	



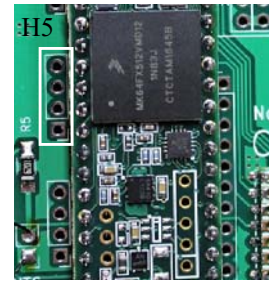
H4 omfatter hovedsakelig analoge og digitale inn- og utganger, og serie kommunikasjon

Header 4 (H4)							
#	DIO	PWM	Analog inn	Analog ut	Serieport	I^2C	Kommentar
25	33	-	A14	-	TX5	-	
26	34	-	A15	-	RX5	-	
27	35	PWM	A16	-	-	-	
28	36	PWM	A17	-	-	-	
29	37	PWM	A18	-	-	SCL1	
30	38	PWM	A19	-	-	SDA1	
31	39	-	A20	-	-	-	
32	-	-	A21	DAC0	-	-	
33	-	-	A22	DAC1	-	-	



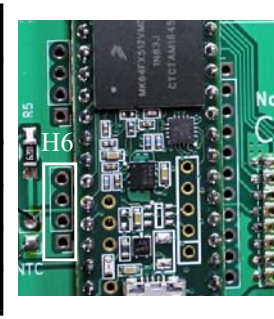
H5 omfatter hovedsakelig analoge innganger og digitale inn- og utganger.

Header 5 (H5)				
#	DIO	PWM	Analog inn	Kommentar
36	14	PWM	A0	
37	15	-	A1	
38	16	-	A2	
39	17	-	A3	



H6 omfatter hovedsakelig analoge innganger og digitale inn- og utganger.

Header 6 (H6)				
#	DIO	PWM	Analog inn	Kommentar
42	20	PWM	A6	
43	21	PWM	A7	
44	22	PWM	A8	
45	23	PWM	A9	

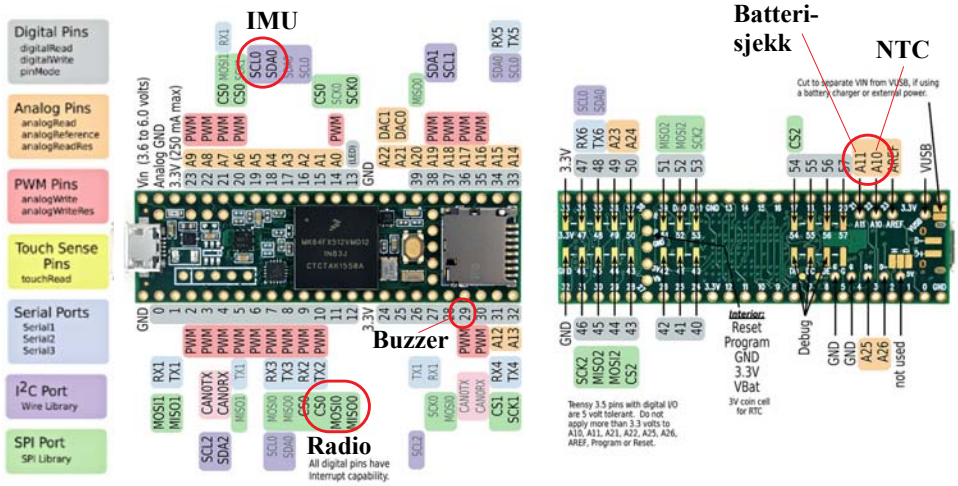


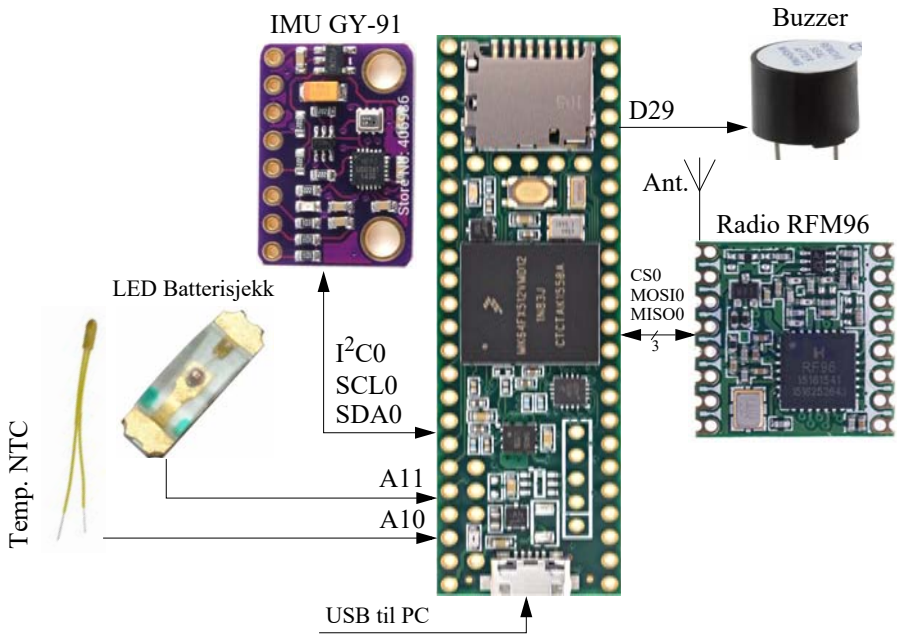
H7 omfatter hovedsakelig analoge innganger og digitale inn- og utganger.

Header 7 (H7)			
#	Analog inn	Power	Kommentar
43B	A11		Batterisjekk
44B	A10		NTC
45B		Aref	Ikke i bruk CanSat
47B		VUSB	Ikke i bruk CanSat

3.1.3 Dedikerte pinner

Noen av pinnene fra Teensy er alt dedikert til spesielle formål. Dette gjelder følgende:





Radioen RF96

Radioen kommuniserer med Teensy via SPI0 (CS0 (p. 12), MOSI0 (p. 13), MIS0 (p. 14))

IMU GY-91

IMU'en kommuniserer med Teensy via I^2C0 (SCL0 (p. 40), SDA0 (p. 41)).

Buzzer (D29)

Buzzeren styres av IO-port 29 (p. 21).

NTC - temperaturmåleren (A10)

NTC motstanden som registrerer temperaturen avleses av analog port A10 (p. 51).

Batterispenningen (A11)

For å holde kontroll med ladenivået i batteriet benyttes en spenningsdeler, som avleses av analog port A11 (p. 52).

3.1.4 Kommunikasjon med omverdenen

USB kontakten er ikke koblet til noen av de eksterne I/O-portene slik som hos Arduino hvor den er koblet til Rx1 og Tx1 (D0 og D1). Rx1 og Tx1 er derfor ledige og er bl.a. ført ut til en ekstra kontakt H2.

4 Programmering

Det fine med å programmere Teensy er at man kan bruke det samme programmeringsverktøyet (IDE) som brukes for Arduino, men da med noen tilleggsbiblioteker og med valg av Teensy 3.5 som kortet som skal brukes.

Språket som anvendes for både Arduino og Teensy er en variant av C hvor programmeringen stort sett består av å bruke ferdig utviklede spesialfunksjoner. Alle kommandoer som kan brukes til Arduino kan også bruks til Teensy 3.5.

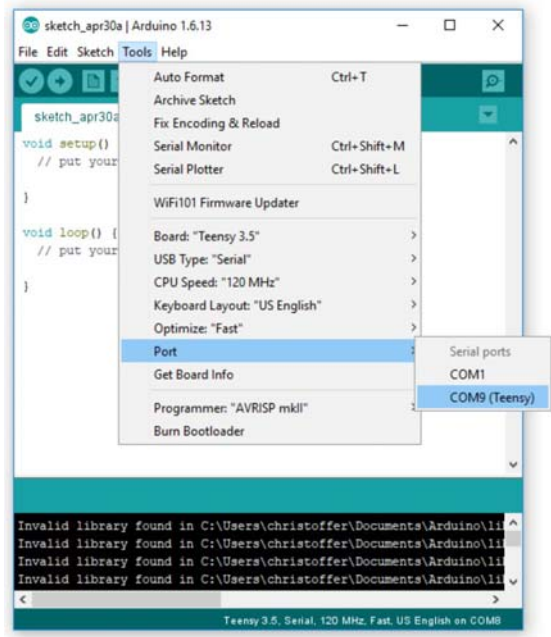
4.1 Installasjon av programvare[4]

Først installeres den vanlige Arduino IDE fra Arduino sin nettside: https://www.arduino.cc/en/Main/Software. I skrivende stund er den siste utgaven for Arduino 1.8.5. Last ned og installer den utgaven som passer for din maskin (Windows, Mac eller Linux).

I tillegg trengs *Teensyduino* som er et programtillegg til den tradisjonelle IDE'en. Denne kan lastes ned fra nettadressen: https://www.pjrc.com/teensy/td_download.html Velg operativsystem og installer programvaren ved å følge anvisningene. I skrivende stund er siste utgave av Teensyduino versjon 1.45.

Når programvaren er installert, åpne Arduino editoren (IDE) og du vil se at tillegget for Teensy er inkludert i menyen under *Tools* (*Verktøy*). Når Teensy er tilkoblet vil den vises på en av portene. Velg porten hvor Teensy er tilkoblet. En må også sørge for å velge riktig versjon av kortet (Teensy 3.5).

Ved oppstart første gang skal det være installert et blink-program som blinker med et blink hvert andre sekund. Neste gang man starter prosessoren vil sist innlastede program starte og gå til strømmen brytes eller et nytt program lastes ned til prosessoren.



4. Stoffet er hentet delvis fra https://www.narom.no/undervisningsressurser/the-cansat-book/v2018/getting-started-with-the-primary-mission/

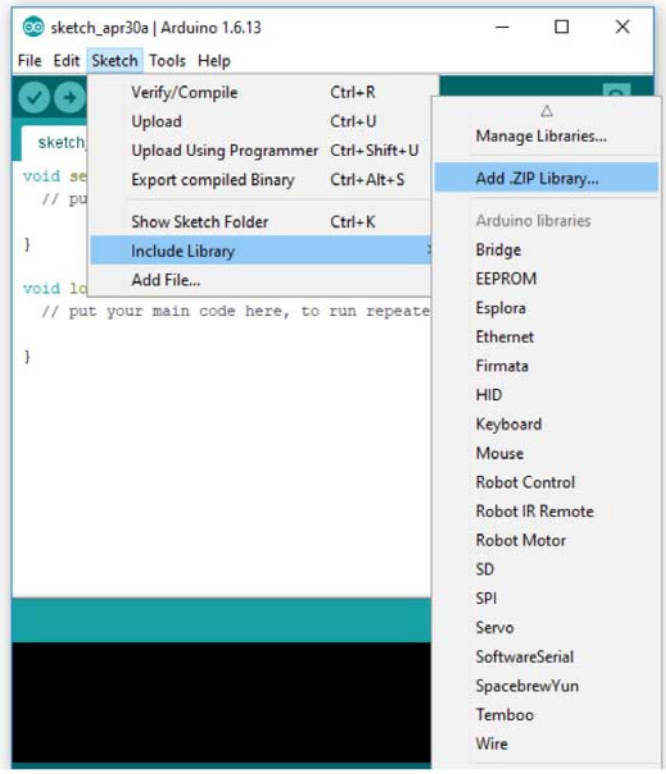
4.2 Installasjon av biblioteker

Dernest må man installere biblioteker for sensorkortet *GY91* og radioen *RFM96*. Begge bibliotekene som skal installeres *GY91* og *CanSat_RFM96* er biblioteker videreutviklet av NAROM.

De zip-pakkede bibliotekene kan installeres direkte i IDE'en ved å velge *Skisse*, *Inkluder Bibliotek*, *Legg til .ZIP bibliotek* og så velge den ønskede .ZIP fila, GY91.zip eller *CanSat_RFM96.zip*.

Bibliotekene skal da installeres automatisk og dukke opp i lista over tilgjengelige biblioteker.

Dersom det mot formodning ikke skulle skje, kan man pakke ut (unzip) de to bibliotekene og legge de to filene (.cpp og .h) direkte inn under henholdsvis katalogen *Dokumenter/Arduino/Libraries/GY91* og *Dokumenter/Arduino/Libraries/CanSat_RFM96*



4.3 Opplasting og kjøring av programmer

Man kan nå laste opp eksisterende programmer eller nye til Teensy 3.5. Dersom man har Arduino-programmer så skal disse kunne kjøres på Teensy som de er. Når man kompilerer og overfører programmet til Teensy, så vil det dukke opp et lite vindu som vist på figuren til venstre. Ved første gang kompilering og overføring vil man normalt måtte trykke knappen på Teensy-kortet for manuelt å sette kortet i programmeringsmodus. Dette vil resultere i at det kommer opp en "status bar" som forteller at programmet overføres (*Programming*) og som avsluttes med *Reboot OK*. Dessuten aktiveres en knapp i vinduet som forteller at overføringen fra nå av vil skje automatisk uten at man trenger å trykke på knappen på Teensy-kortet.

Dersom programmet ikke overføres vil det komme en beskjed med rød farge i meldingsvinduet på IDE'en hvor det f.eks. kan stå:

Teensy did not respond to a USB-based request to automatically reboot. Please press the PROGRAM MODE BUTTON on your Teensy to upload your sketch.

Sjekk at Teensy er skikkelig tilkoblet og at IDE'en er riktig konfigurert (riktig kort, riktig port, o.l.). Forsøk så last over programmet på nytt samtidig som du trykker knappen på kortet. Det er nok med et lett trykk. Det er svært sjelden at dette inntreffer.

4.4 Nye programmeringskommandoer.

Normalt vil alle kommandoer som kan brukes for Arduino familien også kunne brukes for Teensy 3.5. Det handler egentlig om at IDE'en nå er utstyrt med en kompilator som oversetter de kjente Arduino-kommandoene til maskinkode som Teensy 3.5 forstår.

Fast funksjoner

Vi kan likevel merke oss følgende.

Det er utviklet raskere utgaver av analogWrite() og analogRead() som kun er tilgjengelige i Teensy kompilatoren. For å få tilgang til disse skriver man:

```
analogWriteFast(pin,value);
```

... brukes normalt i forbindelse med puls-bredde-modulasjon (pwm), og vil være vesentlig raskere enn den tradisjonelle analogWrite() funksjonen.

```
variabel = analogReadFast(pin);
```

... leser verdien på den analoge porten "pin", og legger verdien i variabelen "variabel".

Set ADC og DAC oppløsning

Disse kommandoene er ikke nye, men de har så langt ikke vært implementert for Arduinoe UNO, kun for Arduino Due og Zero, og nå også for Teensy 3.5. Default verdier for oppløsningen er 10 bit, men siden Teensy har ADC med

```
analogReadResolution(<#bit>);
```

... setter opp den analoge lesefunksjonen til *#bit*, og maks. 13 bit som er oppløsningen til de analoge inngangene. Dersom denne utelates vil oppløsningen være 10 bit (default verdi). Dersom den settes til en høyere verdi en det antallet bit AD-konverteren tilbyr vil det bli fylt på med nuller. Dersom den settes lavere en det tilgjengelige antallet bit, så vil de overskytende minst signifikante bitene bli strøket.

```
analogWriteResolution(<#bit>);
```

... setter opp oppløsningen til DAC'ene. Teensy 3.5 har til forskjell fra Arduino, to porter (A21/DAC0 (p. 32) og A22/DAC1 (p. 33)) hvor man kan omforme en digital variabel til en spenning med et tilsvarende nivå, såkalte DAC (Digital to Analog Converter). Disse har hos Teensy 3.5 en oppløsning på 12 bit. 12 bit gir 4096 nivåer fra 0 – 4095 og gir ut en spenning mellom 0V og 3,3V.

Dersom man bruker denne kommandoen for en digital IO-port med pwm så vil den kunne sette opp oppløsningen til pulsbreddemodulasjonen (pwm). Normalt er den på 8 bit hos de fleste Arduino kontrollerkortene, men Due har 12 bits pwm oppløsning, det samme har Teensy 3.5[5].

Det kan være greit å spesifisere antall bit for lesing og skriving i setup().

5. https://www.pjrc.com/teensy/td_pulse.html

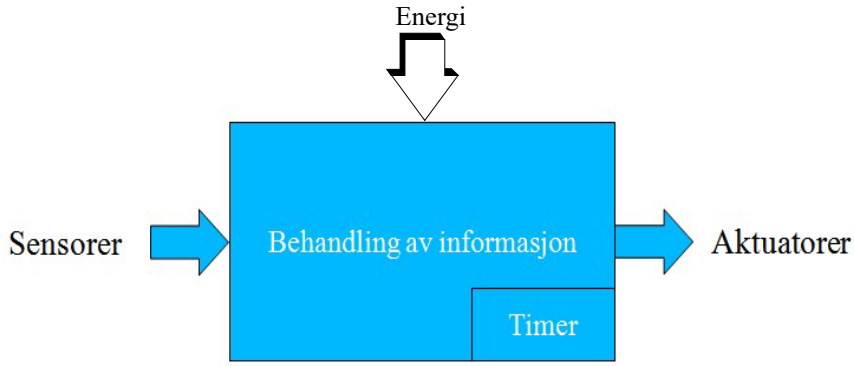
5 Sensorer

I dette kapittelet skal vi se nærmere på de mest aktuelle sensorene. Men la oss først gi en mer generell innledning til sensorer.

5.1 Innledende betraktninger om sensorer

5.1.1 Hva er en sensor?

La oss først sette sensoren inn i en sammenheng. På figuren under ser vi i midten en enhet for *behandling av informasjon*. Til venstre *sensorer* og til høyre *aktuatorer*. En sensor er en enhet som registrerer kvantiteten til en fysisk størrelse i omgivelsene (f.eks. temperatur, lysstyrke, trykk o.l.), mens en aktuator er i denne sammenheng, en innretning som utfører en aksjon (f.eks. tenner et lys, åpner en dør, lager en lyd, viser et bilde o.l.). I noen tilfeller finnes en klokke (timer) som f.eks. gjør at aksjoner utsettes i tid. I tillegg må systemet tilføres energi (f.eks. fra batteri, solceller, strømnettet e.l.). *Databehandlingsenheten bestemmer hvilken aksjon aktuatorene skal utføre og til hvilken tid på bakgrunn av informasjon innhentet fra sensorene.*



Nesten uansett hvilken mekanisk, elektrisk eller elektronisk hjelpemiddel vi tar for oss så kan det puttes inn i denne modellen. La oss se på noen eksempler som drar definisjonen på sensor langt.

- **Strykejernets** "sensor" kan være av/på bryteren og termostaten. Aktuatoren er varmeelementet som gjør at jernet blir varmt.
- **Radioens** "sensorer" er antennen som fanger opp radiosignalene. Det kan også være alle knappene som styres av lytteren. På bakgrunn av informasjonen fra antenne og knapper, gir den ut et hørbart signal, som i denne sammenhengen er lydsignaler i høyttaleren (aktuatoren).
- **Den automatiske døråpneren** har gjerne en bevegelsessensor som registrerer at noen nærmer seg døra. Denne informasjonen behandles og systemet bestemmer at motorer eller hydraulikk (aktuator) skal åpne dørene.

- **Trafikklys** kan være et eksempel på et system hvor sensordelen kan være noe uklar dersom lyset kun slavisk gjennomgår en sekvens. Dette blir enklere dersom det er knapper for å be om grønt lys for fotgjengere eller magnetfeltsensorer i gatelegemet som registrerer kjøretøyer. Aktuatoren er lysene og ev. lydsignaler for fotgjengere.

Finn gjerne flere eksempler og diskuter hvor grensen bør gå for hva vi kan kalle en sensor eller en aktuator.

Definisjon på sensor og aktuator

Følgende er et forsøk på en forenklet definisjon på en sensor og en aktuator:

Definisjon av *sensor*:
*En **sensor** er en transduser som omdanner én fysisk størrelse i omgivelsene til en annen fysisk størrelse som gjerne er elektrisk målbar.*

Definisjon av *aktuator*:
*En **aktuator** er en transduser som omdanner én fysisk størrelse (ofte elektrisk) til en annen fysisk størrelse som kan utføre en oppgave eller et mekanisk arbeid på omgivelsene.*

Veien fra en fysisk størrelse kan noen ganger være kronglete. Lista under viser eksempler på mulige veier fra den fysiske størrelsen til et målbart signal.

En fysisk størrelse, det være seg lys, temperatur, lyd, partikler eller noe annet, vil i sensoren omdannes til en av størrelsene til venstre i lista under:

Fra fysisk størrelse → endring i:

- resistans → spenning/strøm
- induktans → resonanskrets → frekvens/fase (telling eller måling av tid)
- kapasitans → resonanskrets → frekvens/fase (telling eller måling av tid)
- piezo-resistivitet → spenning
- piezo-elektrisk (spenning)
- utvidelse (elastisitet) → resistivitet → spenning
- temperatur (termisk) → resistivitet/ledningsevne → spenning/strøm
- bevegelse → telling eller måling av tid

Selv om den fysiske størrelsen skaper en endring i resistivitet eller kapasitet eller lignende, så er ikke dette noe vi uten videre kan måle. Vi må ofte gå veien om måling av *strøm*, en *telling* eller måling av *tid*. Disse vil så kunne representere den opprinnelige størrelsen.

Sammenhengen mellom den fysiske størrelsen og den målte verdien må modelleres matematisk eller det må utføres en kalibrering som knytter det fysiske fenomenet sammen med den presenterte verdien. Som oftest må man gjøre begge deler, både modellere og kalibrere. For å kunne utføre en kalibrering må man ha en standard som har større nøyaktighet enn det systemet vi skal kalibrere. I slike sammenhenger er det noen egenskaper som er særdeles viktige.

5.1.2 Egenskaper ved sensorer

Sensorer kan karakteriseres med ulike parametere, her er de viktigste:

Måleområde:

Måleområdet til en sensor er området mellom minste og største verdi av den målte størrelsen som ikke gir større avvik enn det vi kan tillate. Øker vi tillatt avvik mellom målt verdi og virkelig verdi, kan som oftest måleområdet utvides. *F.eks. kan -25 °C til +125°C være måleområdet for en temperatursensor. Innen dette området skal ikke avviket fra den virkelige verdien være større enn ± 2°C.*

Oppløslighet:

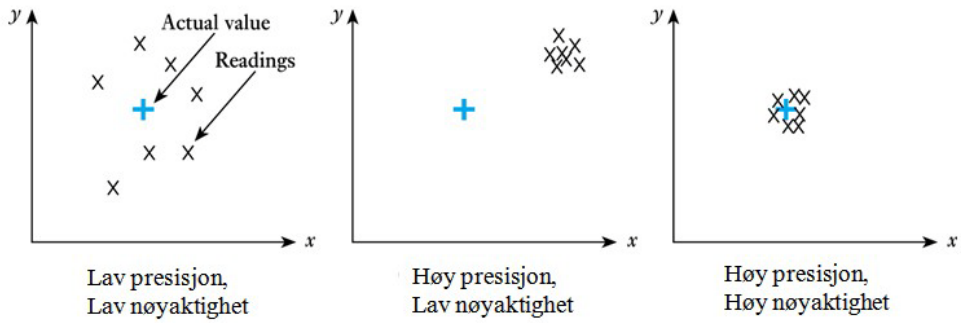
Dette er den minste målbare verdi av den fysiske størrelsen som sensoren er i stand til å registrere, og uttrykkes gjerne som en prosentandel av sensorens totale måleområde. *F.eks. kan 0,15°C være en minste forskjell som en temperatursensor kan måle. Av et måleområde på 150°C vil dette gi en oppløselighet på 1 ‰.* Ofte oppgis nøyaktighet som en prosentandel av fullskala utslaget – prosent av FS.

Målefeil:

Målefeilen er forskjellen mellom målt verdi og virkelig verdi. En målefeil kan deles inn i *tilfeldige* og *systematiske* feil. Tilfeldige feil er ofte spredt omkring den riktige verdien og hyppigheten av de målte verdiene vil avta når avstanden til den riktige verdien øker. Tilfeldige feil vil ha en middelverdi nær den riktige verdien. Systematiske feil er spredt omkring et målepunkt som ligger tilside for den riktige verdien. Midling av flere målinger vil derfor ikke gi riktigere verdier.

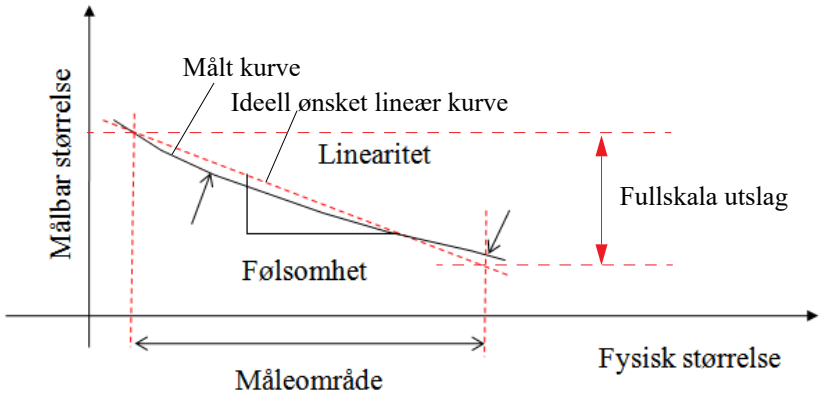
Presisjon og nøyaktighet:

Figuren under viser forskjellen på en måleseries *presisjon* og *nøyaktighet*.



Linearitet:

Ofte er det praktisk at det er en *lineær* sammenheng mellom verdien til den fysiske størrelsen og signalet ut av sensoren. Dette er ikke alltid tilfelle. Men heller ikke en sammenheng som i utgangspunktet skal være det vil være i praksis være helt lineær. Figuren under viser hvordan en kurve kan avvike fra den lineære ønskede verdien.



Vi ser at den målte kurven avviker fra den ideelle. Dette avviket måles mellom maksimale avvik på begge sider av den lineære kurven. Avviket angis i prosent av *fullskala verdien*.

Følsomhet:

Følsomheten til en sensor angis som endring av signalet ut av sensoren for en gitt forandring av den fysiske størrelsen inn på sensoren (se figuren over). *F.eks. kan en temperatursensor ha en følsomhet på 10 mV/°C.*

Vi skal nå se nærmere på noen aktuelle sensorer som er inkludert på CanSat kortet (versjon 7.0). Vi begynner med temperatursensoren.

5.2 Temperaturfølsom motstand (NTC- og PTC-motstander)

Metaller vil normalt ha økende resistans med økende temperatur. I et halvledermateriale vil flere ladningsbærere løftes opp i ledningsbåndet slik at ledningsevnen går opp, dvs. at resistansen blir mindre.

De fleste motstandsmaterialer endrer resistans som funksjon av temperaturen. Som regel er dette uønsket, men i noen spesielle tilfeller ønsker man nettopp en slik endring og utformer komponenten og materialet deretter. Slike motstander brukes også i forbindelse med måling eller deteksjon av temperaturendringer, eller til å motvirke uønsket temperaturdrift i elektronisk utstyr.

- NTC – Negative Temperatur Coefficient, dvs. at resistansen avtar med økende temperatur.
- PTC – Positive Temperatur Coefficient, dvs. at resistansen øker med økende temperatur.

5.2.1 NTC-motstanden

NTC-motstander er laget av et materiale hvis resistivitet varierer sterkt med temperaturen. Som navnet sier (Negative Temperature Coefficient – NTC) så avtar resistansen med økende temperatur.

NTC-motstander er derfor vanligvis bygget opp som en polykrystalinsk *halvleder* som kan bestå av en blanding av krom, mangan, jern, kobolt og nikkel, som sintres[6] sammen med et plastisk bindemiddel.

En forenklet sammenheng mellom resistansen (R_{NTC}) og temperaturen (T) kan uttrykkes som:

$$R_{NTC} = Ae^{B/T} \tag{5.1}$$

hvor A og B er *tilnærmet konstante* innen begrensede temperaturområder.

I datablader for NTC-motstander oppgis gjerne resistansen (R_r) for en referansetemperatur (T_r). I et temperaturområde rundt denne referansetemperaturen antas B-verdien å være tilnærmet konstant ($B_{25/85}$ – B-verdien er tilnærmet konstant innen området 25°C til 85°C).

Vi kan da sette opp følgende to ligninger:

$$R_{NTC} = Ae^{\frac{B_{25/85}}{T}} \tag{5.2}$$

$$R_r = Ae^{\frac{B_{25/85}}{T_r}} \tag{5.3}$$

Ved å eliminere A fra disse uttrykkene, kommer vi fram til følgende sammenheng, løst med hensyn til resistansen R_{NTC} i NTC-resistoren:

$$R_{NTC} = R_r \cdot e^{\left(\frac{B_{25/85}}{T} - \frac{B_{25/85}}{T_r}\right)} \tag{5.4}$$

Dette uttrykket går under betegnelsen *Beta-formelen*.

Når vi skal beregne verdien for en NTC-motstand ved en gitt temperatur, slår vi opp B-verdien, R_r og T_r i databladet, sørger for at de aktuelle temperaturene ligger innenfor området til B-verdien, og beregner R ved å sette inn ønsket temperatur T. Temperaturen angis i grader Kelvin.

Siden det ofte er B for området 25° – 85°C som er oppgitt kan en lett oppleve at man havner på siden av det spesifiserte området, siden vi ofte ønsker å måle i området 0° – 25°C. Siden vi linearisere og kalibrerer sensoren trenger ikke dette bety så ny for vår anvendelse.

6. Sintring betyr at metallpulver knyttes sammen ved hjelp av oppvarming, men uten å smelte.

5.2.2 NTCLE201E3103S

Fra databladet[7] for *NTCLE201E3103S* finner vi følgende: R_{25} er referansemotstand (R_r) ved 25 °C (T_r = 298 K):

ELECTRICAL DATA AND ORDERING INFORMATION				
R_{25} (Ω)	R_{25}-TOL. (± %)	$B_{25/85}$ (K)	$B_{25/85}$-TOL. (± %)	SAP MATERIAL AND ORDERING NUMBER
3000	2.18	3977	0.75	NTCLE201E3302SB
5000	2.18	3977	0.75	NTCLE201E3502SB
10 000	2.18	3977	0.75	NTCLE201E3103SB
3000	2.18	3977	0.75	NTCLE300E3302SB
5000	2.18	3977	0.75	NTCLE300E3502SB
10 000	2.18	3977	0.75	NTCLE300E3103SB

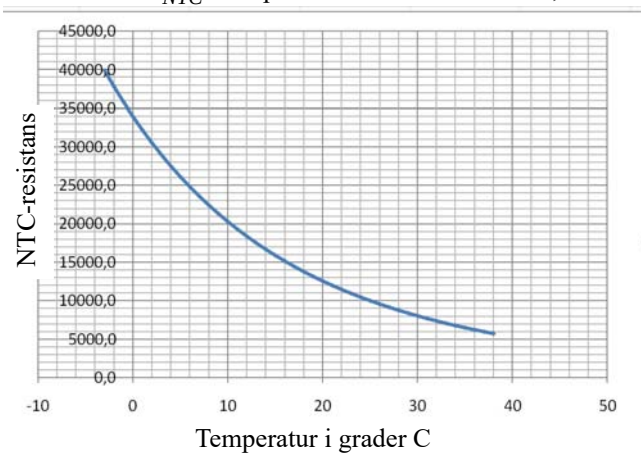
Figur 5.1 Datablad for NTC-motstand NTCLE201E3103SB, 3–10 kΩ

Med disse dataene kan vi skrive:

$$R_{NTC} = 10\text{k} \cdot e^{\left(\frac{3977}{T} - \frac{3977}{298}\right)} \tag{5.5}$$

hvor $B_{25/85}$ = 3977 (NTCLE201E3103SB – 10 kΩ) og referansetemperaturen T_r = 298 K.

Dersom vi beregner verdier for R_{NTC} i temperaturområdet 25° – 85°C, får vi følgende graf:



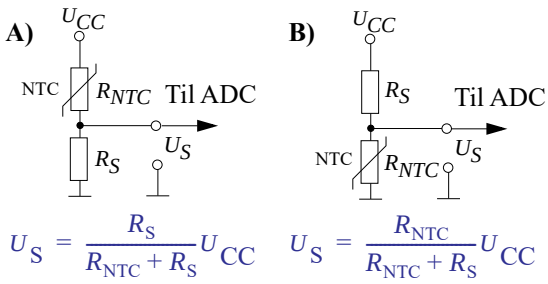
Figur 5.2 NTC motstand som funksjon av temperaturen NTCLE201E3103SB – 10 kΩ

7. Databladet er hentet fra: http://www.elfa.se/pdf/60/06027916.pdf

En annen viktig parameter for NTC-motstander, er hvor raskt resistansen endrer seg ved sprang i temperaturen. Denne parameteren betegnes *NTC-motstandens tidskonstant* (τ), og angir den tiden det tar for resistansen og endre seg til 63,2% av den nye resistansen etter at temperaturen har endret seg 1 K (Kelvin) over omgivelsestemperaturen. En antar at temperaturendringen ikke er forårsaket av indre oppvarming på grunn av elektrisk strøm som flyter gjennom motstanden.

Vi var ikke istand til å finne en verdi for tidskonstanten for den aktuelle NTC-motstanden, men det er rimelig å anta at den er under 10 sek. avhengig av omgivelsene (stillestående eller luft i bevegelse).

5.2.3 Oppkobling mot ADC



Siden grensesnittet til kontrolleren krever en spenning, kobles NTC-motstanden i serie med en motstand som vist i figuren til høyre. Dersom vi velger verdien til seriemotstanden lik referanseverdien til NTC-motstanden (R_{25}), så vil det vise seg at sammenhengen mellom temperatur og spenning blir relativt lineært i området rundt referansetemperaturen (T_r). Spenningsnivået, U_S, beregnes fra formlene som antydet på figuren. Legg merke til at oppkoblingen på tegning A gir økende spenning, U_S, med økende temperatur, mens oppkoblingen i tegning B gir fallende spenning, U_S, med økende temperatur.

I vårt tilfelle er NTC-motstanden plassert nærmest GND som vist i figur B over. Vi får da en fallende spenning som funksjon av økende temperatur.

5.2.4 Optimal seriemotstand

Vi har registrert at motstandsverdien til NTC-motstanden er svært ulineær som funksjon av temperaturen. Nå er det ikke motstanden vi måler, men spenningen på utgangen av spenningsdeleren. Så la oss se hvordan spenningen avhenger av temperaturen og hvordan lineariteten varierer med verdien til seriemotstanden.

På bakgrunn av ligningene foran kan vi sette opp et uttrykk for temperaturen som funksjon av spenningen som ev. kan legges inn i programmet i mikrokontrolleren.

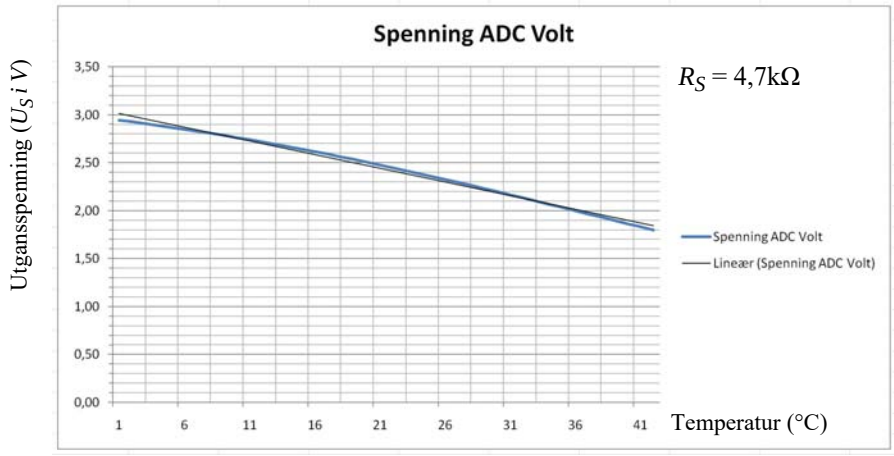
Vi setter:

$$R_{NTC} = 10\text{k} \cdot e^{\left(\frac{3977}{T} - \frac{3977}{298}\right)} \tag{5.6}$$

inn i ligningen:

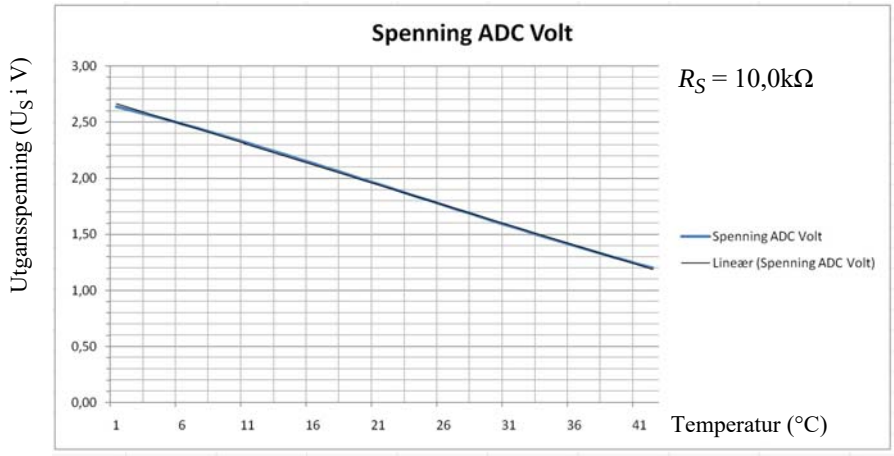
$$U_S = \frac{R_{NTC}}{R_{NTC} + R_S} U_{CC} \tag{5.7}$$

... og vi kan beregne U_S som funksjon av temperaturen for ulike verdier av seriemotstanden, R_S. I figuren under har vi modellert spenningen U_S som funksjon av temperaturen i området 0 – 42°C med en seriemotstand på R_S = 4,7 kΩ:



Sammen med spenningskurven har vi lagt inn den best tilpassede lineære sammenhengen (tynn rett linje). Vi legger merke til at avvikene er betydelig i endene av området. I temperaturområdet 0 – 42°C er spenningssvinget lik U_{diff} = 0,82 V.

Dersom vi imidlertid velger en seriemotstand R_S = 10kΩ ser kurven langt mer lineær ut.



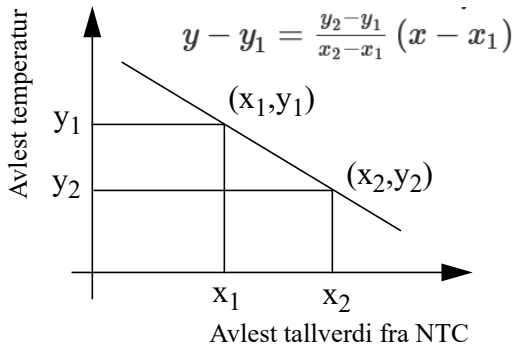
Vi registrerer dermed at avvikene ved endene av området er langt mindre med 10kΩ enn ved 4,7kΩ. Dessuten er spenningssvinget i området 0 – 42°C lik U_{diff} = 1,08 V. Dvs. 10kΩ gjør ikke bare kurven mer lineær, men man utnytter det dynamiske området til ADC'en bedre.

Vi anbefaler derfor å bruke en seriemotstand på 10kΩ for *NTCLE201E3103SB* dersom man ønsker å linearisere funksjonen. Vi skal senere se hvordan man kan beregne temperaturen direkte med et uttrykk for resistansen i NTC-motstanden som er bedre tilpasset den virkelige sammenhengen.

5.2.5 Kalibrering av temperatursensoren

Denne beregningen antar at det er en omtrent lineær sammenheng mellom måleverdi og temperaturer i det aktuelle temperaturområdet. Dette er ikke langt fra sannheten i et avgrenset område (f.eks. 0 – 42°C) dersom man velger motstandsverdien i seriemotstanden lik referansemotstanden. Under denne betingelsen kan følgende metode benyttes med tilstrekkelig nøyaktighet:

1. Bruk et termometer å mål "virkelig" temperatur f.eks. innendørs (y_1), les av tallverdien (gjerne den digitale verdien) som kommer fra NTC-motstanden (x_1).
2. Ta så Teensy'en med ut samtidig som du logger data (vi antar at det er kaldere ute). Mål "virkelig" temperatur med et termometer (y_2) og les av tallverdien fra CanSat (x_2).



Figuren over viser hvordan topunktsformelen kan brukes for å finne et uttrykk for sammenhengen mellom målte verdier og temperatur.

Topunktsformelen for lineære ligninger kan også skrives slik:

$$y = ((y_2 - y_1)/(x_2 - x_1)) * (x - x_1) + y_1 \quad (5.8)$$

Hvis $k = (y_2 - y_1)/x_2 - y_2)$ (stigningskoeffisienten), kan vi skrive ligningen slik:

$$y = k\,(x - x_1) + y_1 \quad (5.9)$$

Sett verdiene inn i formelen over og finn et uttrykk for y (temperatur i °C) som funksjon av tallverdien hentet fra AD-konverteren som måler spenningen fra NTC-motstanden (x).

5.2.6 Bruk av interpolasjonsformel for sammenheng mellom resistans og temperatur

I databladet for *NTCLE201E3103SB*[8] finner vi et interpolasjonsformel for sammenhengen mellom resistansen for NTC-motstanden og temperaturen. Ligning 5.10 er et uttrykk for resistansen i NTC-motstanden som funksjon av temperaturen T målt i K.

$$R_{NTC} = R_{ref} \times e^{\left(A + \frac{B}{T} + \frac{C}{T^2} + \frac{D}{T^3}\right)} \quad (5.10)$$

Hvor *A*, *B*, *C* og *D* er konstanter som er avhengig av framstillingen av NTC-motstanden. R_{ref} er referansemotstanden ved 25°C. T er temperaturen målt i K.

Ligning 5.10 kan vi løse mht. temperaturen *T* og vi får da følgende nye uttrykk:

$$T = \left(A_1 + B_1 \ln\left(\frac{R_{NTC}}{R_{ref}}\right) + C_1 \ln\left(\frac{R_{NTC}}{R_{ref}}\right)^2 + D_1 \ln\left(\frac{R_{NTC}}{R_{ref}}\right)^3\right)^{-1} \quad (5.11)$$

Hvor A_1, B_1, C_1 og D_1 er andre konstanter enn de nevnt foran. Konstantene finnes i databladet og man velge riktig i henhold til hvilken motstandsverdi NTC-motstanden har. Man må derfor først gå inn i tabellen og finne riktig NTC-motstand for deretter å finne riktige parametere.

ELECTRICAL DATA AND ORDERING INFORMATION

R_{25}	$B_{25/85}$-VALUE		UL APPROVED	SAP MATERIAL NUMBER	OLD 12NC CODE	COLOR CODE [3]		
(Ω)	(K)	(± %)	(Y/N)	NTCLE100E3....B0/T1/T2 [2]	2381 640 3/4/6.... [1]	I	II	III
470	3560	1.5	Y	471*B0	*471	Yellow	Violet	Brown
680	3560	1.5	Y	681*B0	*681	Blue	Grey	Brown
1000	3528	0.5	Y	102*B0	*102	Brown	Black	Red
1500	3528	0.5	Y	152*B0	*152	Brown	Green	Red
2000	3528	0.5	Y	202*B0	*202	Red	Black	Red
2200	3977	0.75	Y	222*B0	*222	Red	Red	Red
2700	3977	0.75	Y	272*B0	*272	Red	violet	Red
3300	3977	0.75	Y	332*B0	*332	Orange	Orange	Red
4700	3977	0.75	Y	472*B0	*472	Yellow	Violet	Red
5000	3977	0.75	Y	502*B0	*502	Green	Black	Red
6800	3977	0.75	Y	682*B0	*682	Blue	Grey	Red
10 000	3977	0.75	Y	103*B0	*103	Brown	Black	Orange

Siden vår NTC-motstand har verdien 10kΩ ser vi at det valgte materialet har en $B_{25/85}$-verdi på 3977K. Denne verdien bruker vi så i den neste tabellen hvor vi finner verdiene for konstantene A_1, B_1, C_1 og D_1.

8. Datablad for NTCLE201E3103SB: https://www.vishay.com/docs/29049/ntcle100.pdf

PARAMETER FOR DETERMINING NOMINAL RESISTANCE VALUES											
NUMBER	$B_{25/85}$ (K)	NAME	TOL. B (%)	A	B (K)	C (K^2)	D (K^3)	A_1	B_1 (K^{-1})	C_1 (K^{-2})	D_1 (K^{-3})
1	2880	Mat O. with Bn = 2880K	3	- 9.094	2251.74	229098	- 2.744820E+07	3.354016E-03	3.495020E-04	2.095959E-06	4.260615E-07
2	2990	Mat P. with Bn = 3990K	3	- 10.2296	2887.62	132336	- 2.502510E+07	3.354016E-03	3.415560E-04	4.955455E-06	4.364236E-07
3	3041	Mat Q. with Bn = 3041K	3	- 11.1334	3658.73	- 102895	5.166520E+05	3.354016E-03	3.349290E-04	3.683843E-06	7.050455E-07
4	3136	Mat R. with Bn = 3136K	3	- 12.4493	4702.74	- 402687	3.196830E+07	3.354016E-03	3.243880E-04	2.658012E-06	- 2.701560E-07
5	3390	Mat S. with Bn = 3390K	3	- 12.6814	4391.97	- 232807	1.509643E+07	3.354016E-03	2.993410E-04	2.135133E-06	- 5.672000E-09
6	3528 (1)	Mat I. with Bn = 3528K	0.5	- 12.0596	3687.667	- 7617.13	- 5.914730E+06	3.354016E-03	2.909670E-04	1.632136E-06	7.192200E-08
	3528 (2)			- 21.0704	11903.95	- 2504699	2.470338E+08	3.354016E-03	2.933908E-04	3.494314E-06	- 7.712690E-07
7	3560	Mat H. with Bn = 3560K	1.5	- 13.0723	4190.574	- 47158.4	- 1.199256E+07	3.354016E-03	2.884193E-04	4.118032E-06	1.786790E-07
8	3740	Mat B. with Bn = 3740K	2	- 13.8973	4557.725	- 98275	- 7.522357E+06	3.354016E-03	2.744032E-04	3.666944E-06	1.375492E-07
9	3977	Mat A. with Bn =3977K	0.75	- 14.6337	4791.842	- 115334	- 3.730535E+06	3.354016E-03	2.569850E-04	2.620131E-06	6.383091E-08

Her finner vi følgende verdier for konstantene (nederste linje):
$A_1 = 3{,}354016 \times 10^{-03}$
$B_1 = 2{,}569850 \times 10^{-04}$
$C_1 = 2{,}620131 \times 10^{-06}$
$D_1 = 6{,}383091 \times 10^{-08}$

Ved å sette disse inn i ligning 5.11 har vi et uttrykk for temperaturen.

Siden Arduino gir oss en avlest spenning, U_S, må vi beregne R_{NTC} når vi kjenner seriemotstanden, R_S, og spenningene, $U_{CC} = 3{,}3\text{V}$, som brukes.

$$R_{NTC} = \frac{U_S}{U_{CC} + U_S} R_S \tag{5.12}$$

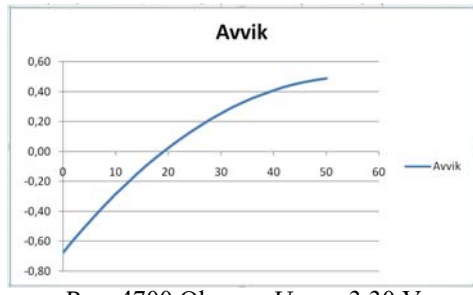
Ved å sette ligning 5.12 (R_{NTC}) inn i ligning 5.11 har vi et absolutt uttrykk for temperaturen i Kelvin.

Under vist funksjonen som beregner absolutt temperatur i testprogrammet *Cansat_TestCB*

```
double read_temp_direct() {
  double R_NTC, log_NTC;
  uint16_t ARead = analogRead(A10);
  R_NTC = 4700*ARead/(4095.0-ARead);
  log_NTC = log(R_NTC/10000);
  return 1/(_A1 + _B1*log_NTC + _C1*log_NTC*log_NTC +
_D1*log_NTC*log_NTC*log_NTC)-273.15;
}
```

Det kan bemerkes at notasjonen `log()` angir naturlig logaritme (ln()).

Simuleringer viser at avvikene fra ønsket verdi er små dersom vi benytter eksakte verdier for R_S og U_{CC} ser vi at avviket er mindre enn ± 0,7°C i temperaturområdet fra 0° – 50°C.

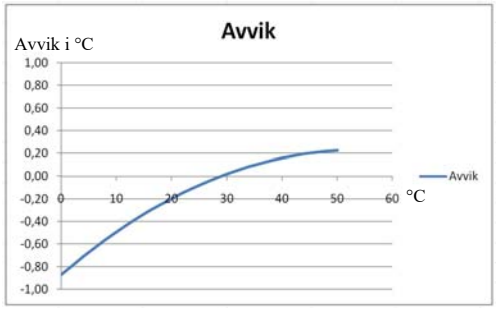


R_S = 4700 Ohm og U_{CC} = 3,30 V

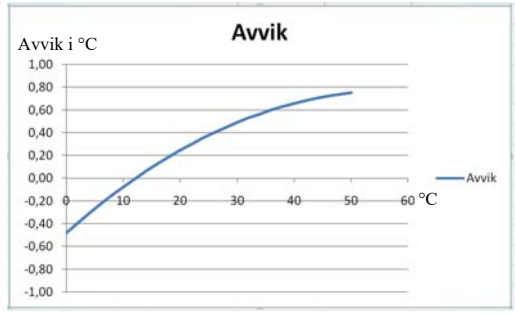
5.2.7 Kalibrering av temperatursensoren når man bruker interpolasjons formel

Normalt skulle denne formelen gi en rimelig nøyaktig verdi uten kalibrering som vist foran. Imidlertid inngår både seriemotstanden R_S og spenningen U_{CC} i omregningsfomelen fra endring i motstand til endring i spenning, og begge disse kan ha mindre avvik i forhold til den verdien som brukes i beregningene.

R_S er en motstand med toleranse ±1%, hvilket betyr at avviket i vårt tilfelle kan være typisk ±47Ω, hvilket ikke er mye. Figuren under viser at avviket er svært liten innen toleransene for seriemotstanden.

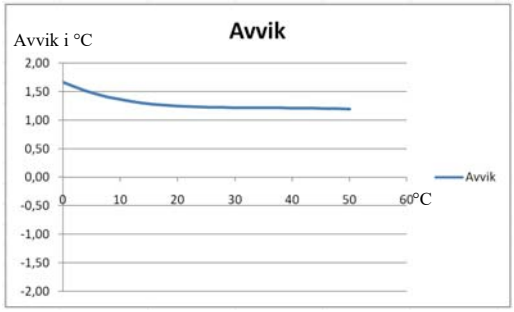


R_S = 4700 – 47Ohm

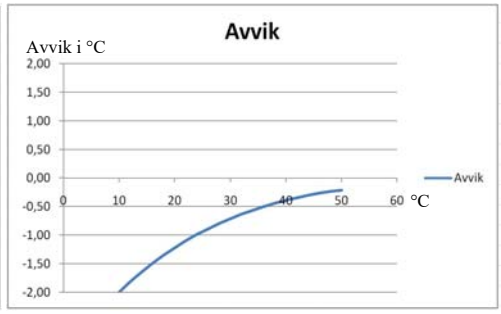


R_S = 4700 + 47Ohm

Vi kan gjøre tilsvarende beregninger med variasjoner i forskyningsspenningen U_{CC}.



U_{CC} = 3,3 – 0,05V



U_{CC} = 3,3 + 0,05V

Som det framgår av figuren over kan avviket som skyldes avvik i supplyspenning være betydelig mer enn ± 2°C innen temperaturområdet 0° – 50°C med en variasjon på ±0,05 V i U_{CC}.

5.2.8 Noen didaktiske refleksjoner

Grunnen til at man har beholdt en NTC-motstand i oppkoblingen er flere. Ved hjelp av en slik sensor er det lett å plassere den på utsiden av CanSat og dermed få en riktigere utvendig luft-temperatur, videre vil den kunne være liten og dermed ha en kort tidsrespons. Den er dessuten lett å bruke og enkel å kalibrere dersom det er et poeng.

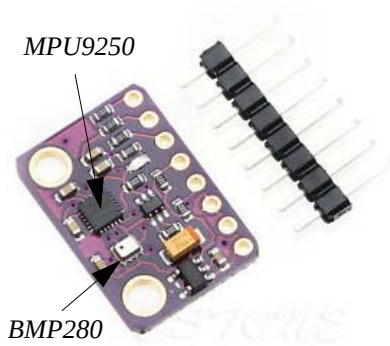
En kan også bruke sensoren til didaktiske formål ved at man kan la elevene studere lineariteten mht. valg av gunstig verdi av seriemotstanden innenfor det aktuelle temperaturområdet. Der ligning 5.11 er en "svart boks" kan en tilnærming som omtalt i avsnitt 5.2.1 – 5.2.5 være langt mer lærerik om enn noe mer tidkrevende. Å simulere sammenhengen mellom temperatur og målt spenning ut fra ligning 5.6 og ligning 5.7, kan også være en nyttig øvelse for mange elever. I så fall vil de kunne studere lineariteten og ev. lage en enkel sammenheng mellom spenning og temperatur ved hjelp av topunktsligningen i det aktuelle temperaturområdet. Denne muligheten forsvinner dersom man benytter ligning 5.11. Dog har vi sett at også denne bør kalibreres.

5.3 IMU - GY91

Dette er et lite kretskort som inneholder svært mange funksjoner og går under betegnelsen IMU (Inertial Measurement Unit). GY91 inneholder følgende sensorer:

BMP280 – Barometer og temperatur, nøyaktighet omregnet i høydemeter: +/- 1 m

MPU9250 – Kombinert tre akse, akselerometer, gyrometer og magnetometer. Egentlig består kretsen av to chiper, en for gyrometeret og akselerometeret, og en for magnetometeret (AK8963)



Den har derfor mange av de samme funksjonene som GY80 og 87, men er langt mer kompakt.

Alle funksjonene kan nås via I^2C bussen.

5.3.1 Spesifikasjoner for GY-91

Kort oppsummert er spesifikasjonene som følger:

- Supply spenning: 3 – 5V
- Kommunikasjon: Standard I^2C/SPI kommunikasjonsprotokoll
- Innebygget 16 bit ADC, dvs 16 bits datautgang
- Gyrometer områder (x, y, z): +/– 250/500/1000/2000 grader/s (dps)
- Akselerometer områder (x, y, z): +/– 2, 4, 8 og16 g
- Magnetometer område (x, y, z): +/– 4800µT
- Lufttrykk: 300 – 1100hPa

- Fysisk størrelse: 14.3mm x 20.5 mm

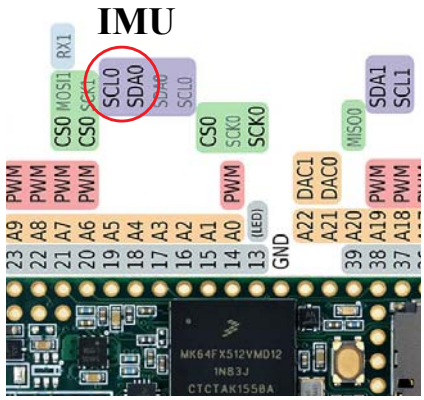
Kretsskjemaet for kortet finnes i vedlegg A.2.

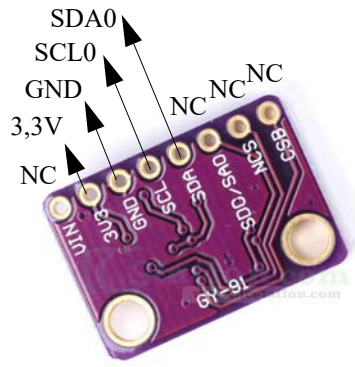
Leverandør:

www.Banggood.com selger den for kr. 67,74 med gratis leveranse. Det kan imidlertid ta litt tid[9].

5.3.2 GY91 integrert del av CanSat-kortet

GY-91 kommuniserer med Teensy 3.5 via I^2C buss SCL0 (p40) og SDA0 (41) i tillegg til GND of 3,3 V.





De øvrige pinnene kan man la være å tilkoble.

Hver av de fire sensorene er koblet på I^2C bussen og kan nås med ulike adresser.

5.3.3 Akselerometer – MPU9250/55

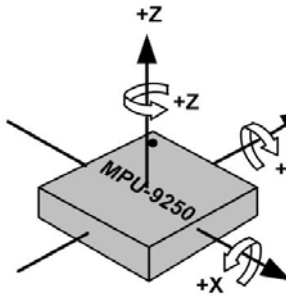
Kretsbeskrivelse:

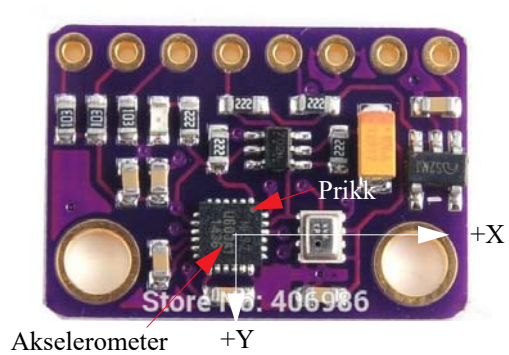
Akselerometer delen av MPU9250 måler akselerasjon i x-, y- og z-retning med en oppløsning på 16 bit (eller egentlig kraft som resultat av akselerasjon[10]).

En kan velge å måle ±2 g (+/–2 048), ±4 g (+/–4 096), ±8 g (+/–8 192 bit) og ±16 g (+/–16 384). Verdiene angis i 2'ers komplement. Målebåndbredden er opp til 260 Hz (høyeste knekkfrekvens for lavpass filteret), som forteller oss noe om hvor raskt kretsen kan registrere endringer i akselerasjonen.

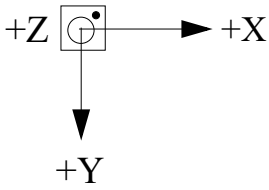
9. https://www.banggood.com
10. Ref. Skule Sørmo

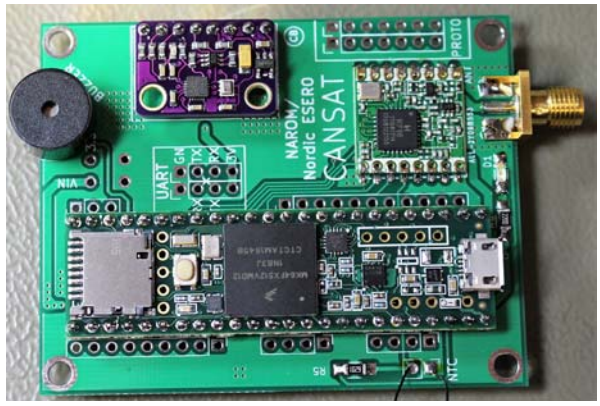
Figuren under viser orienteringen av akselerometeret som kan være til hjelp under kalibreringen.





Figuren under angir plasseringen av retningene på CanSat-kortet.





Figuren over definerer retningene for positive verdier for tyngdekrafta i x-, y-, og z-retning. Legg merke til at når kretsen ligger horisontalt er positiv z-retning rett opp, dvs. tyngdeakselerasjonen vil bli negativ lik –1g når kretsen ligger flatt på bordet med oppsiden opp.

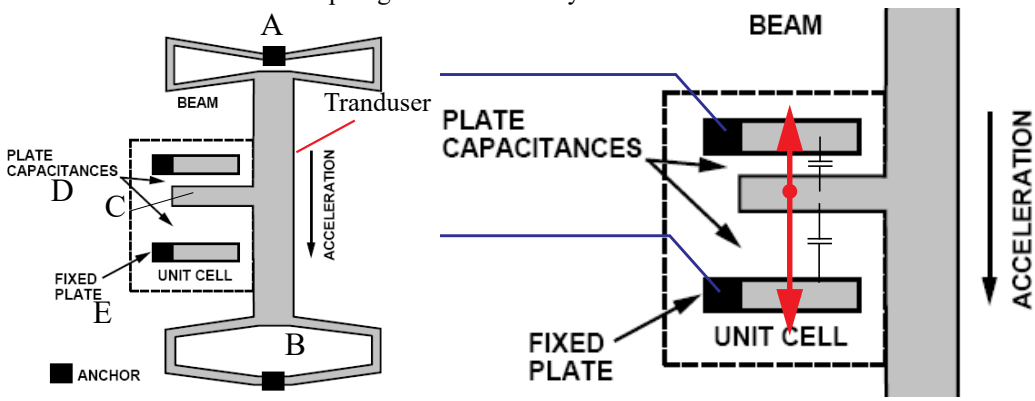
Enkel kalibrering kan gjøres ved å holde kretsen i ulike retninger i forhold til jordgravitasjonen. Da vil man sannsynligvis oppdage at det er store avvik. Disse avvikene kan lett korrigeres ved å legge til eller trekke fra avviksverdier i programmet (mer om dette under kalibrering side 49). Det er rimelig å anta at skaleringen er riktig.

På bakgrunn av dette så er det forståelig at GY91 bør orienteres parallelt med CanSat-kortet ved motering.

Teknologi

Prinsippet bak målingen av akselerasjonen baserer seg på massens treghet. I akselererte systemer vil massen påvirkes av krefter som igjen medfører en forskyvning som kan måles.

Ved hjelp av en etseteknikker lages det *transducere* som henger fritt mellom to punkter A og B (se figuren under). Transduseren er laget ved å legge et lag med polysilisiumstruktur med ønsket form (se figuren under) på toppen av et lag med silisiumoksid. Oksidlaget på undersiden etses bort slik at transduseren blir hengende fritt. En flik C av transduseren kan bevege seg mellom to faste kondensatorplater (D og E). Når akselerometeret utsettes for en akselerasjon i lengderetningen til transduseren, vil den påvirkes av krefter siden den har treghet (masse). Fliken C vil dermed forskyves i forhold til de to faste metallplatene D. Flikene C og D vil ikke berøre hverandre, men danne kondensatorer som vist på figuren under til høyre.



Figur 5.3 Prinsippskisse av akselerometerets kapasitive sensor.

Når fliken beveger seg endres de kapasitive verdiene, en endringen som genererer et signal som kan relateres til akselerasjonen.

Kretsen måler statisk akselerasjon og vil dermed også måle tyngdeakselerasjonen *g*, og vil kunne fungere som tilt-sensor (måler helningsvinkelen). Dvs. den kan registrere hvordan de ulike sensorene heller i forhold til retningen til tyngdefeltet. Det kan i enkelte tilfeller være nyttig å bruke informasjon fra en slik sensor for å studere hvordan en CanSat er orientert når den faller gjennom atmosfæren, men ofte vil måleresultatene bli ganske kaotiske og vanskelige å tolke.

Kalibrering – akselerometer:

Alle sensorer bør kalibreres. Uansett er det en nyttig øvelse for å få en forståelse for hva kalibrering er og hvorfor man gjør det.

En hver kalibrering krever at man har en kjent standard som kan kalibrere opp mot. Kalibreringen blir ikke mer nøyaktig enn nøyaktigheten til standarden. Når det gjelder kalibrering av akselerometeret så benytter vi den allestedsnærværende gravitasjonskrafta. Vi definerer at stedets tyngdeakselerasjon er 1,0 g. Dersom akselerometeret avviker fra denne så korrigerer vi verdien slik at den kommer nærmest mulig 1,0 g.

Ved kalibrering går du fram som følger:

1. Koble opp CanSat-kortet, legg inn testprogrammet og sørg for at programmet skriver ut verdiene for akselerasjon i x-, y- og z-retning.

2. Hold CanSat-kortet slik at GY91 er orientert som vist på figur A over. Det viktig at kortet er orientert slik at x-retningen er mest mulig loddrett. Bruk gjerne et vater.
3. Åpne monitoren i IDE'en og les av akselerometerverdien (ax) i x-retning.
4. Gå inn i programmet og trekk fra verdien av avviket slik at resultatet blir nærmest mulig lik ax = 1,0 g
5. Drei kretsen 90° som vist i figur B til høyre.
6. Les av akselerometerverdien (ay) i y-retning.
7. Gå inn i programmet og trekk fra verdien av avviket slik at resultatet blir nærmest mulig lik ay = 1,0 g

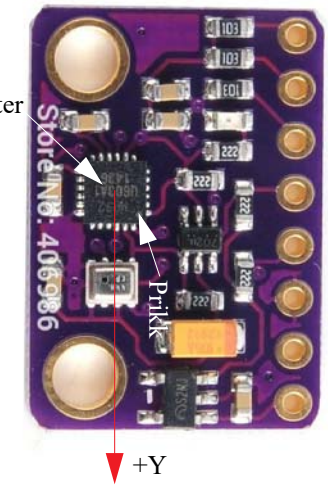
A)

Akselerometer +X

ax = 1,0 g
ay = 0,0 g
az = 0,0 g

B)

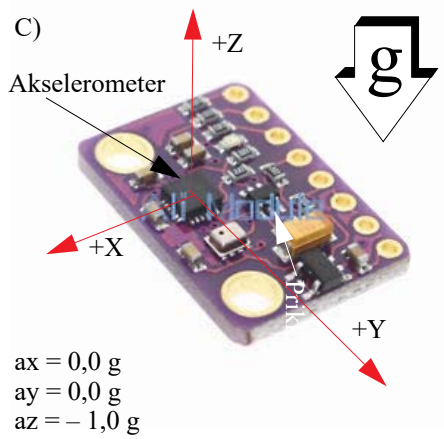
Akselerometer



ax = 0,0 g
ay = 1,0 g
az = 0,0 g

+Y

8. Legg CanSat-kortet flatt på bordet med komponentsiden opp som vist på figur C) til høyre.
9. Les av akselerometerverdien (az) i z-retning. Siden positiv z-retning er rett opp og tyngde akselerasjonen peker rett ned så vil den avleste verdien være i nærheten av –1,0 g.
10. Gå inn i programmet og trekk fra verdien av avviket slik at resultatet blir nærmest mulig lik az = 1,0 g
11. Programkoden vil kunne se ut omtrent som vist under:




```
ax = gy91.ax + avvik_x;
ay = gy91.ay + avvik_y;
az = gy91.az – avvik_z;
```

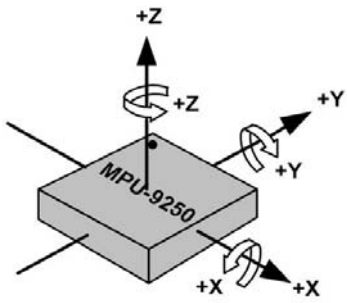
Fortegnet foran avviket er kanskje ikke så viktig. Det viktigste er at resultatet blir at hver av akselerometerets verdier i x-, y- og z-retning blir nærmest mulig 0,0 g når kortet holdes i de tre respektive posisjonene.

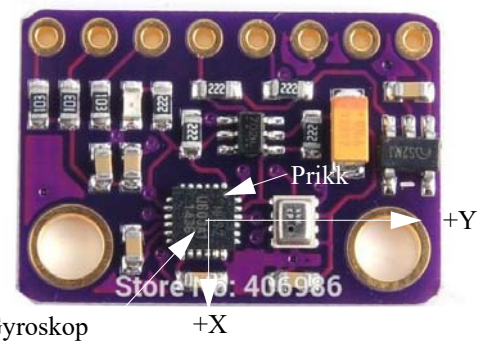
5.3.4 Gyroskop – MPU9250/55

Kretsbeskrivelse:

Kretsen har sensorer som registrerer rotasjon i alle tre retninger. Rotasjonen angis med et heltall med fortegn i grader pr. sekund (dps). Det kan også oppgis som omdreininger pr. sekund (rps). 360° pr. sek. (360 dps) vil i så fall bli 1 rotasjon pr. sek. (1 rpm). Sensoren består av tre strukturer som settes til å vibrere i x-, y- og z-planet, når kretsen roteres i forhold til disse tre planene vil det skapes krefter (forflytninger) som kan registreres og omregnes til vinkelhastigheter omkring de tre retningene x, y og z.

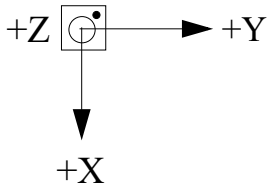
Figuren under viser orienteringen av gyroskopet som kan være til hjelp under kalibreringen.

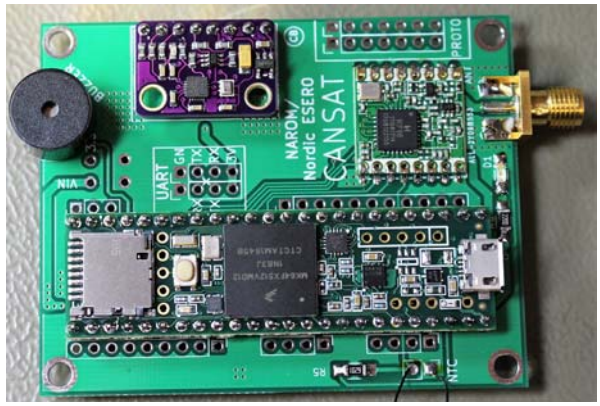




Gyroskop

Figuren under angir plasseringen av retningene på CanSat-kortet.





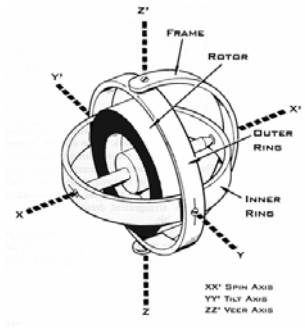
Figuren over viser hvordan rotasjonsretningene er definert i forhold til kretsen (GY91).

Avlesning av verdiene:

Kretsen gir mulighet for å velge ett av fire måleområder (FS – Full skala):
FS = +/– 250 dps med en oppløsning på 7,6 mdps[11]/enhet (milligrader pr. sekund pr. enhet)
FS = +/– 500 dps med en oppløsning på 15,3 mdps/enhet
FS = +/– 1000 dps med en oppløsning på 30,5 mdps/enhet
FS = +/– 2000 dps med en oppløsning på 61,0 mdps/enhet
Det betyr at dersom gyroen gir ut en verdi på f.eks. 425 og gyroen er satt opp til FS = 2000 dps, så tilsvarer dette en rotasjonsfart på 0,061 dps/enhet x 425 enheter = 25,9 dps.

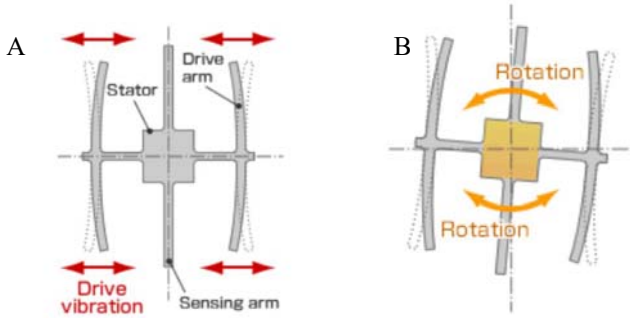
Teknologi

De fleste forbinder gyroskoper med roterende hjul som har en ekstra tyngde langs periferien. Når hjulet settes i bevegelse (rotasjon) vil man merke krefter dersom man forsøker å vri hjulets rotasjonsplan ut av posisjon. Jo raskere man forsøker å vri rotasjonsplanet, jo større motstand merker man. Ved å måle denne kraften får man et mål for rotasjonshastigheten til hjulet. Imidlertid er det vanskelig å integrere små roterende hjul, med tilstrekkelig masse. Man har imidlertid funnet ut at man oppnår en lignende effekt ved å rotere vibrerende masser.

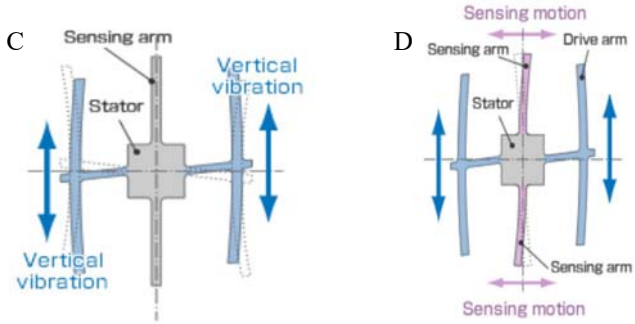


11. mdps - milli degrees pr. sec.

La oss tenke oss en dobbel T-formet struktur festet til et sentralt fast punkt (stator) som vist i figur A under. De T-formede armene kan bestå av et piezoelektrisk materialet som begynner å svinge mekanisk dersom det påføres en oscillerende spenning..



"Statoren" er festet til sensorkretsen som vil bevege seg med utstyret det skal måles på (i vårt tilfelle GY91). Loddrett gjennom statoren står en piezoelektrisk arm som registrerer vriding. Når så utstyret med sensorkretsen påføres en rotasjon om en akse loddrett på papirplanet, som vist på figur B, vil den vertikale armen i midten registrere en horisontal bevegelse som vist i figur D. Siden armen er piezoelektrisk vil den mekaniske svingingen i armen resultere i en elektrisk spenning som kan måles.



Størrelsen på denne vridingen er et direkte mål på rotasjonshastigheten til utstyret, omkring en akse loddrett på papirplanet.

Strukturen miniatyriseres og det monteres tre stykker plassert loddrett på hverandre i en integrert krets. Dermed vil kretsen være istand til å måle rotasjon omkring alle tre akse-retninger.

Dette eksempelet på et integrert gyroskop er hentet fra EPSON's hjemmside.[12]

12.https://www5.epsondevice.com/en/information/technical_info/gyro/

Det er også interessant å vite at denne teknikken er benyttet av insekter, sannsynligvis gjennom millioner av år. På norsk kalles denne innretningen hos enkelte insekter for *svingkølle* og skal hjelpe insektet til å føle dreining av kroppen og slik at insektet blir istand til å stabilisere flukten. Vi leser fra Wikipedia:

Svingkøller eller halterer er et par vedheng man finner på forkroppen (thorax) hos tovinger (mygg og fluer). Disse kølleformede organene er egentlig et redusert andre (bakre) vingepar og har rester av et årenett. Når insekter flyr, svinger de opp og ned i stor hastighet, og det er ofte svingkøllene, ikke vingene, som lager den høyfrekvente summelyden når fluer og mygg flyr. Man antar at svingkøllene kan fungere som et balanseorgan, en gyro. Svingkøller finnes også hos gruppen viftevinger (Strepsiptera), men her er de dannet fra det første (fremre), ikke det andre vingeparet.[13]

Kalibrering – Gyrometeret

Kalibrering av gyrometeret er ikke så enkelt som for akselerometeret da vi ikke har åpenbar kjent standard tilgjengelig. Dvs. vi trenger noe som roterer med en konstant og kjent rotasjonshastighet.

Et nærliggende valg er å bruke en grammofon dersom man er så heldig å ha en slik. Hastigheten til en grammofon skal kunne stilles inn med stor nøyaktighet for at musikken får en mest mulig naturtro gjengivelse.

Det var nettopp det elevene fra Silkeborg skole gjorde da de deltok på den nordiske CanSat-konkurransen på Andøya april 2015. De brukte en grammofon som hadde tre hastigheter 33, 45 og 78 omdreininger pr. minutt. Bildet til høyre viser måleoppstillingen. Selve CanSat-en er plassert på en plattform slik at selve gyroen står mest mulig sentrert på tallerkenen. Vi ser også batteriet. Skal det være mulig å gjøre disse målingene i fart må man enten lagra dataene ombord i CanSat-en eller overføre dem via radio. CanSat gir jo mulighet til begge deler.

13.https://no.wikipedia.org/wiki/Svingkølle

Kjenner vi hastigheten til grammofonen er det ikke vanskelig å beregne antall grader pr sekund (198, 270 og 468). Imidlertid valgte de å kontrollmåle hastigheten ved å ta tiden det tok for platen å gå 10 omdreininger, dvs. den gjennomløper 3 600°. Resultatet viser noe avvik fra den teoretisk riktige hastigheten. Resultatene er vist i tabellen under.

"Speed" (turns per minute)	Measured time for 10 turns (sec)	angular speed (degree per sec)	16 bit digital output			angular speed calculated from sensor output (degree per sec)		
			x-gyro	y-gyro	z-gyro	x	y	z
33	17,2	209	3030	3060	3080	185	187	188
45	12,5	288	4120	4180	4200	251	255	256
78	7,2	500	7200	7300	7350	439	446	449

Dernest har de lest av verdien fra AD konverteren som i følge tabellen har 16 bit. Siden gyroen skal kunne levere både positive og negative rotasjonsdata vil oppløsningen være på 15 bit dvs. verdier i området ± 32 767. Dette området utgjør i dette tilfelle ca. ± 5,6 rps (omdreininger pr. sekund) eller ca. ± 2000 dps (grader pr. sekund).

Av resultatene så kan det se ut til at de målte verdiene ligger noe lavere enn de beregnede verdiene. Vi legger også merke til at avviket mellom målt hastighet og gyroens registrerte vinkelhastighet er økende med økende vinkelhastighet (33rpm, 209 – 187 = 22 dps, 45 rpm, 288 – 254 = 24 dps og 78 rpm, 500 – 447 = 53 dps). Avviket blir imidlertid noe mindre dersom vi sammenligner måleresultatene med de oppgitte hastighetene til grammofonen.

Vårt gyroskop har også 16 bit oppløsning dvs. ± 32 767 mulige verdier. I tillegg kan vi velge mellom følgende områder +/– 250/500/1000/2000 dps. Har vi en grammofon kan vi gjennomføre kalibreringen på samme måte som våre venner fra Silkeborg. Stoler vi på grammofonens hastigheter må vi, dersom vi får et tilsvarende resultat, legge inn korreksjoner. Siden feilen øker med hastigheten så må vi vurdere å legge inn en multiplikasjonsfaktor og ikke bare legge til eller trekke fra en konstant verdi.

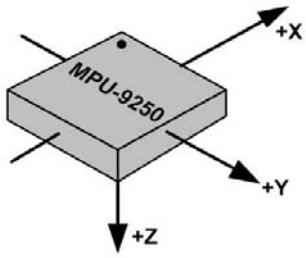
5.3.5 Magnetometer (kompass)

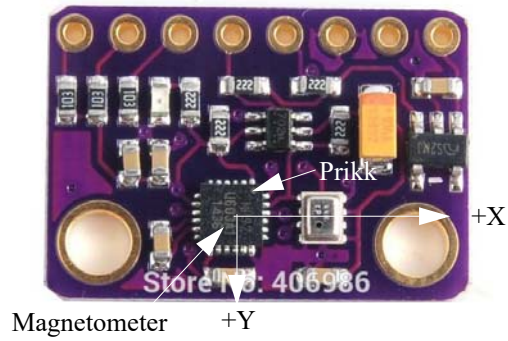
Kretsbeskrivelse:

Magnetometeret er en selvstendig chip montert i samme hus som akselerometeret og gyroskopet og er opprinnelig designet av firmaet Asahi Kasai og går som selvstendig krets under betegnelsen AK 8963. Den er laget for å kunne registrere svake magnetfelt som f.eks. jordmagnetfeltet. Kretsen egner seg derfor som kompass. Selve sensoren er en type Hall-sensor. Sensoren er naturlig nok retningsfølsom slik at feltkomponenten i x-, y- og z-retning lar seg registrere individuelt. Kretsen har en 14/16 bit ADC (analog til digital konverter) som medfører en følsomhet på inntil 0,6 µT/LSB[14] (Least Significant Bit) for 14 bit, og 0,15 µT/LSB for 16 bit. Det dynamiske spennet for kretsen er fra 0,15 µT (16 bit) til 4900µT i to områder.

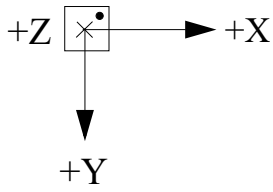
14.1 Tesla = 10 000 Gauss

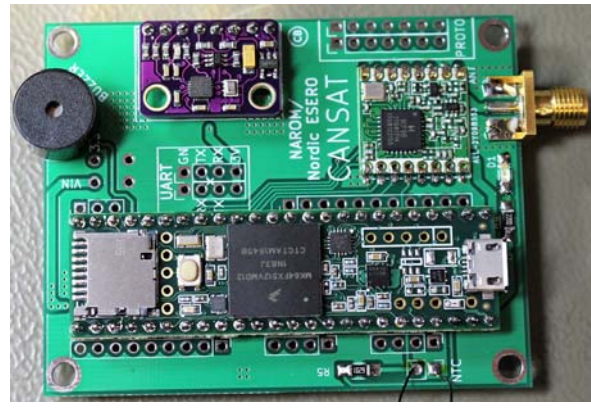
Figuren under viser orienteringen av magnetometeret som kan være til hjelp under kalibreringen.





Figuren under angir plasseringen av retningene på CanSat-kortet.





Figuren over viser hvordan kompassretningene er definert i forhold til kretsen. Legg spesielt merke til at retningene er dreid i forhold til akselerometeret og gyroskopet.

Jordmagnetfeltet har verdier i området 25 – 65 µTesla. Dermed vil en følsomhet på inntil 150 nT være mer enn bra nok for å fungere som kompass (1 Tesla = 10 000 Gauss).

I Trondheim har jordmagnetfeltet følgende komponenter:

- Helningsvinkel med jordoverflata: 74° 47' 45"
- Absoluttverdi langs feltlinjene: 51 881,8 nT
- Projeksjon langs jordoverflata nordlig retning: 13 582.6 nT
- Projeksjon langs jordoverflata i østlig retning: 804.7 nT

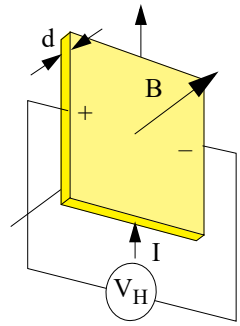
Dataene er hentet fra nettstedet: https://www.ngdc.noaa.gov/geomag-web/#igrfwmm

NOAA - National center for environmental information (NOAA, National Oceanic and Atmospheric Administration).

Vi ser her at siden følsomheten ligger på 150 – 600 nT så burde det være mulig å bruke dette som en hjelp til kalibrerering spesielt dersom man legger magnetometeret i feltretning på 74° 47' 45". Dette kommer vi tilbake om litt.

Teknologi

Halleffekten ble oppdaget i 1879 av ***Edwin Hall*** *(1855–1938)*. Han observerte at det oppsto en elektrisk spenning i et strømførende materiale dersom det ble utsatt for et magnetfelt (se figuren til høyre). Hall oppdaget effekten da han arbeidet med sitt doktorarbeid ved *Johns Hopkins university* i Baltimore i Maryland, USA.



Effekten er en naturlig følge av Lorentz-kraften på ladninger som beveger seg i et magnetfelt. Et magnetfelt som står loddrett på strømretningen, vil påvirke ladningen med en kraft som står normalt på strømretningen og på magnetfeltet. Dermed vil det oppstå en liten, men målbar ladningsforskjell mellom sidekantene til den elektriske lederen. Dette er vist på figuren til høyre.

Når det er oppnådd en balanse mellom det elektriske feltet på grunn av ladningsforflytningen og magnetfeltet, vil forflytningen av ladningen opphøre (dvs. at tilflytende ladning er lik ladninger som forlater området) og ladningsforskjellen stabiliseres. Ved å måle styrken til denne spenningen kan man få en verdi for magnetfeltets styrke.

Kalibrering av magnetometeret

Et magnetometer vil måle verdien av jordmagnetfeltet i x-, y- og z-retning og kan således brukes som et kompass. I forbindelse med styringssystemer (f.eks. droner) kan et magnetometer brukes som et hjelpemiddel for å korrigere for drift i gyroen.

Avvik kan skyldes flere ting og ikke alle avvik kan korrigeres. De avvik som kan korrigeres skyldes nærliggende gjenstander som gir en konstant påvirkning av magnetfeltet rundt sensoren. I en mobiltelefon kan dette være f.eks. høyttaleren i telefonen som gjerne kan inneholde en magnet dersom den er elektrodynamisk. Denne typen kilder kalles gjerne “hard iron” og er relativt lett å korrigere. I tillegg kan det være metaller som selv ikke er magnetiske, men som lar seg magnetisere og som dermed vil påvirke feltbildet rundt sensoren. Disse går gjerne under navnet “soft iron” og er mer krevende å kalibrere bort. Uansett kan konsekvensene av begge disse reduseres med kalibrering.

Dernest vil det finnes kilder som kommer og går avhengig av omgivelsene. Slike kilder vil ikke kunne korrigeres ved en kalibrering.

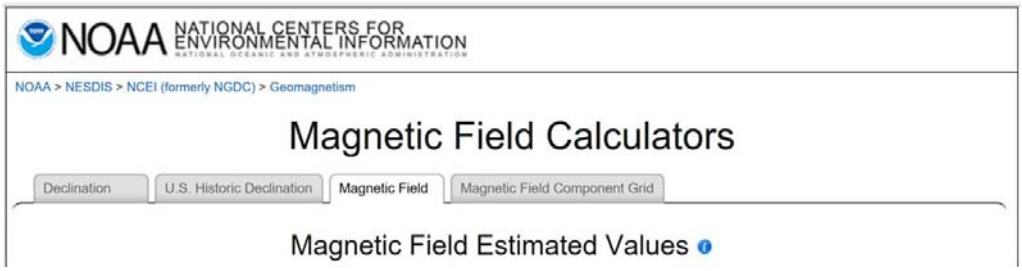
Normalt vil vi, for å kalibrere magnetometeret, enten trenge en:

- ... en kalibrert kilde som vi kjenner styrken og retningen til i en gitt avstand.
- ... et kalibrert måleinstrument slik at vi kan sammenligne målingene utført av vårt magnometer med hva det kalibrerte måleinstrumentet måler.

Normalt er ingen av disse to instrumentene tilgjengelig, ev. er de for dyre for å anskaffe.

Imidlertid har vi en magnetisk kilde som er relativt stabil, jordmagnetismen. Dersom vi kjenner den nøyaktige styrken og retningen av denne der vi befinner oss, ville vi ha en kalibrert kilde som vi kan sammenligne målingene våre med.

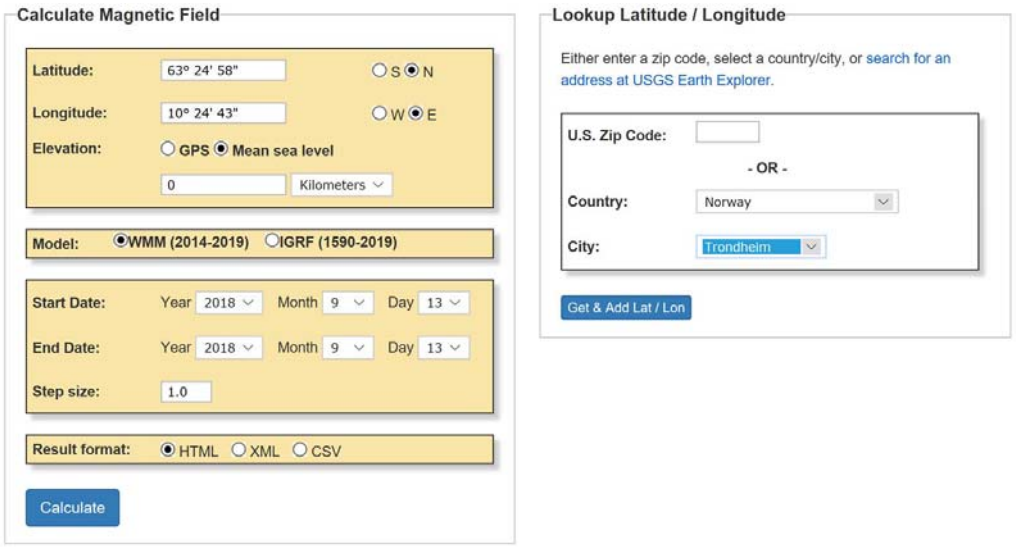
Som vi har antydet foran så finnes det et nettsted som med stor nøyaktighet angir den magnetiske feltstyrken på et hvilket som helst sted på jorda: https://www.ngdc.noaa.gov/geomag-web/#igrfwmm



Dette nettstedet gir et vell av informasjon om jordens magnetfelt.

La oss se nærmere på fanen: *Magnetic Field* i bildet over.

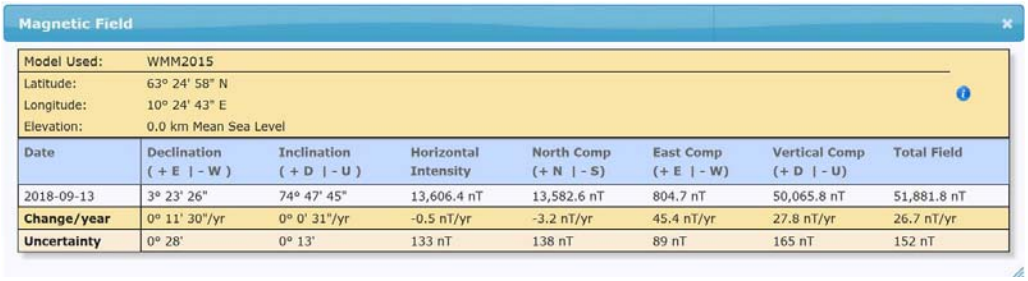
Da får vi opp følgende meny:



- I menyen til høyre velger vi land og sted. La oss velge *Norge* og *Trondheim*.
- Trykk deretter: *Get & Add Lat/Lon*. Dermed settes Trondheims langde- og breddegrad inn i skjemaet til venstre.
- Sett *Elevasjon* til *Sea level* (dvs. hoh. lik 0 m)
- Velg så dato f.eks. *Start Date: 2018.09.13* og *End Date: 2018.09.13*
- Velg deretter *HTML*

- Og til slutt *Calculate*

Dermed vil vi få en detaljert oversikt over de jordmagnetiske forholdene i Trondheim på den aktuelle datoen.



Magnetic Field

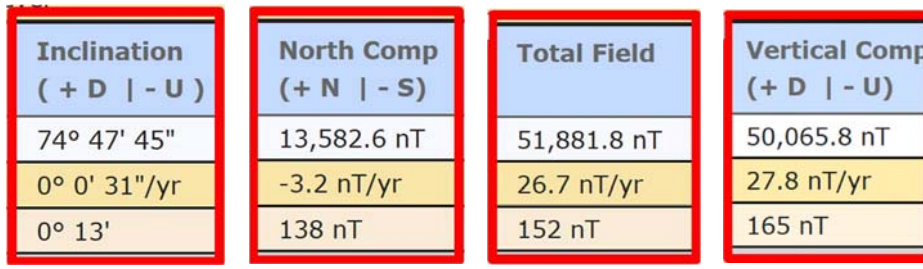
Model Used:	WMM2015
Latitude:	63° 24' 58" N
Longitude:	10° 24' 43" E
Elevation:	0.0 km Mean Sea Level

Date	Declination (+ E | - W)	Inclination (+ D | - U)	Horizontal Intensity	North Comp (+ N | - S)	East Comp (+ E | - W)	Vertical Comp (+ D | - U)	Total Field
2018-09-13	3° 23' 26"	74° 47' 45"	13,606.4 nT	13,582.6 nT	804.7 nT	50,065.8 nT	51,881.8 nT
Change/year	0° 11' 30"/yr	0° 0' 31"/yr	-0.5 nT/yr	-3.2 nT/yr	45.4 nT/yr	27.8 nT/yr	26.7 nT/yr
Uncertainty	0° 28'	0° 13'	133 nT	138 nT	89 nT	165 nT	152 nT

Her er vi spesielt interessert i

- *Inclination* = 74° 47’ 45” (vinkelen til magnetfeltet som treffer jorda i Trondheim) og
- *North Comp* = 13 582,6 nT (styrken til feltkomponenten mot geografisk nord) og
- *Total Field* = 51 881,8 nT (styrken til feltkomponenten langs feltlinjene i Trondheim)
- *Vertical Comp* = 50 065,8 nT (styrken til den loddrette feltkomponenten i Trondheim)

Disse er forstørret opp og vist på figuren under.



Inclination (+ D | - U)	North Comp (+ N | - S)	Total Field	Vertical Comp (+ D | - U)
74° 47' 45"	13,582.6 nT	51,881.8 nT	50,065.8 nT
0° 0' 31"/yr	-3.2 nT/yr	26.7 nT/yr	27.8 nT/yr
0° 13'	138 nT	152 nT	165 nT

Dersom disse tallene er riktige så har vi en “kalibrert” kilde.

Det neste vi må finne ut av er hvilket format dataene som sendes over databussen har. I tabellen[15] til høyre ser vi at kretsen kan settes til å operere med 14 eller 16 bit. La oss anta at vi bruker 14 bit. Vi vet også at ett av bit’ene (det mest signifikante) er et fortegnsbit. Så da har vi 13 bit til å overføre selve tallet. 2^{13} gir oss 8192 verdier, som betyr at

Measurement data (each axis) [15:0]			Magnetic flux density [μT]
Two's complement	Hex	Decimal	
14-bit output			
0001 1111 1111 1110	1FFE	8190	4912(max.)
|	|	|	|
0000 0000 0000 0001	0001	1	0.6
0000 0000 0000 0000	0000	0	0
1111 1111 1111 1111	FFFF	-1	-0.6
|	|	|	|
1110 0000 0000 0010	E002	-8190	-4912(min.)
16-bit output			
0111 1111 1111 1000	7FF8	32760	4912(max.)
|	|	|	|
0000 0000 0000 0001	0001	1	0.15
0000 0000 0000 0000	0000	0	0
1111 1111 1111 1111	FFFF	-1	-0.15
|	|	|	|
1000 0000 0000 1000	8008	-32760	-4912(min.)

15. Tabellen er hentet fra databladet til AK8963C:

vi kan måle ± 4096 tilstander. Fra tabellen ser vi at vi kan måle verdier fra ± 0,6µT (LSB) til ± 4912 µT. Det betyr at hvert sprang i tallverdi utgjør: 1,1992 µT eller tilnærmet 1,2 µT. Ved å multiplisere tallet vi mottar, inkludert fortegn, med 1,2 vil vi få den magnetiske feltstyrken i µT.

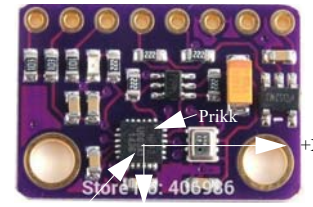
Dersom vi bruker den loddrette verdien av jordmagnetismen som referansenivå, så skulle vi få en tallverdi lik:

$$50{,}0658\mu T/1{,}19921 \approx 41$$

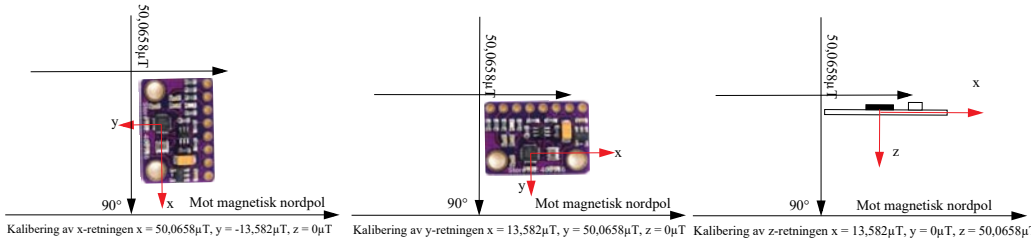
Dersom vi bruker 16 bit burde vi få et fire ganger så høyt tall:

$$50{,}0658\mu T/(1{,}19921/4) = 166$$

Dersom vi snur kretsen opp ned burde vi få det samme tallene med motsatt fortegn.



På figuren under har vi illustrert hvordan dette kan gjøres for x-, y- og z-retning.



Ved å lese av verdien i de tre retningene og legge inn korreksjoner i programmet burde det være mulig å kalibrerer resultatene fra magnetometeret i kretsen. Det forutsettes også at kortet er orientert mot nord som vist på figuren, for at verdiene til de øvrige komponentene skal bli som antydet.

For nærmere studier av kalibrering av magnetometeret se: https://appelsiini.net/2018/calibrate-magnetometer/ eller: https://learn.sparkfun.com/tutorials/mpu-9250-hookup-guide

5.3.6 Trykk- og temperaturmåler – BMP280

Kretsbeskrivelse:

BMP280 er en trykksensor for måling av atmosfærisk lufttrykk. Selve sensoren anvender en piezo-resistiv teknologi, dvs. at med økende lufttrykk vil motstandsverdien i det piezo-resistive materialet endres. Den har dessuten en innebygget temperatursensor som i tillegg til å kunne leses av via I^2C bussen som en vanlig temperatursensor, også kan levere data for omregning fra trykk til høyde. Trykkdata leveres med en oppløsning på 16 – 19 bit, mens temperatur leveres med 16 bit.

Gjennomsnittlig måletid er 5,5 msek, så en kan foreta raske avlesninger med en punkprøvingsrate på 157 sps (samles pr. sec.). Temperaturmålingen kan nøye seg med langt lavere samlingstakt (1 Hz). Økt samlingstakt vil også øke strømforbruket.

Kretsen leverer absolutte trykkmålinger med en nøyaktighet på ±1hPa og en relativt nøyaktighet på ±0,12 hPa (±1m) i området fra 300 hPa – 1100 hPa (fra –500 til 9000 moh.). Oppløsningen er imidlertid 0,16 Pa. Den leverer også temperaturmålinger med en absolutt nøyaktighet på ±0,5°C @ 25°C, ±1,0°C fra 0 – 65°C, men med en oppløsning 0,01°C. Her kan man lett bli lurt ved at man tror at oppløsningen tilsvarer nøyaktigheten.

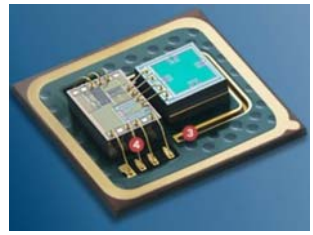
Kretsen kan operere på batterispenninger fra 1,7 V til 3,6 V. Vanlig spenning er 3,3 V. Kretsen kan kobles direkte til mikrokontrolleren via en I^2C buss.

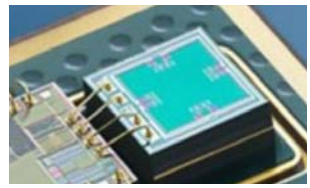
Tabellen under viser en sammenligning mellom BMP180 og BMP280:

Parameter	BMP180	BMP280
Footprint	3.6 × 3.8 mm	2.0 × 2.5 mm
Minimum V_{DD}	1.80 V	1.71 V
Minimum V_{DDIO}	1.62 V	1.20 V
Current consumption @3 Pa RMS noise	12 µA	2.7 µA
RMS Noise	3 Pa	1.3 Pa
Pressure resolution	1 Pa	0.16 Pa
Temperature resolution	0.1°C	0.01°C
Interfaces	I²C	I²C & SPI (3 and 4 wire, mode '00' and '11')
Measurement modes	Only P or T, forced	P&T, forced or periodic
Measurement rate	up to 120 Hz	up to 157 Hz
Filter options	None	Five bandwidths

Teknologi

Teknologien bygger på piezo-resistiv effekt som gjør at resistiviteten i en tynn skive (figur nederst til høyre) av piezo-resistivt materiale endrer resistivitet når den utsettes for trykk og dermed nedbøyning. Denne endringen i resistivitet vil så omdannes til en spenning i en målebro slik at de små endringene i resistivitet medfører endringer i en tilsvarende elektrisk spenning. For å øke følsomheten i målingen kobles de piezo-resistive elementene i skiva inn i en elektrisk målebro.





Ved måling av absolutt lufttrykk legges den tynne plata over et vakuumkammer slik at alle målinger blir gjort relativt til vakuum.

Måling av høyde basert på trykkmålinger

Trykk måles normalt i Pascal hvor 1 Pa = 1 N/m^2.

Tidligere ble trykk målt i atmosfærer (atm), mmHg eller Bar.

En normalverdi for lufttrykket er:

1 atm = 760 mmHg = 1.01325 Bar = 1013.25 mBar = 101325 Pa = 1013.25 hPa

Vi legger merke til at h(ekto)Pa er det samme som m(illi)bar.

Lufttrykket er bestemt av tyngden til det "havet" av luft som vi befinner oss på bunnen av. Lufttrykket er derfor avhengig av den mengden av luft som til en hver tid befinner seg over hodet på oss. Vekta av luftmengden er avhengig av tyngdekraften, tykkelsen på luftlaget og tettheten, som igjen er avhengig av hvordan lufta forflytter seg og av temperaturen, dvs. værforholdene. Som vi ser er det mange faktorer å ta hensyn til. Likevel finnes det gode håndregler som gjør at en kan gjøre tilstrekkelig nøyaktige høydemålinger på bakgrunn av trykkmålinger.

En regner normalt at trykket faller med 1 millibar pr. 8 meter, eller ca 12.5 millibar pr. 100 meter. Dette stemmer ikke så verst for de første 2000 meter, deretter minker trykket mindre for hver 1000 meter.

Normalt refereres alle trykkmålinger til havnivået. En meteorologisk stasjon som oppgir barometerstand ved stasjonen, har vanligvis regnet om verdiene til havnivået (dette variere noe fra stasjon til stasjon).

Tabellen under viser typiske verdier for sammenhengen mellom trykk, lufttetthet, temperatur og høyde over havet.

HoH	**Temperatur**	**Lufttrykk**	**Tetthet**	
(m)	*(C)*	*(hPa)*	*(kg/m^3)*	
0000	15.0	1013	1.2	
1000	8.5	900	1.1	
2000	2.0	800	1.0	(Galdhøpiggen)
3000	-4.5	700	0.91	
4000	-11.0	620	0.82	
5000	-17.5	540	0.74	
6000	-24.0	470	0.66	
7000	-30.5	410	0.59	
8000	-37.0	360	0.53	
9000	-43.5	310	0.47	(Mount Everest)
10000	-50.0	260	0.41	(Marsjhøyde rutefly)
11000	-56.5	230	0.36	
12000	-56.5	190	0.31	
13000	-56.5	170	0.27	
14000	-56.5	140	0.23	
15000	-56.5	120	0.19	
16000	-56.5	100	0.17	
17000	-56.5	90	0.14	
18000	-56.5	75	0.12	
19000	-56.5	65	0.10	
20000	-56.5	55	0.088	
21000	-55.5	47	0.075	
22000	-54.5	40	0.064	
23000	-53.5	34	0.054	
24000	-52.5	29	0.046	
25000	-51.5	25	0.039	
26000	-50.5	22	0.034	

27000	-49.5	18	0.029
28000	-48.5	16	0.025
29000	-47.5	14	0.021
30000	-46.5	12	0.018
31000	-45.5	10	0.015
32000	-44.5	8.7	0.013
33000	-41.7	7.5	0.011
34000	-38.9	6.5	0.0096
35000	-36.1	5.6	0.0082

Omregningen fra trykk til høyde må også ta hensyn til temperaturen. Temperaturen vil dessuten forandre seg med høyden.

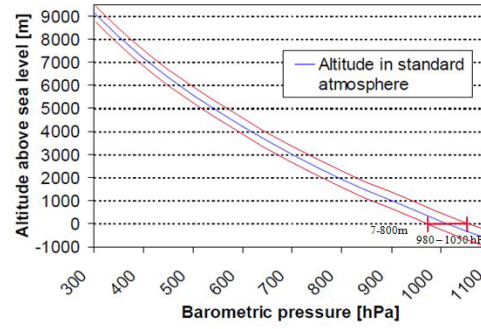
Det er normalt lettere å forholde seg til en omregningsformel enn en tabell. Ulempen med en formel er at de mange parametrene kan gi stor usikkerhet i beregningen. I *The CanSat book* er følgende sammenheng referert:

$$h = \frac{T_1}{a}\left(\left(\frac{p}{p_1}\right)^{-\frac{aR}{g_0}} - 1\right) + h_1 \tag{5.13}$$

Hvor:

h	Beregnet høyde i meter
h_1	Starthøyde i meter
T	Temperatur i Kelvin
T_1	Starttemperatur i starthøyden h_1
a	Temperaturgradient, foreslått verdi -0,0065 K/m
p	Trykk i Pa
p_1	Trykk i Pa ved starthøyden
g_0	Tyngdeakselerasjonen 9,81 m/s^2
R	Den spesifikke gasskonstant 287,06 J/kg K

Denne formelen kan enten legges inn i datainnsamlingsenheten i CanSat, men bedre i programvaren som behandler data nede på bakken. Har man rådataene fra sonden, har en større mulighet til etterbehandling enn om man bare har de omregnede dataene.



Diagrammet til høyre viser sammenhengen mellom trykk og høyde med økende høyde over havnivået. Normale variasjoner i lufttrykket ved bakken kan være fra 980 hPa til 1050 hPa (millibar). Denne naturlige variasjonen kan derfor gi en absolutt endring i høydeberegningen på 7 – 800 meter dersom man ikke kalibrerer på bakkenivå.

Kalibrering

Dersom man ønsker å bruke trykkmåleren for å måle absolutt høyde over havet så er det særdeles viktig at man kalibrerer sensoren. Dette gjør man enklest ved å finne ut hva starthøyden over havet er og så anta at lufttrykket er relativt stabilt mens målingene skjer.

5.3.7 Biblioteket for GY91

Biblioteket som følger med GY91 definerer noen overordnede kommandoer som gjør det lettere å kommunisere med GY91. NAROM har tilpasset biblioteket til vår applikasjon.

Initialisering

```
#include <gy91.h>
```

... gjør biblioteket til GY91 tilgjengelig for programmet. Denne kommandoen skal alltid stå nær starten av programmet dersom man bruker funksjoner fra biblioteket.

```
GY91 gy91;
```

... definerer objektet *gy91* som er av typen GY91. Navnet *gy91* kan i prinsippet velges fritt, men det kan være praktisk å bruke et navn som gir mening og som gjør at vi vet hva vi snakker om. Dersom man har flere kretser av typen GY91 kan man f.eks. kalle dem med forskjellig navn. Alle funksjoner som angår denne kretsen vil derfor i vårt tilfelle begynne med gy91. ...

Deklarasjonen av objektet gy91 settes sammen med deklarasjon av de andre globale variablene og gjerne nær starten av programmet.

```
gy91.init();
```

... initialiserer objektet *gy91* og sjekker at kretsen er klar til å levere data. Denne funksjonen har ikke noe argument, men leverer en 1'er om alt er i orden og den er klar til å levere data eller 0 og kan ev. skrive ut en feilmelding. Som i testprogrammet vårt:

```
if (!gy91.init())
{
Serial.println("Could not initiate");
while(1);
}
```

Feilmeldingen "Could not initiate" skrives ut dersom gy91.init returnerer 0 (dvs. !gy91.ini gir 1).

Akselerometeret

```
gy91.read_acc();
```

... henter inn data fra akselerometeret og legger resultatet i:

```
gy91.ax
gy91.ay
gy91.az
```

Disse kan så legges over i tre variable for x-, y- og z-retningen som f.eks. her:

```
ax = gy91.ax;
ay = gy91.ay;
az = gy91.az;
```

Gyrometeret

```
gy91.read_gyro();
```

... henter inn data fra gyrometeret og legger resultatet i:

```
gy91.gx
gy91.gy
gy91.gz
```

Disse kan så legges over i tre variable for x-, y- og z-retningen som f.eks. her:

```
gx = gy91.gx;
gy = gy91.gy;
gz = gy91.gz;
```

Magnetometeret

```
gy91.read_mag();
```

... henter inn data fra magnetometeret og legger resultatet i:

```
gy91.mx
gy91.my
gy91.mz
```

Disse kan så legges over i tre variable for x-, y- og z-retningen som f.eks. her:

```
mx = gy91.mx;
my = gy91.my;
gz = gy91.mz;
```

Lufttrykk

```
gy91.readPressure()
```

... henter inn data fra trykkmåleren BMP 280 som f.eks. kan legges over i variabelen:

```
pressure = gy91.readPressure();
```

5.4 Tranceiver (sender/mottaker) - RFM96[16]

5.4.1 Kretsbeskrivelse[17]

RFM96

RFM96 er en liten radio for bruk på ISM-båndet 433 – 434 MHz. Radioen er utviklet av Semtech og bruker sitt eget format for overføring av data (protokoll) kalt LoRa (Long range). Den inneholder både en mottaker og en sender slik at den både kan sende og motta data, den kalles derfor en Tranceiver (Trans(mitter/Re)ceiver). Den kan kun operere i såkalt "simplex", dvs. at den må vekselvis sende og motta, aldri begge deler samtidig, som for de gamle Walkie Talkiene. Senderen har en maksimal utgangseffekt på 100 mW (20 dBm). Mottakeren har en følsomhet på -148 dBm hvilket er meget bra.

Radioen har tidligere vært brukt ved NAROM for overføring av data fra værballonger. Med lav datarate og fri sikt har rekkevidden vært opp i 60 km. Dataraten kan i teorien være så høy 300kbit/sek, imidlertid vil økt datarate redusere rekkevidde og øke feilraten i tillegg til at kravet til båndbredde øker. Det er derfor sterkt å anbefale lavere datarater som f.eks. 9600 bit/sek.

Som det framgår av tabellen under finnes det flere varianter av radioen med litt ulike spesifikasjoner. Vi legger også merke til at den praktiske dataraten er avhengig av valgt båndbredde på kanalen og kan i beste fall komme opp i 37,5 kbps (kilobit pr. sek.) med en kanalbåndbredde på 500kHz. Med en kanalavstand på 100 kHz vil båndbredde måtte være noe lavere som resulterer i lavere datarate.

Part Number	Frequency Range	Spreading Factor	Bandwidth	Effective Bitrate	Est. Sensitivity
RFM95W	868/915 MHz	6 - 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm
RFM97W	868/915 MHz	6 - 9	7.8 - 500 kHz	0.11 - 37.5 kbps	-111 to -139 dBm
RFM96W/RFM98W	433/470MHz	6- 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm

Radioen kan kommunisere med ulike modulasjonsformer – OOK (On Off Keying), FSK (Frequency Shift Keying), GFSK (Gaussian FSK), MSK (Minimum Shift Keying) og GMSK (Gaussian MSK). I tillegg kan man benytte en spredekode som smører effekten utover et større bånd for så samle effekten ved mottak, dette bedrer signal-støyforholdet.

Radioen kommuniserer med Teensy 3.5 via et SPI-seriegrensesnitt som er meget raskt (klokkes med 8 MHz). Datahastigheten mellom sender og mottaker er likevel noe begrenset, så det er viktig å begrense datamengden. Overføringen skjer ved at dataene pakkes og overføres i sekvenser av data (en datapakke). Senderen varsler når en datapakke er sendt og den er klar for å sende en ny. Lar man programmet følge med på når datapakker er sendt kan man utnytte kapasiteten optimalt.

16. Datblad finnes her: http://www.hoperf.com/upload/rf/RFM95_96_97_98W.pdf
17. Stoffet til dette avsnittet er dels hentet fra NAROM hjemmeside og dels fra databladet for radioen

Fordeling av frekvenser

Når flere CanSat-lag skal overføre data samtidig er det nødvendig at man blir enige om en fordeling av frekvenser. Dersom to lag velger å bruke samme eller en nærliggende frekvens vil de kunne forstyrre hverandre. Vi anbefaler derfor at hvert lag velger frekvenser med en avstand på minst 100 kHz.

En annen fordel med denne radioen er at frekvensen kan settes fra programkoden f.eks. i setup()-rutinen. I tillegg til at man setter opp frekvensen blir det automatisk satt opp et unikt synkroniseringsord som gjør at data som tilhører en annen bruker blir ekskludert slik at man unngår å blande sammen data.

NAROM har utviklet et eget bibliotek (CanSat_RFM96) for bruk av radioen slik at den skal være enkelt å bruke. Ved at biblioteket inkluderes i starten av programkoden, blir en rekke funksjoner for håndtering av sending og mottaking tilgjengelig ved programmering. Før biblioteket kan brukes må det installeres i programeditoren for Arduino (IDE) (avsnitt 4.2 på side 38).

Biblioteket CanSat_RFM96 inneholder også funksjoner for skriving og lesing fra SD-kort.

5.4.2 Etablering av telemetri kanal fra CanSat til jordstasjonen (PC)

Vi har nå bygget opp to CanSat-kort CCB (CanSat Control Board – CCB) og bakkekontrollkortet, GCB (Ground Control Board) som er litt forskjellige. I CanSat'en finner vi et komplett kort (CCB) som inneholder alt inkludert sensorkortet GY-91 og NTC temperaturmåleren. Dette kortet vil normalt operere som sender og overføre data til bakkestasjonen samtidig som den skriver dataene til SD-kortet ombord.

Det andre kortet, bakkekontrollkortet, GCB, kobles til PC'en som står på bakken og som primært skal brukes som mottaker. På mottakerkortet har vi utelatt sensorkortet (GY91) og NTC temperaturmåleren i tillegg til noen motstander. Dette kortet trenger heller ikke batteritilkobling eller spenningsregulator siden det får spenning fra USB-kontakten.

Siden begge kortene har SD-kort holdere, så er PC'en strengt tatt ikke nødvendig, dataene kan lagres direkte på et kort på GCB-kortet. Imidlertid kan det være greit å kunne observere at data kommer inn. Dessuten leverer PC'en spenning til kortet.

Det må dermed lages to programmer ett for mottakeren og ett for senderen som er litt forskjellige. Samtidig som den tar imot og skriver data til SD-kortet kan den også skrive til Arduono monitoren i IDE'en. Dermed trenger vi ikke prinsippet ikke å bruke terminalprogrammer.

Testprogrammer for sender og mottaker

Listing av testprogrammet for senderen finnes i vedlegg B.2 på side 130 og for mottakeren i vedlegg B.3 på side 133.

5.4.3 Test og bruk av radioen

Vi skal nå koble opp de to kortene å se at vi får overført data fra CanSat-kortet (CCB) til bakkekontrollkortet (GCB).

Det er utviklet to programmer for testing av radiokommunikasjonen mellom de to kortene som nevnt i forrige avsnitt:

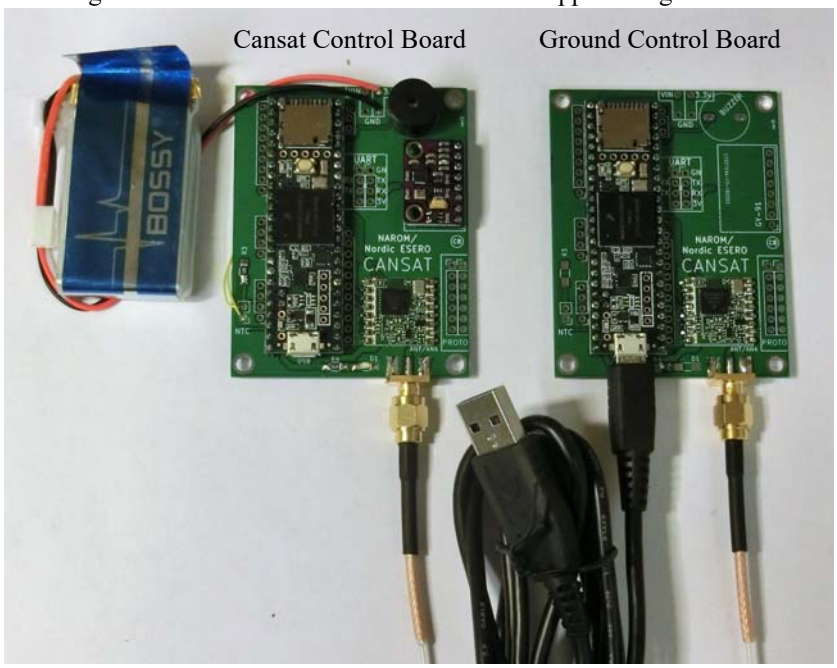
- *Cansat_RadioSendTest* – som skal installeres i CCB og som sender ut data til GCB
- *Cansat_RadioRecvTest* – som skal installeres i BCB som mottar data fra CCB

Dersom de nevnte filene og andre testfiler legges i katalogen *Dokumenter/Arduino* så vil de dukke opp i Arduino programeditoren under menyfanen: F*ile/skissebok* og kan hentes opp i editoren.

Gjør følgende:

- Starte Arduino programeditoren (IDE)
- Koble CCB til USB porten på PC'en
- Hent *Cansat_RadioSendTest* programmet i IDE'en og last det over til Teensy 3.5 (CCB)
- Siden kortet står i USB-porten skal du se de sendte dataene dersom du åpner monitoren (IDE)
- Koble fra CCB og koble det til LiPo-batteriet (pass på at LiPo-batteriet er fulladet.
- Koble GCB til USB porten på PC'en
- Hent *Cansat_RadioRecvTest* programmet i IDE'en og last det over til Teensy 3.5 (GCB)
- Åpne monitoren å sjekk at det overføres data fra CanSat-kortet (CCD)

Dersom dette fungerer er forbindelsen mellom de to kortene opprettet og kortene er klare til bruk.



5.4.4 Testprogrammene for RFM96 og noen sentrale kommandoer

De kommandoene som nevnes i dette avsnittet er ikke alle som er inkludert i biblioteket. I tillegg finnes det kommandoer hvor man kan endre båndbredde, datarate og spredningsfaktor, i tillegg til kommandoer for feilfinning.

NAROM har utviklet et bibliotek for radioen RFM96 som må inkluderes i toppen av programmet som skal bruke radioen:

```
#include <Cansat_RFM96.h>
```

Grunnleggende kommandoer

Her er noen flere viktige kommandoer.

Definer radioen som et objekt og initier den:

```
Cansat_RFM96 rfm96(433500);
```

Denne kommandoen definerer et objekt som vi kaller ***rfm96*** samtidig vi setter frekvensen knyttet til dette objektet til 433500 kHz). Her er det viktig å bruke den frekvensen det enkelte CanSat laget får oppgitt som sin frekvens. Kommandoen vil dessuten automatisk medføre at dataene leses ned på SD-kortet. Dersom man ikke ønsker at data skal lagres på SD-kort, kan man istedet bruke kommandoen:

```
Cansat_RFM96 rfm96(433500, False);
```

Denne hindrer kommandoer i programmet som forsøker å skrive til SD-kortet.

```
Cansat_RFM96 init();
```

Denne kommandoen initialiserer objektet `rfm96` (radioen). Den har ingen argumenter, men vil returnere en melding om at radioen er klar for å sende/motta data. Kommandoen returnerer "1" dersom radioen er klar og "0" om den ikke er klar. Det kan derfor være lurt å bruke denne før sending:

```
if (!rfm96.init()) {
    Serial.println("Init of radio failed, stopping");
    while(1);
```

Vi legger merke til utropstegnet foran `rfm96.init()` som gjør at if() funksjonen iverksetter en feilmelding dersom funksjonen returnerer en "0".

Skriv til radio og SD-kort:

Når man ønsker å sende data over radioen, skrive til SD-kortet eller begge deler, bruker man følgende kommandoer:

```
rfm96.printToBuffer(1001);
rfm96.printToBuffer(this_is_a_variable_containg_numbers);
rfm96.printToBuffer("This is some text");
```

Tilsvarende med påfølgende linjeskift:

```
rfm96.printlnToBuffer(1001);
```

```
rfm96.printlnToBuffer(this_is_a_variable_containg_numbers);
rfm96.printlnToBuffer("This is some text");
rfm96.printlnToBuffer();
```

Før dataene kan sendes må de skrives inn i et mellomlager (buffer). Dette er kommandoer som er likt med det vi er vant til fra Arduino, bare at her bruker vi `rfm96.printlnToBuffer` og `rfm96.printToBuffer` for å skrive med og uten linjeskift i stedet for `Serial.println();` og `Serial.print();` Alle disse kommandoene konverterer data til ASCII tegn som kan leses som tall og bokstaver når fila åpnes av en teksteditor eller f.eks. Excel.

Dersom man vil unngå konvertering til ASCII-kode og ønsker å få lagret bitene "rått" kan man benytte:

```
rfm96.writeToBuffer(101);
```

Dette er en kommando man sjelden benytter, men som noen ganger kan være nyttige.

Send data til bakkestasjon og skriv til file:

Når så alle data er overført til bufferet utføres sendkommandoen og dataene sendes i pakker ned til bakkestasjonen:

```
rfm96.send();
```

Dersom radioen ikke er klar for sending, vil hele prosessen være blokkert til senderen gir klar-signal. Send-kommandoen tømmer også bufrene slik at de sendte dataene blir utilgjengelige etter at kommandoen er utført.

Alternativt kan man skrive dataene til SD-kortet med denne kommandoen:

```
rfm96.writeToFile();
```

Eller vi kan gjøre begge operasjonene med en kommando:

```
rfm96.sendAndWriteToFile();
```

Denne kommandoen skriver alt som er i bufrene til SD-kortet, samtidig sjekkes om radioen er klar til å sende data. Om den er klar sendes dataene, om den ikke er klar vil sendingen avbrytes og programmet går videre. Med andre ord, så lenge dataene er lagret på SD-kortet kan den "leve med" at radioen ikke får sendt noe. Uansett vil den slette bufrene.

Kommandoene under viser hele sekvensen ved sending med kommentarer:

```
uint8_t Cansat_RFM96::sendAndWriteToFile() {
  uint8_t tmp_length;
  tmp_length = writeToFile(); // This is always ready,
                              // and will also flush every time
  if (isTxReady())
    tmp_length = send();      // IF we send, then this is the length we return

  clear();                    // We empty the buffer and are ready to fill it up
                                 again

  return(tmp_length);
```

```
}
```

Tøm bufrene:

Følgende kommando vil tømme alle bufrene slik at de er klare til å lagre nye data:

```
rfm96.clear()
```

Denne må brukes dersom man kun skriver til file. Dersom man kun sender data til bakkestasjonen så vil send-kommandoen selv inkludere tømming av bufrene, noe som kommandoen "skriv til file" **`rfm96.writeToFile();)`** ikke gjør.

Les av signalstyrken

Radioen er utstyrt med en signalstyrkemåler (RSSI – Received Signal Strength Indicator) når den brukes som mottaker. Denne måleverdien kan leses av med følgende kommando. Verdien blir i dette tilfellet liggende i variabelen: *last_RSSI_value_in _dBm*:

```
int last_RSSI_value_in_dBm = rfm96.last_RSSI();
```

Dette er et godt hjelpemiddel for å få en indikasjon på hvor nær man er fra å miste signalet. CanSat'en vil normalt begynne å miste kontakten med en signalstyrke ved -80dBm.

Sendeeffekt

Ved hjelp av denne kommandoen kan man sette sendeeffekten fra +5 dBm til +20 dBm (100mW. Verdien settes i dBm. dBm er en måleenhet som angir effekten i forhold til 1 mW. For hver dobling av effekten i mW så økes dB-nivået med 3dB slik at:

0 dBm = 1 mW
3 dBm = 2 mW
6 dBm = 4 mW
9 dBm = 8 mW
12 dBm = 16 mW
15 dBm = 32 mW
18 dBm = 64 mW
20 dBm = 100 mW

Kommandoen for å sette sendeeffekten er:

```
rfm96.setTxPower(20);
```

I eksempelet over settes effekten til +20 dBm eller 100 mW.

5.5 Bruk av GPS - NEO-6M

GPS eller *Global Positioning System* består av 24 satellitter som kretser omkring jorda med en omløpstid på 11 t 58 min. i en høyde av ca. 20200 km over jordoverflata. Normalt vil dette antallet være tilstrekkelig for, til enhver tid, å ha fri sikt til 8–10 satellitter i åpent terreng. Hver satellitt sender ut et kodet tidssignal som mottas av mottakerne. I tillegg til å inneholde informasjon om nøyaktig tid, inneholder signalet tidspunkt for utsendelse og en lang kode som mottakerne bruker

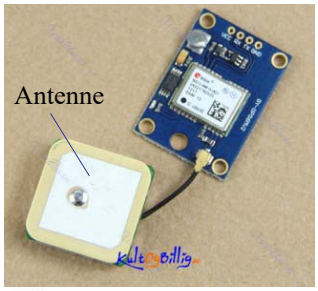
for nøyaktig å bestemme tidspunktet for mottatt signal. På denne måten kan mottakersystemet bestemme tiden hvert av signalene bruker fra hver av satellittene til mottakeren. De målte tidsforsinkelsene brukes så til å beregne posisjonen til mottakeren [1].

En ***GPS-modul*** er et sett av GPS mottakere som gjør det mulig å følge flere GPS-satellitter samtidig. Seks er ikke uvanlig. 4 er et minimum for å kunne beregne koordinater pluss høyde.

5.5.1 GY-NEO-6MV2[18] Flight (GPS-modul)

Teknisk - spesifikasjon

Dette er en annen vanlig brukt GPS-modul. Den kan bl.a. kjøpes fra www.kultogbillig.no for kr. 198,- inkl. MVA, eller fra AliExpress for ca. kr. 35,00 (U$ 4,0)[19]. Den består av to separate enheter forbundet med en koaksialkabel. Det ene er antennemodulen, den andre analyse- og datamodulen.

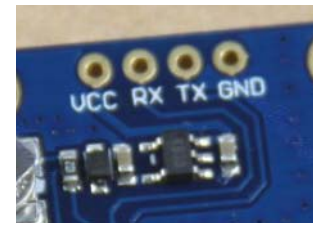


- Enheten er bygget opp rundt chip-settet: ARM
- Mottakerfrekvens: 1575,42 MHz
- Antall kanaler: 50 kanaler
- Følsomhet: -161dBm ("Tracking & Navigation)
- Nøyaktighet: - Horisontal: 2,5 m
  - Hastighet: 0,1 m/s
  - Kurs retning: 0,5°
  - Tid: 30 ns (tidspulssignalets nøyaktighet)
- Dato: Leverer data og tid ned til sekunder
- Låsetid: Varm (Hot) start: 1 sek. i gjennomsnitt
  Kald start: 27 sek. i gjennomsnitt
  Dette er viktig å merke seg siden den aktive tiden til CanSat er så kort
- Maks verdier: Maks. målehøyde: 50 000 meter
  Maks. hastighet 500 m/s
  Maks. akselerasjon 4 g
  Maks. oppdatering 5 Hz
- Forsyning: Spenning: 2,7 – 3,6 V DC
  Strømforbruk: Max. 47 mA

18. Datablad: https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/GY-NEO6MV2-GPS-Module-Datasheet.pdf
19. https://www.aliexpress.com/

- Dig. utgang: Spenningsnivå: TTL nivå (0 – 2,85 V)
  UART, USB (12 Mbit/sek), SPI (100 kbit/sek)
  Baud rate: 9 600 baud (symboler/sek.)
  Format: NMEA GSV, RMC, GSA, GGA, GLL, VTG, TXT
- Størrelse: 25 x 25 x 8 mm (antennemodul) +
  25 x 35 x 5 mm (kontrollmodul)
- Temperaturområde: –40°C til 85°C
- Backup batteri: For å holde data ved “Power down”

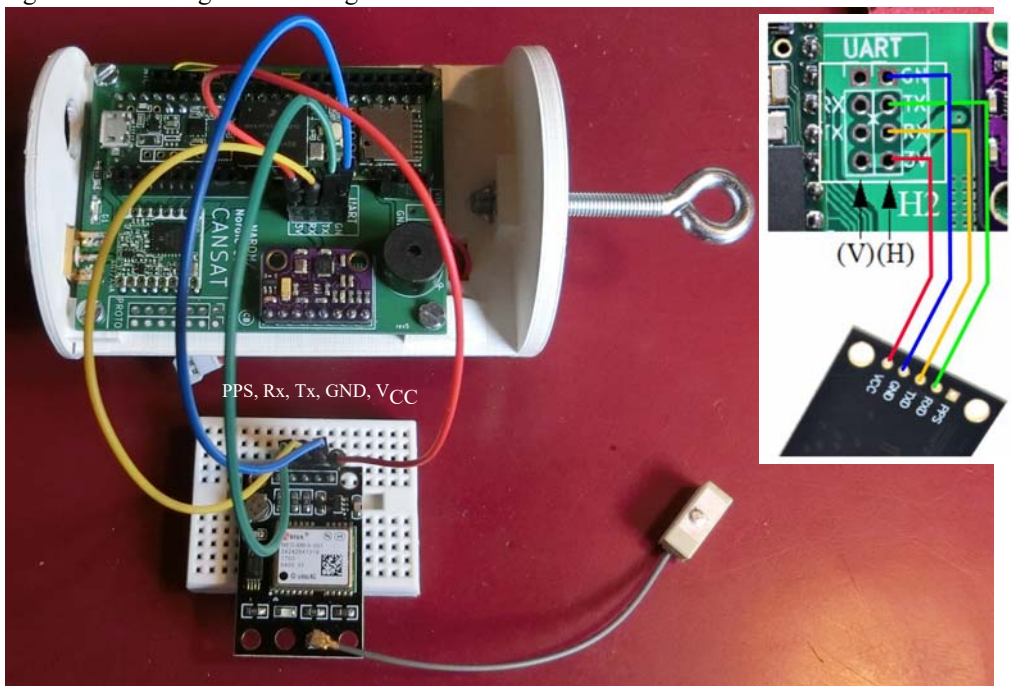
Oppkobling mot Teensy 3.5



Kontroll-modulen kobles til Teensy 3.5 via fire terminaler (på noen varianter finnes en femte terminal PPS). Mikal Hart [3] argumenterer for at den serielle UART knyttet til pinne I/O-port 0 () og 1 () kan brukes siden disse normalt ikke brukes for å kommunisere med PC’en slik f.eks. Arduino serien gjør. Teensy har en egen USB-port. UART’en er på CanSat-kortet tatt ut til egne terminaler hvor en kan montere hylsekontakter. Vi kan da gjøre oppkoblingen på følgende måte:

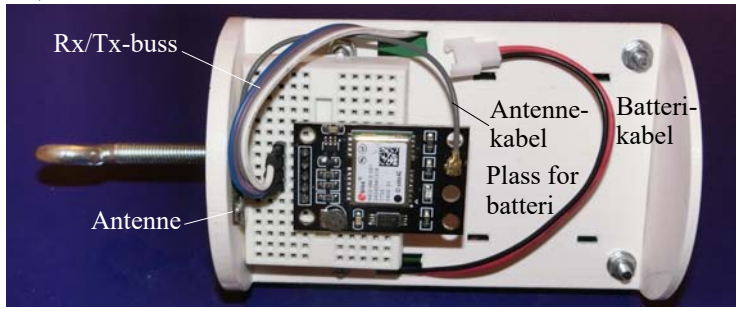
V_{CC}	3,3 V
Rx (GPS)	I/O-port 1 (Tx1 – Teensy 3.5)
Tx (GPS)	I/O-port 0 (Rx1 – Teensy 3.5)
GND	GND
PPS	Ikke tilkoblet

Ved å lodde en stiftlist til terminalene på GSM-modulen kan en lett koble opp modulen på et koblingsbrett for testing som vist i figuren under.



Figuren over til høyre viser i detalj hvordan oppkoblingen er utført.

Bildet under viser hvordan GPS-modulen er montert på baksiden av kortet med en flatkabel i mellom (Rx/Tx-buss).



5.5.2 Lagring av data fra GPS-modul NEO-6M

I dette avsnittet skal vi hente inn data fra GPS-enheten NEO-6M og lagre disse på SD-kortet som er på CanSat Teensy 3.5 kortet.

Installasjon av bibliotek

Biblioteket *TinyGPS* må installeres for å kunne hente ut data fra GPS-kontrollkortet.

Biblioteket kan hentes fra:

https://github.com/mikalhart/TinyGPS/releases/tag/v13

Hent ned zip-fila som heter: *Source code (zip)*. Ikke pakk ut fila, men lagre den på et sted du kan finne den igjen. Biblioteket installeres ved å åpne velge Skisse/Include Library/Add .zip library i Arduino editoren (IDE) og hent opp fila[20].

Enkleste kode

Fra avsnitt 5.5.3 på side 85 så ser vi at foretrukket format på koordinat- og høydedataene er:

lengdegrad,breddegrad,høyde

Alle verdier oppgitt som desimaltall og uten mellomrom etter komma.

Koden under viser den enkleste utgaven av programvaren for å hente inn koordinatdata og skrive til både monitoren i Arduino (IDE):

#include <TinyGPS.h>

```
TinyGPS gps;

/* Denne koden er bygget på arbeidet til Mikal Hart sitt arbeid
   Hentet fra https://www.pjrc.com/teensy/td_libs_TinyGPS.html
   Koden er bearbeidet og forenkelt for plotting i Google Earth
   av Nils Kr. Rossing 11.08.18*/

HardwareSerial Uart = HardwareSerial();

void setup()
{
  Serial.begin(115200);
  Uart.begin(9600);

  delay(1000);
  Serial.println("Bredde, lengdegrad og høyde ");
  Serial.println("ved Mikal Hart (Nils Kr. Rossing 11.08.18");
  Serial.println();
}

void gpsdump(TinyGPS &gps);
```

20.Ev. se https://github.com/mikalhart/TinyGPS/releases/tag/v13

```
void printFloat(double f, int digits = 2);

void loop()
{
  bool newdata = false;
  unsigned long start = millis();

  // Every 5 seconds we print an update
  while (millis() - start < 5000)
  {
    if (Uart.available())
    {
      char c = Uart.read();
      if (gps.encode(c))
      {
        newdata = true;
      }
    }
  }

  if (newdata)
  {
    gpsdump(gps);
  }
}

void gpsdump(TinyGPS &gps)
{
  float flat, flon;
  unsigned long age;

  gps.f_get_position(&flat, &flon, &age);
  printFloat(flat, 6);
  Serial.print(",");
  printFloat(flon, 6);
  Serial.print(",");
  printFloat(gps.f_altitude());
  Serial.println();
```

```
}

void printFloat(double number, int digits)
{
  // Håndtering av nagative tallverdier
  if (number < 0.0) {
     Serial.print('-');
     number = -number;
  }

  // Sørger for korrekt avrunding slik at print(1.999, 2) skrives som
"2.00"
  double rounding = 0.5;
  for (uint8_t i=0; i<digits; ++i)
    rounding /= 10.0;

  number += rounding;

  // Hent ut heltallsverdien av mottatt tall
  unsigned long int_part = (unsigned long)number;
  double remainder = number - (double)int_part;
  Serial.print(int_part);

  // Skriv ut desimalpunktet
  if (digits > 0)
    Serial.print(".");

  // Hent ut resten når heltallsveriden er trukket fra og skriv ut
  while (digits-- > 0) {
    remainder *= 10.0;
    int toPrint = int(remainder);
    Serial.print(toPrint);
    remainder -= toPrint;
  }
}
```

Etter innsamling av data kan disse lastes opp i f.eks. Notepad++ eller i Excel, se avsnitt 7.4 på side 100. Dersom man ønsker å visualisere traseen i Google Earth så klippes dataene inn i kml-koden vist i avsnitt 5.5.3 på side 85.

Kode med feilsjekk og flere data

Den enkle programkoden omtalt i forrige avsnitt inneholder ingen sjekk mht. om dataoverføringen mellom GPS-enheten og Arduino er korrekt. En mer omfattende kode med feilsjekking finnes i B.5 på side 135.

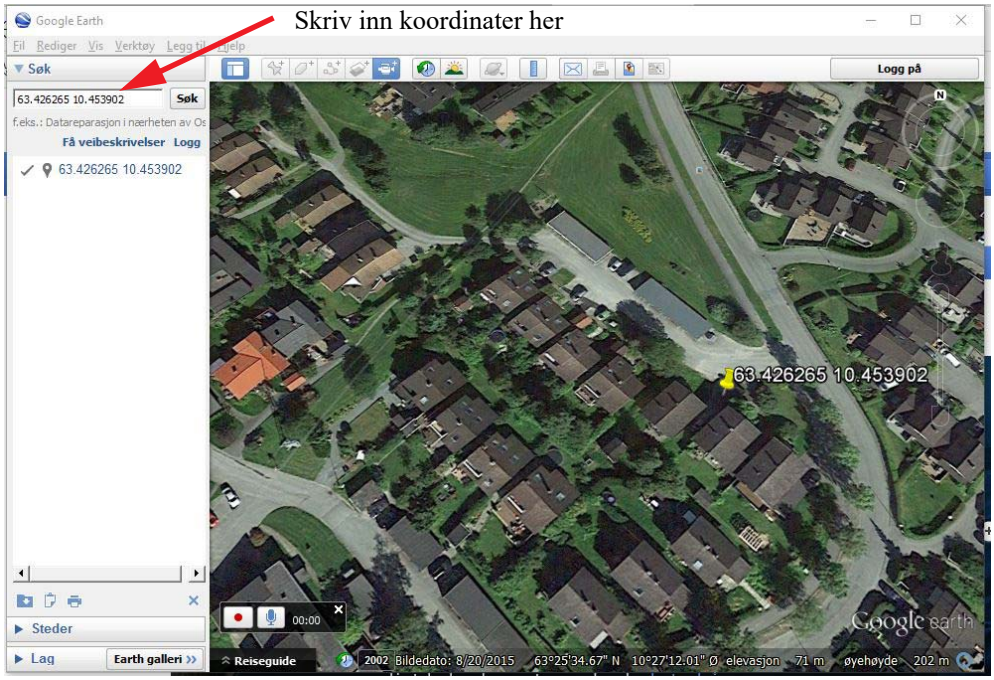
5.5.3 Visualisering av GPS-data i Google Earth

Data fra GPS-moduler krever gjerne en litt annen behandling enn ved hjelp av regneark. Slike data kan f.eks. plottes i Google Earth. I dette avsnittet skal vi se hvordan vi kan hente inn de lagrede dataene og plotte dem i Google Earth.

Installasjon av Google Earth

Google Earth kan lastes ned og installeres fra følgende adresse: https://www.google.com/earth/

Skjermbildet under viser brukergrensesnittet for Google Earth.



I øverste venstre hjørne finnes et søkefelt. Her kan man enten skrive en adresse, eller man kan skrive inn koordinater (med desimaler). I dette tilfellet har jeg skrevet inn:

Breddegrad: 63.426265
Lengdegrad: 10.453902

Da vil Google Earth zoome inn akkurat til disse koordinatene som i dette tilfellet var rett utenfor veggen der jeg sitter. Dvs. det kan se ut som den bommer med et par meter (ringen angir plasseringen av mottakeren).

Høydeangivelsen for det angitte stedet varier mellom 60 – 80 meter over havet (Google Earth angir en høyde som varierer mellom 65 – 75 meter).

Plotting av en trase i Googel Earth

Dersom man ønsker å plotte en trase i Google Earth må man benytte et programmeringsspråk kalt "Keyhole Markup Language" (KML). Dette er et "markup language" utviklet for visualisering av to- og tredimensjonale strukturer knyttet til kartdata. Språket ble utviklet i forbindelse med etableringen av Google Earth som ble lansert i 2004. I 2008 ble KML godkjent som en internasjonal standard for denne type geografisk visualisering.

Siden språket er temmelig omfattende og vi kun trenger å bruke en beskjeden del av det, så benytter vi en ferdige programkode og klipper inn våre data for lengde-, breddegrad og høyde. Disse legges inn som en liste med data i kml-koden (se under), før kml-fila lagres med et ønsket navn.

Koordinater og høyde data legges inn som vist under:

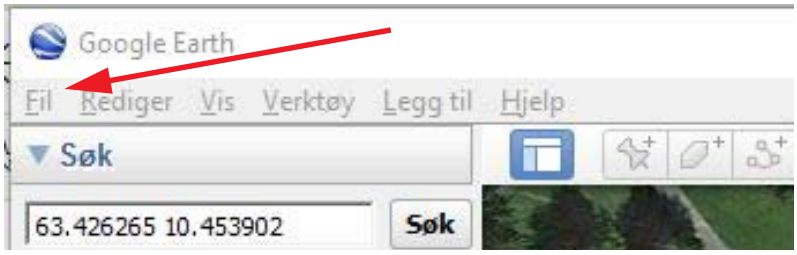
-112.2550785337791,36.07954952145647,2357

Lengdegrader [°], Breddegrader [°], Høyde [m]

Programkode skrevet i kml (legg merke til at et sett med eksempelkoordinater og høyde er klippet inn).

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
 <Document>
  <name>Paths</name>
  <description>Examples of paths. Note that the tessellate tag is by default
   set to 0. If you want to create tessellated lines, they must be authored
   (or edited) directly in KML.</description>
  <Style id="yellowLineGreenPoly">
   <LineStyle>
    <color>7f00ffff</color>
    <width>4</width>
   </LineStyle>
   <PolyStyle>
    <color>7f00ff00</color>
   </PolyStyle>
  </Style>
  <Placemark>
   <name>Absolute Extruded</name>
```

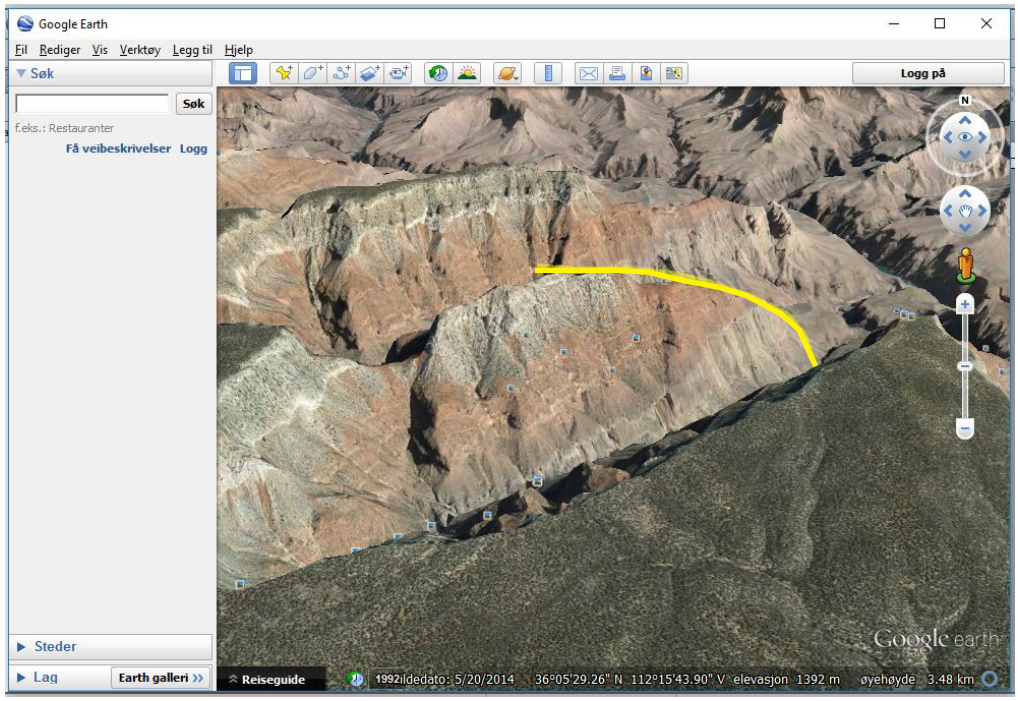
```
    <description>Transparent green wall with yellow outlines</description>
    <styleUrl>#yellowLineGreenPoly</styleUrl>
    <LineString>
      <extrude>0</extrude>
      <tessellate>0</tessellate>
      <altitudeMode>absolute</altitudeMode>
      <coordinates>
        -112.2550785337791,36.07954952145647,2357
        -112.2549277039738,36.08117083492122,2357
        -112.2552505069063,36.08260761307279,2357
        -112.2564540158376,36.08395660588506,2357
        -112.2580238976449,36.08511401044813,2357
        -112.2595218489022,36.08584355239394,2357
        -112.2608216347552,36.08612634548589,2357
        -112.262073428656,36.08626019085147,2357
        -112.2633204928495,36.08621519860091,2357
        -112.2644963846444,36.08627897945274,2357
        -112.2656969554589,36.08649599090644,2357
      </coordinates>
    </LineString>
  </Placemark>
 </Document>
</kml>
```



De koordinatene som pr. i dag ligger i eksempelet angir en helikopterflyvning over Grand Canyon i Colorado, USA. Eksempeldataene byttes ut med de aktuelle dataene og kodefila lagres under et ønsket filnavn som ender med .kml.

Filen hentes opp i Google Earth ved å velge *Fil* og *Åpne* for å laste opp den filen hvor koden og dataene ligger.

En vil da få tegnet inn traseen som angitt av lista med koordinater og vist på riktig sted. Eksempelet under viser traseen til helikopteret over Grand Canyon (gul linje).



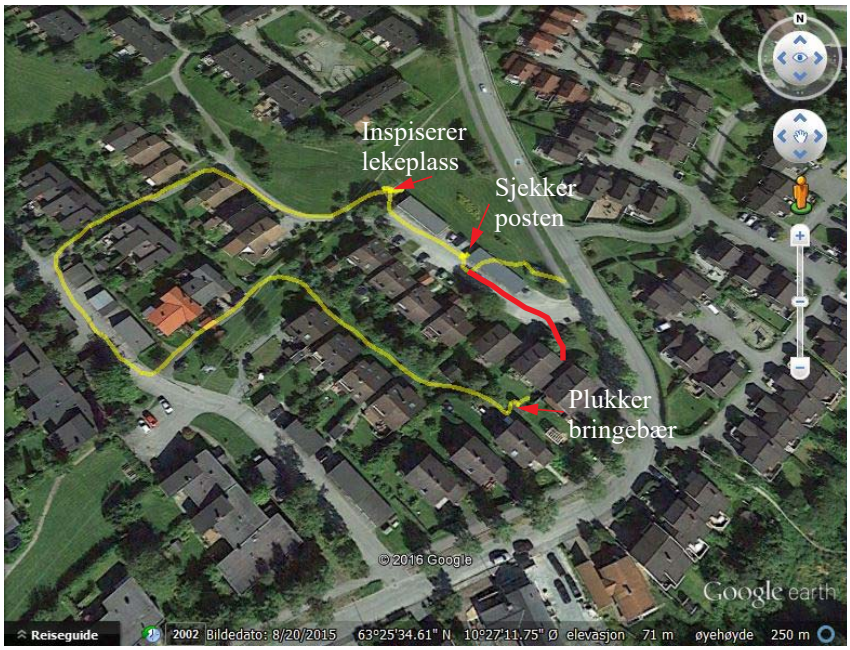
Dersom man ønsker å vise hvor man har godt en tur så kan det være upraktisk å måtte vise absolutt høyde. Høydemålinger kan ha store avvik slik at en kan oppleve at traseen går mange meter over bakken eller forsvinner under bakken til tross for at man hadde begge beina på jorda. I slike situasjoner kan det være greit å bytte ut kommandoen:

<altitudeMode>absolute</altitudeMode>

med kommandoen:

<altitudeMode>clampToGround</altitudeMode>

Dermed vil kurven følge bakken som vist på traseen på bildet under.



Vi legger imidlertid merke til at mottakeren har problemer i starten. I dette området er avviket stort før den tar seg inn. Den røde kurven angir den riktige ruta. Deretter er den svært nøyaktig. Posisjonsdata samples hvert 3. sekund.

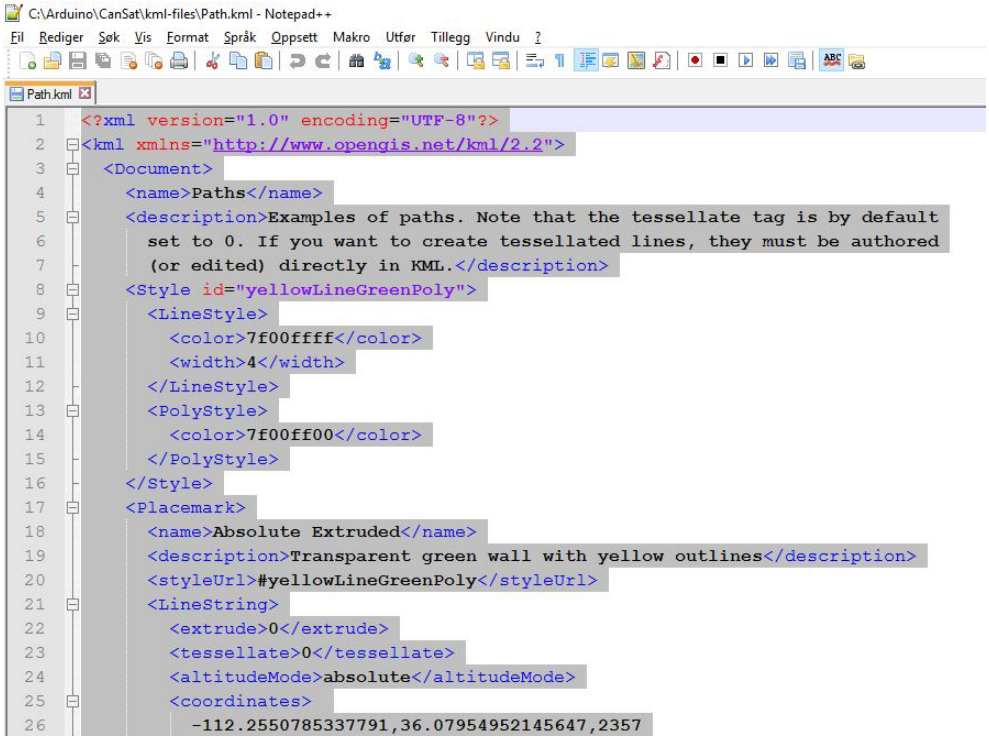
Ved bruk av GPS-data ved oppsending av CanSat kan man selvfølgelig ikke neglisjere høydeinformasjonen. Man bør imidlertid være klar over risikoen for relativt store avvik. En bør vurdere å legge inn høydedata fra høydemåleren framfor GPS-data.

Editering av kml-fila

Hvilket program skal man så bruke for å legge inn koordinater og høyde. Et nyttig editeringsverktøy til dette formålet er Notepad++. Dette programmet er en litt avansert teksteditor som ikke gjør noen endringer med filformatet så fremt man ikke ønsker det. Notepad++ kan lastes ned fra:

https://notepad-plus-plus.org/download/v6.9.2.html

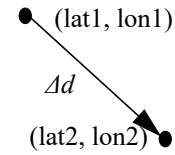
Figuren under viser et utsnitt av brukergrensesnittet til Notepad++.



For å forenkle prosessen med å klippe inn data i kml-koden, er det praktisk å skrive koordinat- og høydedata i det formatet som programmet ønsker: Lengdegrad, breddegrad, høyde. Husk å bruke punktum som desimalpunkt og komma mellom hver verdi. ***NB! Det skal ikke være mellomrom etter kommaene.*** For å se hvordan dette kan gjøres se avsnitt 5.5.2 på side 81.

5.5.4 Kode for beregning av akkumulert distanse

Det finnes i dag en rekke App'er for pader og smarttelefoner som beregner gangavstand etter som man beveger seg i gater eller i terrenget, disse er basert på smarttelefonens eller padens innebygde GPS mottaker. Et eksempel på en slik er Runkeeper[21].



Har vi først koblet en GPS-mottaker til vår Arduino, så burde det også være mulig å beregne tilbakelagt distanse. Siden høydemålingene er relativt usikre så nøyer vi oss med å bruke lengde- og breddegradene.

Som vist på figuren over så har vi to sett med koordinater (lat_1, lon_1 og lat_2, lon_2) og ønsker å beregne avstanden, Δd , mellom disse to settene av koordinater. Siden jorda er krummet som en kule så er ikke dette en triviell beregning. Under er gjengitt formelapparatet for beregningen[22]:

21.https://runkeeper.com/

$$\varphi_1 = \frac{\pi \cdot lat_1}{180} \tag{5.14}$$

$$\varphi_2 = \frac{\pi \cdot lat_2}{180} \tag{5.15}$$

$$\Delta\varphi = \varphi_2 - \varphi_1 \tag{5.16}$$

$$\Delta\lambda = \pi \cdot \frac{(lon_2 - lon_1)}{180} \tag{5.17}$$

$$a = \sin\left(\frac{\Delta\varphi}{2}\right)^2 + \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin\left(\frac{\Delta\lambda}{2}\right)^2 \tag{5.18}$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{(1-a)}) \tag{5.19}$$

$$\Delta d = R \cdot c \tag{5.20}$$

Hvor:

φ_1	*= Breddegrad posisjon 1 (lat_1) målt i radianer*
φ_2	*= Breddegrad posisjon 2 (lat_2) målt i radianer*
atan2	*= Arkus tangens med to argumenter for å beholde informasjon om kvadrant*
Δd	*= Distansen i meter*
R	*= Jordradien i m ($6{,}371 \cdot 10^6$ m)*

Dette er en relativt nøyaktig beregning, men kan være litt plundrete da standardbiblioteket til Arduino f.eks. ikke tilbyr atan2(). Imidlertid finnes det en forenklet versjon som kan være nyttig når man ikke trenger stor nøyaktighet eller når avstandene mellom målepunktene er liten.

Ekvirektangulær tilnærming

I vårt tilfelle er avstandene noen meter slik at det skulle være helt uproblematisk å bruke den forenklede beregningsmetoden:

$$\varphi_1 = \frac{\pi \cdot lat_1}{180} \tag{5.21}$$

$$\varphi_2 = \frac{\pi \cdot lat_2}{180} \tag{5.22}$$

$$\Delta\lambda = \pi \cdot \frac{(lon_2 - lon_1)}{180} \tag{5.23}$$

22.http://www.movable-type.co.uk/scripts/latlong.html

$$x = \Delta\lambda \cdot \cos\left(\frac{(\varphi_1 + \varphi_2)}{2}\right) \tag{5.24}$$

$$y = \varphi_2 - \varphi_1 \tag{5.25}$$

$$\Delta d = R \cdot \sqrt{x^2 + y^2} \tag{5.26}$$

Hvor:

φ_1	*= Breddegrad posisjon 1 (lat_1) målt i radianer*
φ_2	*= Breddegrad posisjon 2 (lat_2) målt i radianer*
Δd	*= Distansen mellom punktene i meter*
R	*= Jordradien i m ($6{,}371 \cdot 10^6$ m)*

Akkumulert distanse

Den akkumulerte distansen finner vi da enkelt ved å summer opp alle de små distansene mellom hvert målepunkt. En må passe på at målepunktene ikke blir for sjeldene slik at en mister de fine detaljene i registreringen.

$$d = \sum_{1}^{n} \Delta d \tag{5.27}$$

Hvor

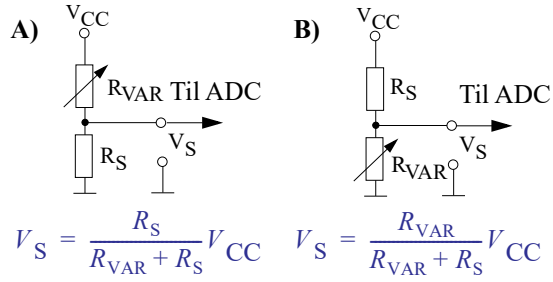
d	*= Lengden av traseen, dvs. summen av de mange små avstandene*
n	*= Antallet målepunkter*

6 Andre sentrale komponenter

I dette avsnittet vil vi omtale noen andre sentral komponenter. Innledningsvis kan dette kapittelet framstå noe knapt, men det kan bygges ut etter behov.

6.1 Spenningsdeleren

Mange sensorer er rent resistive. Dvs. at det aktuelle fenomenet (temperatur, fuktighet, trykk osv.) endrer sensorens indre resistans eller ledningsevne. Noen sensorer har inkludert elektronikk som konverterer endring i ledningsevne til en endring i spenning. Enklere sensorer krever ofte bruk av en spenningsdeler til å utføre konverteringen fra endring i resistivitet til endring i spenning. En slik spenningsdeler er vist på figuren til høyre. R_{VAR} er motstanden som avhenger av den aktuelle parameteren, mens R_S er en valgt fast seriemotstand. V_{CC} er batterispenningen. Ved hjelp av ligningene kan en beregne spenningen ut av spenningsdeleren når motstandsverdiene er kjent.



$$V_S = \frac{R_S}{R_{VAR} + R_S} V_{CC} \qquad V_S = \frac{R_{VAR}}{R_{VAR} + R_S} V_{CC}$$

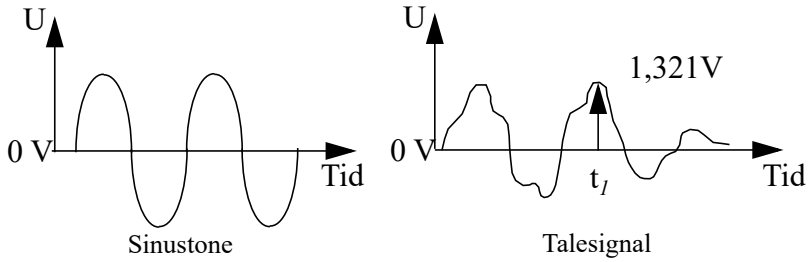
6.2 Analog til digital konverter (ADC)

Analog til digitalkonvertere (AD-konvertere) er viktige komponenter for å omdanne analoge signaler til digitale tallverdier. La oss først se litt på *sampling* og hva *digital representasjon* av et signal er.

6.2.1 Sampling

Vi tenker oss at vi ønsker å måle spenningen på et batteri. Vi tar da et voltmeter som f.eks. gir oss en verdi lik 4,32 V. Om vi kunne måle nøyaktig nok, kunne vi tatt med flere siffer etter komma.

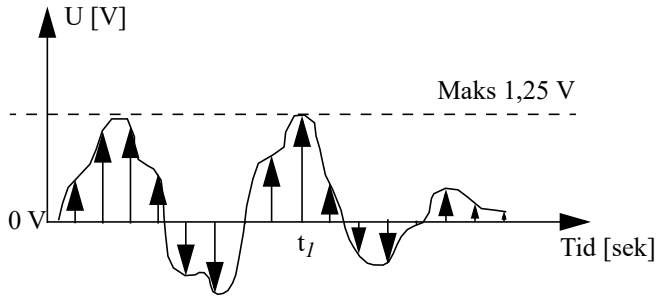
I stedet for å måle spenningen på et batteri, kan vi tenke oss å måle spenningen til et lydsignal eller en sinustone. Disse vil være kontinuerlige og varierende signaler som i prinsippet kan inneholde alle mulige signalnivåer, eller om vi omdanner dem til elektriske signaler, kan inneholde alle mulige spenningsnivåer.



Figur 6.1 Eksempler på kontinuerlige og varierende signaler.

Dersom vi på et gitt tidspunkt måler amplitudeverdien til signalet (*øyeblikksverdien*), kan vi f.eks. tenke oss at vi måler verdien 1,321V. Vi kan til og med tenke oss at vi måler kontinuerlig. Da vil vi få en strøm av tall som endrer seg hele tiden i takt med signalnivået (*øyeblikksverdiene*).

Normalt har vi ikke behov for å måle hele tiden, men i enkelte punkter. Dette kalles å *punktprøve* eller *"sample"* signalet. på denne måten gjør vi om et kontinuerlig varierende signal (analogt signal) til en rekke enkelt verdier eller tall (digitalt signal).



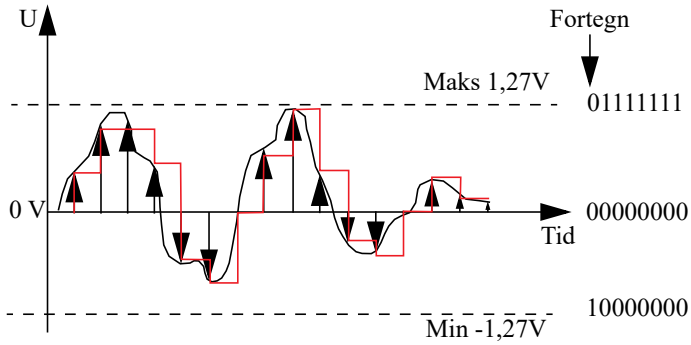
Figur 6.2 Punktprøving av et kontinuerlig signal.

En kan nå spørre seg hvor ofte må vi punktprøve et signal, for at vi senere skal være istand til å gjenskape det analoge signalet mest mulig nøyaktig.

Uregelmessige signaler, som f.eks. tale inneholder mange forskjellige frekvenser. Dersom vi ønsker å gjenskape signalet "eksakt" må vi minst punktprøve signalet dobbelte fort som den høyeste frekvensen signalet inneholder. Skal vi f.eks. punktprøve en sinus med en frekvens på 1000 Hz, må vi sample denne med en frekvens på minst 2000 sampler i sekundet. Skal vi punktprøve et talesignal som inneholder frekvenser fra 150 - 2 500 Hz, må vi minst bruke en punktprøvingshastighet på 5 000 punktprøver i sekundet.

Skal vi gjenskape et analogt signalet som er punktprøvd med en punktprøvingshastighet som er det dobbelte av den høyeste frekvensen i signalet, trenger vi et **ideelt** lavpassfilter med en båndbredde på 1000 Hz, som er nesten umulig å lage. En pleier derfor å *oversample* signalet, slik at kravet til filteret reduseres og lar seg realisere. En vil derfor punktprøve et signal på f.eks. 1000 Hz med en punktprøvingshastighet på 3000 Hz.

I eksemplet over har vi tenkt oss at vi opererer med mange siffer etter komma for å få avlesningen så nøyaktig som mulig. Dette er ofte ikke nødvendig. Det kan f.eks. være tilstrekkelig å ta med to siffer etter komma, i tillegg til at vi begrenser signalnivået på inngangen til maksimalt +/- 1,25 V. Dermed trenger vi ikke å angi signalet med mer enn 250 ulike nivåer.



Figur 6.3 Velger nærmeste nivå med en nøyaktighet på to siffer etter komma

Vi ønsker dessuten å angi signalnivået i det binære tallsystemet. Vi vet da at vi må operere med 8 bit for at vi skal kunne angi minst 250 nivået. 8 bit klarer å angi $2^8 = 256$ ulike nivåer. Noen ganger ønsker vi dessuten at høyeste digitale siffer angir fortegnet. Et ett tall i høyeste siffer angir negative verdier og en null, positive verdier.

Vi ønsker dermed at signalverdien 0 V skal angis med tallet 00000000. Signalverdien 10 mV med tallet 00000001, 20 mV med tallet 00000010 osv. Tilsvarende vil signalverdien -0.10 mV angis med tallet 10000001, verdien -20 mV med tallet 10000010 osv. som vist i figur 6.4. A.

A)		B)	
+ 1,27 V -	0111 1111	+ 1,27 V -	0111 1111
+ 0,03 V -	0000 0011	+ 0,03 V -	0000 0011
+ 0,02 V-	0000 0010	+ 0,02 V-	0000 0010
+ 0,01 V -	0000 0001	+ 0,01 V -	0000 0001
0,00 V -	0000 0000	0,00 V -	0000 0000
– 0,01 V -	1000 0001	– 0,01 V -	1111 1111
– 0,02 V-	1000 0010	– 0,02 V-	1111 1110
– 0,03 V-	1000 0011	– 0,03 V-	1111 1011
– 1,27 V -	1111 1111	– 1,27 V -	1000 0000

Figur 6.4 A) Digital representasjon av spenningsnivåer.
B) Digital representasjon av spenningsnivåer med toer's komplement

Toer's komplement

Det er imidlertid vanligere å bruke såkalt *toer's komplement* (two's complement). Også i dette tilfellet angir det mest signifikante siffer (MSB - lengst til vestre) fortegnet, 0 positivt heltall, 1 negativt heltall. Det minst positive tallet større enn 0 er i vårt tilfelle 0000 0001. Det minst negative tallet vil som vi så tidligere være 1000 0001. I toer's komplement vil imidlertid det minst negative tallet bli 1111 1111 (figur 6.4 B). Årsaken til at vi ofte bruker denne notasjonen for binære negative heltall, er at vi *slipper å tenke på tallets fortegn når vi adderer eller subtraherer binære tall*. Det vil gå av seg selv.

Toer's komplement for et negativt heltall lages ved at vi tar det tilsvarende positive binære heltallet og inverterer alle sifrene (0 => 1 og 1 => 0). Dernest legges 1 til tallet vi da får og vi har fått den negative verdien av tallet representert ved toer's komplement.

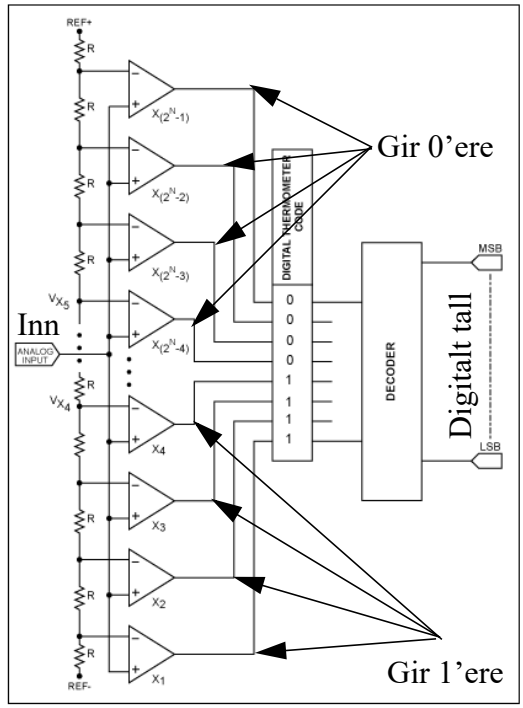
La oss se et eksempel på hvordan AD-konverteringen kan gjøres i praksis.

6.2.2 AD-konverteren

Det finnes mange måter dette kan gjøres på. Vi skal her se på en som er den enkleste og raskeste, men også den dyrest: *Flash converteren*.

Signalet føres parallelt inn til en rekke komparatorer. Tersklinngangen til den enkelte komparatoren er koblet til en rekke av seriekoplede motstander som setter terskelverdiene til hver av komparatorene til riktig nivå. Tar vi utgangspunkt i eksempelet vårt hvor spenningsområdet –1,27 – +1,27 deles opp i 255 ulike nivåer, så må forskjellen mellom hver terskelverdi være 0,01 V.

Dersom signalet inn på en komparator overskrider komparatorens terskelverdi, settes utgangen til 1 (dvs. høy spenning f.eks. 5 V). Om den er under terskelen, settes utgangen til 0 (dvs. lav spenning f.eks. 0 V). Alle komparatorer som har et referansenivå som er laver enn det innkommende signalet, gir 1'ere (nederst), og alle komparatorer som har et referansenivå høyere enn det innkommende signalet gir 0'ere (øverst). Deretter følger en dekoder som omdanner rekken av 0 og 1 til et digitalt tall som antydet i figur 6.4.



Figur 6.5 Flash A/D-konverter

En AD-konverter av denne typen som skal kunne skille mellom 256 forskjellige nivåer, må ha 256 komparatorer. Dekoderen gjør om de 256 0'erne og 1'erne til 8 bit som er tallet som angir spenningsnivået på inngangen. Om 8 bit gir for dårlig nøyaktighet, så finnes det AD-konvertere med både 10, 12, 14 og 16 bit.

AD-konverteren i CanSat er av en litt annen type, som er enklere og billigere, men ikke på langt nær så rask som denne.

7 Behandling av innsamlede data

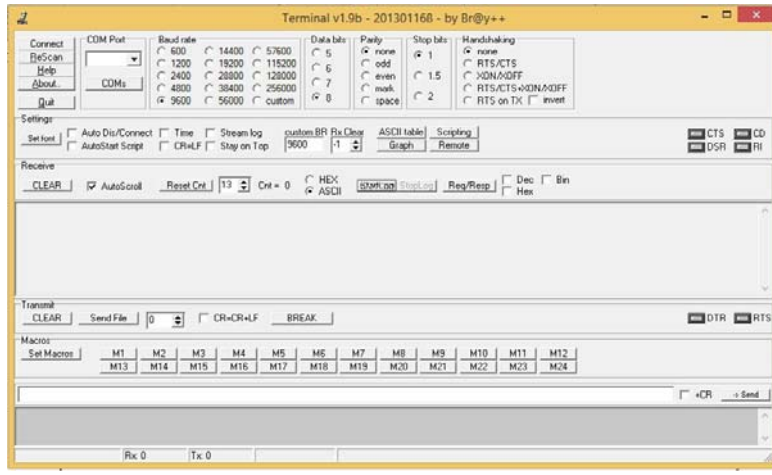
I dette kapittelet skal vi se hvordan vi kan samle opp dataene ved bakkestasjonen ved hjelp av et terminalprogram. Dernest skal vi se på grunnleggende databehandling ved hjelp av Excel.

La oss først se på bruken av et enkelt terminalprogram.

7.1 Terminalprogram for lesing av data

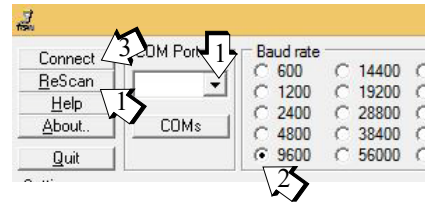
Programmet *Terminal* trengs ikke å installeres, men kan kjøres direkte fra filen som er hentet ned fra nettet: https://sites.google.com/site/terminalbpp/ (eller fra programfilen utlevert på kurset).

Når du trykket på programikonet starter programmet og man får opp følgende vindu som vist til høyre.



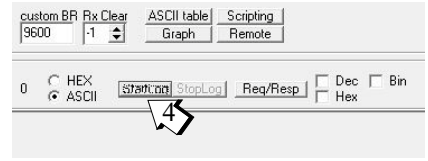
Oppsett av programmet:

1. *Velg COM-port.*
   Velg COM-porten til radioen
   Om du ikke finner den riktige porten kan du prøve å trykke *ReScan.* Da vil alle tilgjengelige porter leses inn på nytt.

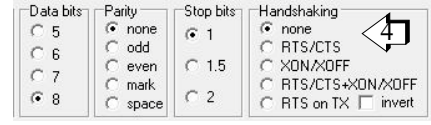


2. *Velg datahastighet*
   Velg 9600 Baud fra listen

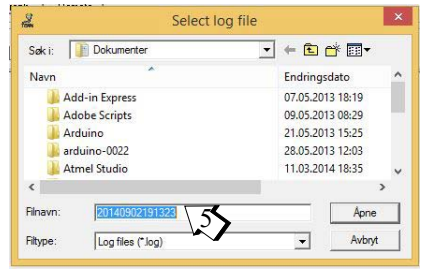
3. *Opprett kontakt*
   Trykk *Connect* for å koble opp forbindelsen mot COM-porten.



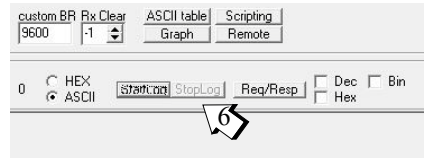
4. *Oppsett av kommunikasjon*
   Oppsett av kommunikasjonen gjøres ved å velge antall *Data bits*, *Parity*, *Stop bit* og *Handshaking*.



5. *Lagring av data*
   Trykk *StartLog* for å starte logging av data. Du vil da få spørsmål om å oppgi et filnavn. Oppgi filnavn og katalog og trykk *Åpne*.
6. *Avslutt lagring*
   Trykk *StopLog* for å avslutte logging av data.



7. *Filformat*
   Dersom dataene er hensiktsmessig organisert, kan filen leses rett inn i Excel.

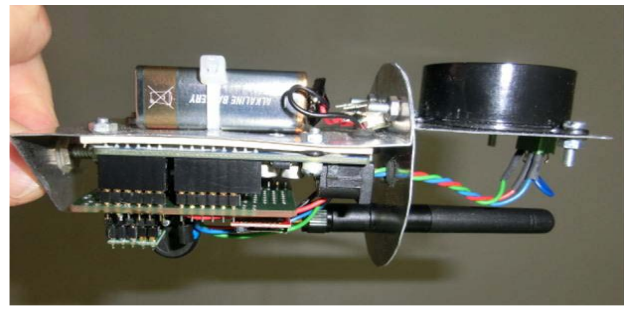


7.2 Grunnleggende behandling av data

Dette avsnittet er introduksjon til grunnleggende etterbehandling av data fra CanSat, primært ved bruk av Excel.

<Inntil videre er dette avsnittet hentet fra utgave 4 av heftet og er ikke tilpasset Teensy 3.5 og GY91>

Et praktisk og vanlig verktøy for analyse av data er Excel. I dette kapittelet skal vi se på de vanligste funksjonene for å importere og tegne ut grafer for innsamlede data fra CanSat. Til dette eksempelet bruker vi en CanSat med temperatur, trykk, 3-akse akselerometer og en CO_2-sensor som vist på bildet til høyre. Det må bemerkes at denne CanSat'en ikke tilfredsstiller krav til normert lengde.



7.3 Skrive data til file

7.3.1 Lagre rådata

Dersom man ønsker å redusere sannsynligheten for å miste data pga. feil i programmeringen, vil det være fornuftig å sende over, eller lagre rådata. Behandlingen av dataene kan likevel lett gjøres i etterbehandlingen i Excel. Overfører man beregnede data og man er så uheldig å regne feil i CanSat'en før data lagres eller overføres til bakkestasjonen, så er dataene tapt for alltid.

For å kalibrere og teste ut CanSat'en kan det være greit å la Arduino'en regne om til gjenkjennbare verdier. Dette er ikke til hinder for at man overfører eller lagrer rådata.

Følgende gir noen anbefalinger for organisering av data:

7.3.2 Signatur

Skriv inn en enkel signatur på hver linje slik at man er sikker på at de dataene man samler inn kommer fra den ønskede CanSat'en. Dersom data skal overføres på radio og det er flere samtidige brukere, kan man risikere at man tar inn data fra naboens CanSat i stedet for sin egen. Dette vil man lett oppdage når man har satt sin signatur på dataene.

7.3.3 Tidsangivelse

Inkluder en teller som angir tiden når dataene samles inn. En teller kan være grei dersom man har god kontroll på tidspunktet for hver gang telleren skrives til filen. En bedre måte er å skrive inn tiden fra Arduino'en ble startet (ev. restartet). Funksjonen:

```
millis();
```

returnerer en verdi som er lik antallet millisekunder siden start/restart. Denne funksjonen vil gå ut over sitt område først etter ca. 50 dager, hvilket skulle holde for de fleste formål.

7.3.4 Skilletegn

Verdiene som skrives til fil skilles med et skilletegn. Her kan man i prinsippet bruke ulike tegn f.eks.: < > ; tab. Det kan imidlertid være fornuftig å unngå "." og "," da disse gjerne brukes som desimalpunkt. Semikoloen kan derfor være et greit tegn å bruke.

7.3.5 Den endelige datafilen

I filen under har vi kun brukt en teller og "," som skilletegn. I dette tilfellet går dette greit fordi vi vet at det tas én punktprøve hvert 4. sekund og vi overfører rådata uten komma.

```
Serial.print(teller);          // Teller tidsintervaller/målinger
Serial.print(", ");
Serial.print(digValuePres);    // Trykk digital rådataverdi
Serial.print(", ");
Serial.print(digValueTemp);    // Temperatur digital rådataverdi
Serial.print(", ");
Serial.print(digValueCO2);     // CO2 innhold digital rådataverdi
Serial.print(", ");
Serial.print(digValueXAksl);   // Akselerometer digital rådataverdi x
Serial.print(", ");
Serial.print(digValueYAksl);   // Akselerometer digital rådataverdi y
Serial.print(", ");
Serial.println(digValueZAksl); // Akselerometer digital rådataverdi z
```

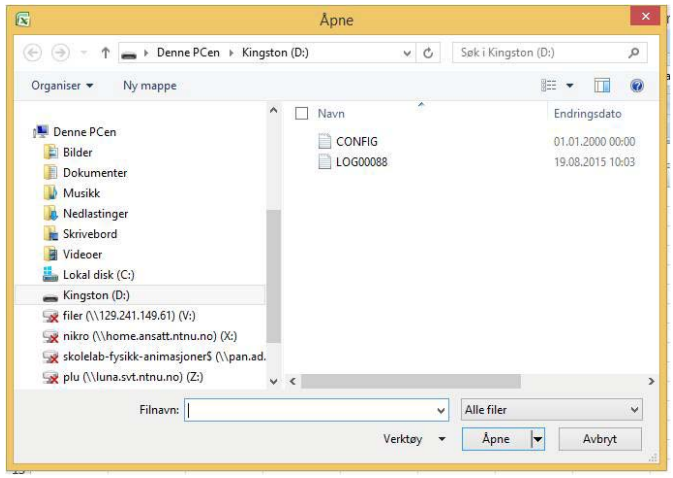
Figuren under viser den endelige filen:

```
2, 837, 56, 566, 346, 408, 408
3, 837, 56, 566, 349, 408, 409
4, 837, 56, 562, 347, 407, 407
5, 837, 56, 564, 346, 407, 407
6, 837, 56, 566, 347, 408, 408
7, 838, 56, 575, 345, 409, 409
8, 837, 56, 570, 346, 408, 408
9, 837, 56, 567, 347, 408, 408
10, 837, 56, 567, 347, 408, 408
11, 838, 56, 570, 344, 409, 409
12, 838, 56, 586, 347, 408, 408
13, 838, 56, 592, 347, 407, 407
14, 837, 56, 587, 346, 409, 408
```

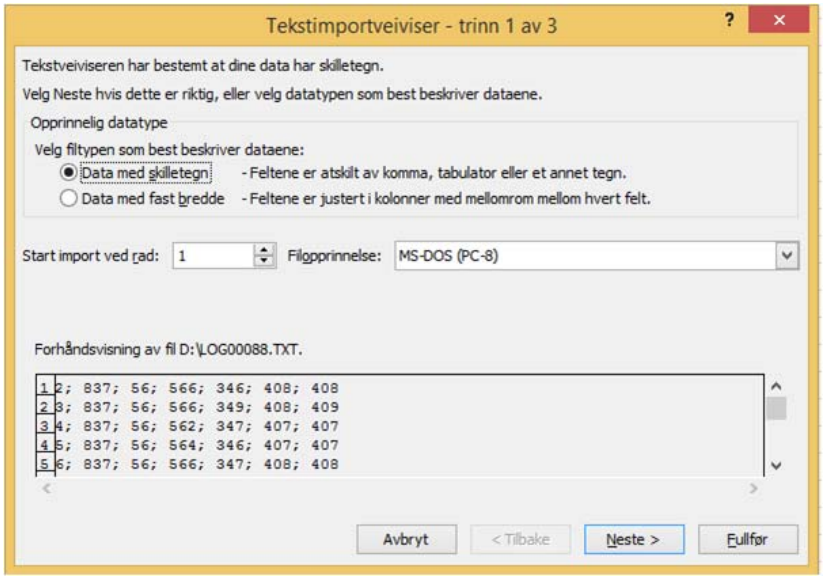
7.4 Importer data til Excel

Importering av data i Excel kan gjøres på ulike måter. En måte som fungerer er å hente inn filen med "Open". Da vil man ledes gjennom følgende vinduer for at dataene skal bli formatert på ønsket måte.

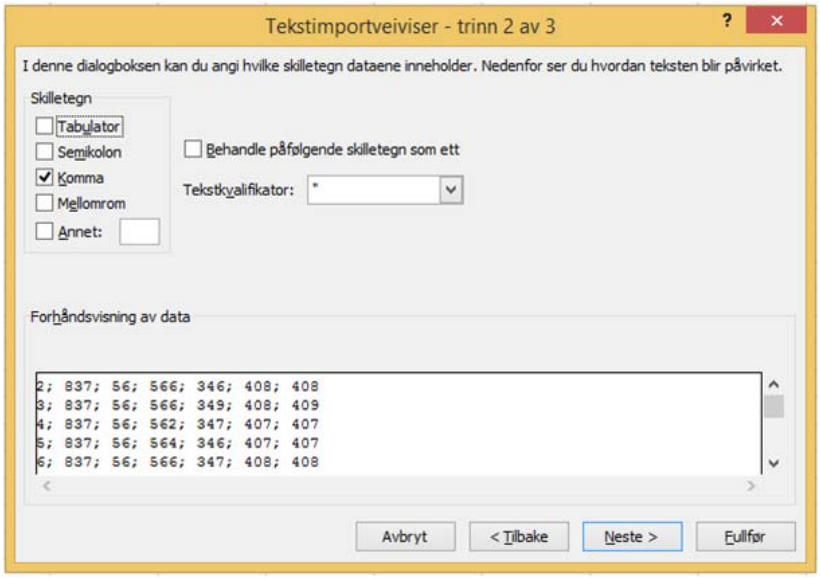
Velg ønsket fil:



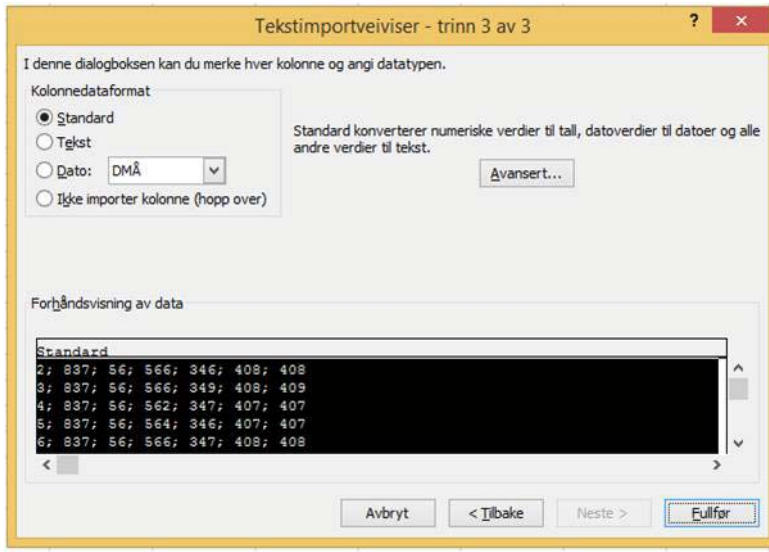
Velg “Data med skilletegn” og bestem fra hvilken rad du ønsker å starte å importere data fra. Trykk så “Neste”:



Velg hvilken type skilletegn som er benyttet. I vårt tilfelle har vi benyttet “komma”.



Tilslutt velges hvilken type data det gjelder, i vårt tilfelle er det standard tallformat. Dersom vi skal importere flyttall (med komma), velges desimalpunktet, "," eller "." ved å velge "Avansert".

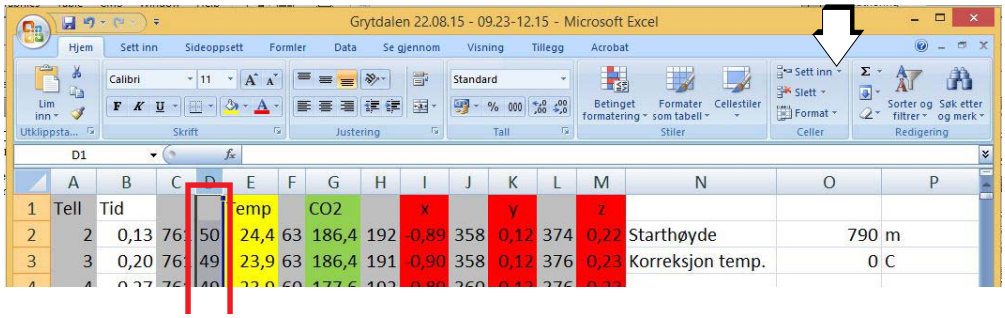


Dermed skal dataene være importert i regnearket.

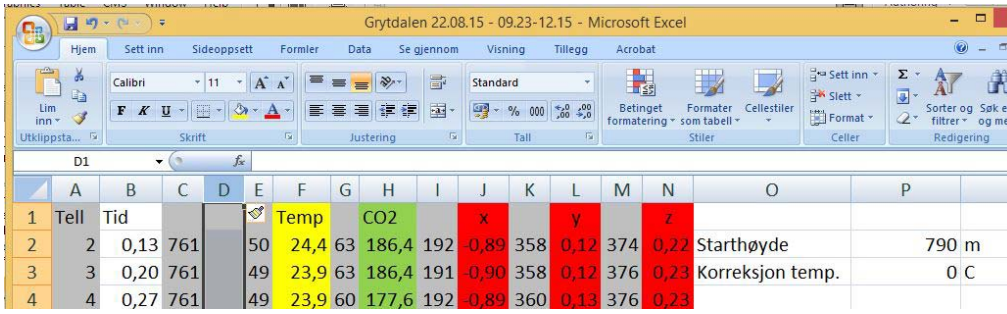
	A	B	C	D	E	F	G
1	2	761	50	63	192	358	374
2	3	761	49	63	191	358	376
3	4	761	49	60	192	360	376
4	5	761	49	58	190	359	375
5	6	761	49	63	192	359	375
6	7	760	49	66	191	359	375
7	8	761	50	63	189	359	374
8	9	761	51	63	191	358	374
9	10	761	51	65	191	360	376
10	11	761	51	68	191	358	374
11	12	761	52	66	192	359	376
12	13	761	52	68	193	358	375
13	14	761	52	71	181	343	362
14	15	761	51	71	107	336	292
15	16	760	50	66	155	360	334
16	17	761	50	63	161	344	296
17	18	761	49	55	156	326	324
18	19	760	49	55	156	328	318
19	20	760	49	60	165	324	330
20	21	761	49	60	168	328	315
21	22	760	49	65	181	346	315
22	23	758	49	61	169	340	308
23	24	754	48	65	145	309	320

7.5 Klargjøring av data

Det neste som må gjøres er å konvertere data fra digitale verdier via spenning til fysisk størrelse. Det er nødvendig å opprette en ny kolonne eller benytte en ledig for hver ny størrelse som skal beregnes. Vi har valgt å opprette en ny kolonne for trykk til høyre for de digitale trykkverdiene. I så fall merkes kolonnene ved å trykke på bokstavmarkeringen over kolonnen til høyre for der vi ønsker å opprette en ny kolonne (merket med rødt i figuren under). Velg så pilen ved siden av "Sett inn" (se pilen på figuren under) og velg "Sett inn arkkolonner".



Dermed opprettes en ny kolonne.



7.6 Legg inn formler

7.6.1 Trykk

Vi skal nå legge inn omregningsformelen fra den digitale verdien for trykket i kolonne C og plassere verdien i den nye kolonne D. Først regner vi om fra digital verdi til spenning (U_p):

$$U_p = 5V*C2/1023 \quad (7.1)$$

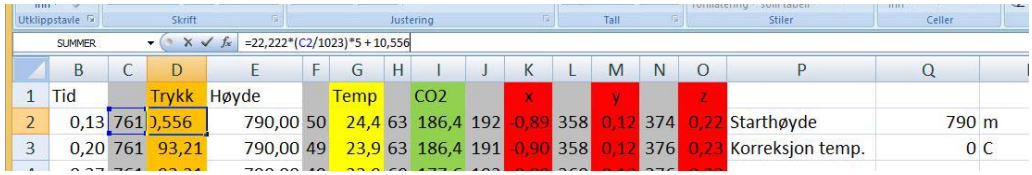
Deretter fra spenning til trykk (p) i henhold til formelen i databladet:

$$p = 22{,}222*U_p + 10{,}556\ [kPa] \quad (7.2)$$

Dersom vi kombinerer de to formlene får vi:

$$p = 22{,}222*(C2/1023)*5 + 10{,}556 \quad (7.3)$$

Som legges inn i funksjonsfeltet i regnearket.



	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
1	Tid		Trykk	Høyde		Temp		CO2		x		y		z			
2	0,13	761	0,556	790,00	50	24,4	63	186,4	192	-0,89	358	0,12	374	0,22	Starthøyde	790	m
3	0,20	761	93,21	790,00	49	23,9	63	186,4	191	-0,90	358	0,12	376	0,23	Korreksjon temp.	0	C

Deretter kopieres feltet D2 til de resterende feltene i kolonne D.

Slik gjør man for alle formler.

7.6.2 Høyde

Høyden (h) baserer seg på kolonnen med verdier for trykk, temperaturen ved start (T_1) og høyden ved start (h_1):

$$h = \frac{T_1}{a}\left(\left(\frac{p}{p_1}\right)^{-\frac{aR}{g_0}} - 1\right) + h_1$$

- h Beregnet høyde i meter
- h_1 Starthøyde i meter (Q2)
- T Temperatur i Kelvin
- T_1 Starttemperatur i høyden h_1 (G2)
- a Temperaturgradient, foreslått verdi -0,0065 K/m
- p Trykk i Pa (D2)
- p_1 Trykk i Pa ved starthøyden (D2)
- g_0 Tyngdeakselerasjonen 9,81 m/s2
- R Den spesifikke gasskonstant 287,06 J/kg K

Dette kan i vårt tilfelle skrives på følgende måte:

$$h = (271+\$G\$2)/(-0{,}0065)*(((D2)/\$D\$2)^(-(-0{,}0065*287{,}06)/9{,}81)-1)+\$Q\$2 \quad (7.4)$$

hvor parameterne er skrevet inn i lista over.

7.6.3 Temperatur

Etter å ha regnet om den digitale verdien for temperaturen til spenning, så settes den inn i formelen for omregning fra spenning til temperatur:

$$T = V_T * 100\ (+\Delta T) \quad (7.5)$$

hvor:

- T Er temperaturen i °C
- V_T Er avlest spenning som beregnes ut fra den digitale verdien (F2)
- ΔT Er ev. korreksjon i temperaturen (Q3)

Satt inn i formelfeltet blir denne i vårt tilfelle:

$$= ((F2/1023)*5)*100+\$Q\$3 \quad (7.6)$$

Legg merke til formen Q3, som betyr at denne cellen skal holdes fast lik Q3 selv om man kopierer formelen ned over hele kolonnen.

7.6.4 CO_2

I vår CanSat har vi en CO_2-måler som erstatter NTC-motstandens plass. Omregningen fra spenning til CO_2 i ppm kan i følge databladet uttrykkes som:

$$CO_{2ppm} = FS\ V_{out}/V_{supply} \quad (7.7)$$

hvor:

- FS Fullskala konsentrasjon (2000 ppm)
- V_{out} Målt spenning på utgangen (H2 er den digitale spenningsverdien)
- V_{supply} Forskyningspenning som tilføres sensoren (3.3 V)

$$CO_{2ppm} = 2000*((H2/1023)*5)/3{,}3 \quad (7.8)$$

7.6.5 Akselerasjon

Akselerometeret måler akselerasjon i x-, y- og z-retning. Dataene fra hver av retningene beregnes og legges i hver sin kolonne. Akselerasjonene (A_X, A_Y, A_Z) beregnes ut fra de tre målte spenningene V_{AX}, V_{AY} ogV_{AZ}. Vi kan sette opp følgende sammenhenger:

$$A_X = (V_{AX} - 1{,}65 + \Delta V_X)/S_g\ [m/s^2] \quad (7.9)$$

$$A_Y = (V_{AY} - 1{,}65 + \Delta V_Y)/S_g\ [m/s^2] \quad (7.10)$$

$$A_Z = (V_{AZ} - 1{,}65 + \Delta V_Z)/S_g\ [m/s^2] \quad (7.11)$$

hvor

- A_X, A_Y, A_Z er den beregnede akselerasjonen i x-, y- og z-retning
- V_{AX}, V_{AY}, V_{AZ} er den målte spenningen fra hver av akselerometerne i x-, y- og z-retningen (de digitale verdiene konverteres til spenning).
- ΔV_X, ΔV_Y og ΔV_Z er korreksjonsspenningen for hver av retningene

- S_g er følsomheten som er 800mV/g i området +/– 1,5 g

Dersom vi setter inn omregningsformler fra digitale verdier til spenning kan vi skrive:

$$\text{Akselerasjon x-retning} = (((J2/1023)*5)-1{,}65)/0{,}8 \qquad (7.12)$$

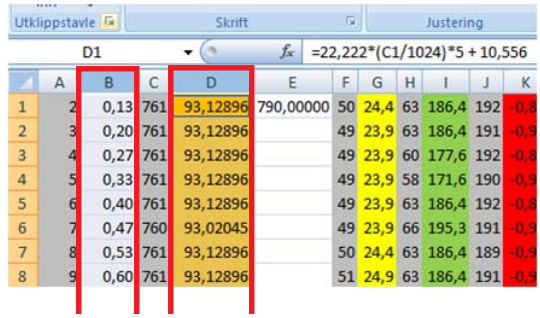
I vårt tilfelle inneholder kolonnen J de digitale verdiene for spenningen i x-retningen. Det er ikke lagt inn noen korreksjonsverdi i ligning (7.12).

Når alle verdiene er beregnet og lagt i en kolonne kan vi lage grafer.

7.7 Lage grafer

Slik kan man lage grafer:

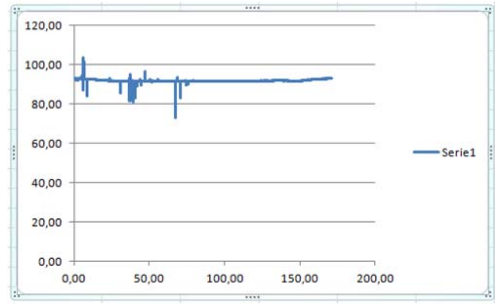
1. Merk de to kolonnene som skal representeres av en graf.
   Det kan være lurt å lage en overskrift på kolonnene. Overskriften vil automatisk tilordnes seriene i grafen.



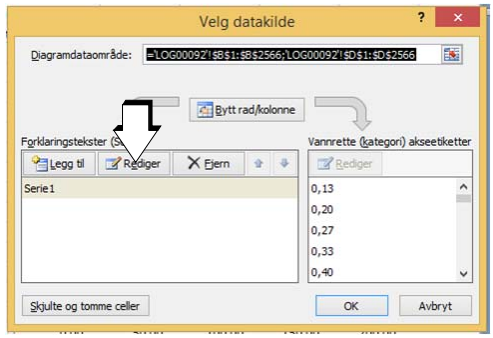
	A	B	C	D	E	F	G	H	I	J	K
1	2	0,13	761	93,12896	790,00000	50	24,4	63	186,4	192	-0,8
2	3	0,20	761	93,12896		49	23,9	63	186,4	191	-0,9
3	4	0,27	761	93,12896		49	23,9	60	177,6	192	-0,8
4	5	0,33	761	93,12896		49	23,9	58	171,6	190	-0,9
5	6	0,40	761	93,12896		49	23,9	63	186,4	192	-0,8
6	7	0,47	760	93,02045		49	23,9	66	195,3	191	-0,9
7	8	0,53	761	93,12896		50	24,4	63	186,4	189	-0,9
8	9	0,60	761	93,12896		51	24,9	63	186,4	191	-0,9

2. Velg type representasjon fra menyen "Sett inn" og velg f.eks. punktdiagram med linjer mellom de ulike punktene i diagrammet:

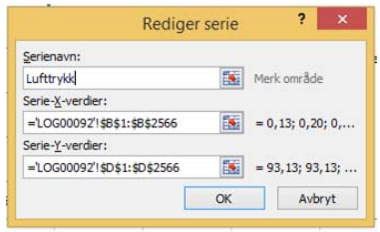
3. Når dette er gjort vil diagrammet tegnes ut som vist på figuren til høyre.



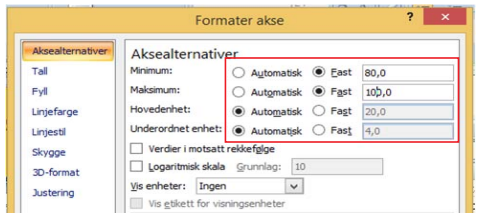
4. Dersom vi skal legge inn et navn på grafen i figuren gjør vi det ved å høyre-klikke på rammen av figuren og velge "Merk data" i menyen som framkommer.



5. Merk "Serie 1" og trykk rediger. Skriv inn nytt "Serienavn".

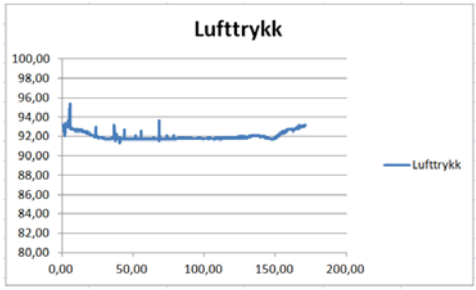


6. Dersom man ønsker å endre på akseverdiene, høyre-klikker man på aksen og får opp boksen vist til høyre. F.eks. kan det være ønskelig å sette faste min. og maks. verdier på den vertikale aksen som vist på figuren til høyre.



7. Det endelige diagrammet kan dermed bli som vist i figuren til høyre.

Slik kan man gjøre for alle sammenhenger man ønsker å lage grafer av.



8 Oppskyting

8.1 Ballongslipp

Et alternativ til rakettoppskyting er ballongslipp. En ballong med helium ca. 150 cm i diameter før slipp, er mer enn tilstrekkelig for å løfte to CanSat'er til 1000 meter om man har tilstrekkelig med tråd. En fjernstyrt utløsermekanisme kan benyttes for å utløse CanSat'ene når man når ønsket høyde. Med en fallhastighet på ca. 8–12 m/s vil falltiden ved vertkalt fall ta ca. 80–130 sek. i vindstille vær. En kan oppnå rikelig med resultater selv ved 300–400 meters høyde. Ev. kan man redusere fallhastigheten.

8.1.1 Utstyr for ballongslipp

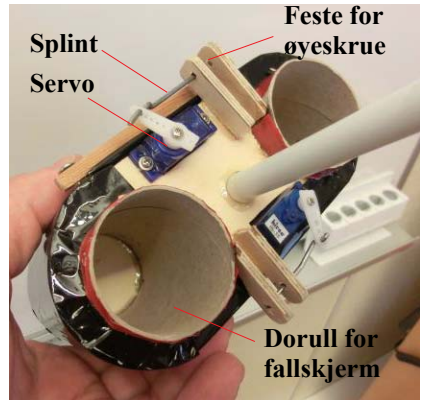
Snora kan f.eks. vindes opp på et fiskehjul for linefiske eller pilk som vist på figuren til venstre. Men enklere utgaver kan brukes. F.eks. en gammel bilfelg lagret opp og påmontert sveiv.

Ballongen er en helium ballong for innsamling av metrologiske data. (f.eks. Totex TX350). Skoler som ønsker å arrangere sitt eget ballongslipp kan henvende seg til NAROM for kjøp av ballong. Pris på forespørsel.

I tillegg trengs helium, som kan kjøpes hos en forhandler av gass, f.eks. AGA eller lignende. Disse selges ofte på stålflasker som returneres etter bruk. En bør kjøpe et munnstykke for fylling av ballonger. Dette kan gjenbrukes hver gang en trenger helium. Slike koster noen hundre kroner. En 50 liters flaske med 200 bars trykk (10 000 liter), holder til 5 - 6 fyllinger.

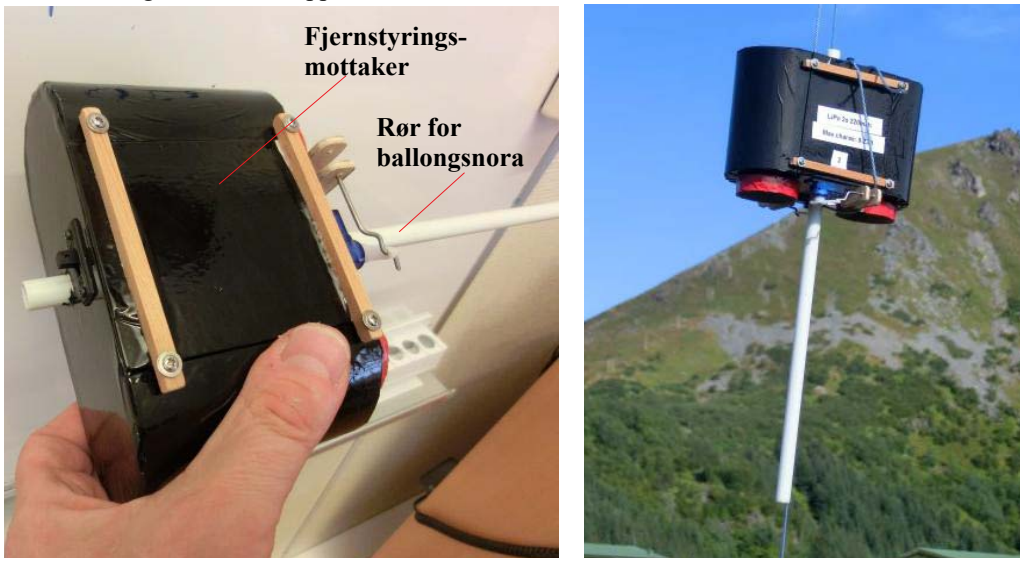
8.1.2 Slippmekanisme

Her er det mange muligheter. Det enkleste er å la utstyret følge ballongen opp og ned. En kan dermed til en hver tid ha god kontroll på hvor CanSat'en befinner seg.



Dersom man ønsker å gjennomføre selve slippet, kan man bruke en ekstra snor for å dra ut en splint slik at CanSat'en slippes. Dette krever at det lages en passende utløsermekanisme til å feste CanSat'en til ballongen.

Eller man kan lage en fjernstyrt utløsermekanisme. En slik kan bygges med deler for fjernstyring av modellfly. Bildet over til høyre viser utløsermekanismen som ble benyttet under lærerkurset ved NAROM august 2011. Denne kan slippe to CanSat'er uavhengig av hverandre. De to servo-ene betjener hver sin splint som holder CanSat'ene oppe. Fallskjermen pakkes inn i et rør (dorull) slik at festesnorene til fallskjermen ikke floker seg. Det er særdeles viktig at den pakkes slik at den folder seg ut idet den slippes.

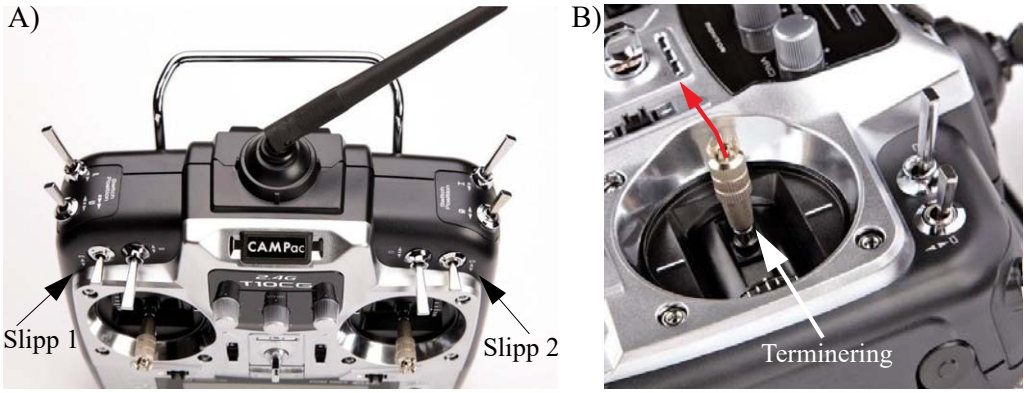


Figur 8.1 Fjernstyrt slippmekanisme for to CanSat'er

Fjernstyringsenheten som ble benyttet i dette tilfellet var en FutabaT10CAG 10 kanals sender og en F6106 HFC 6 kanals mottaker fra samme firma[23].

Figur 8.2 Fjernstyringsenheten FutabaT10CAG ti-kanals sender og en F6106 HFC seks-kanals mottaker fra samme firma.

Styringen av slipp 1 (kanal 5 og kanal 6) og slipp 2 var tilkoblet de to bryterne som antydet på figur 8.3A.Ved a skyve den høyre spaken mot venstre og holde den der i 5 sek. utløses termineringen, figur 8.2B.



Figur 8.3 A) Slipp 1 og slipp 2, hendelen presses forover ved slipp og trekkes bakover ved festing av CanSat. B) Ved a skyve den høyre spaken mot venstre og holde den der i 5 sek. utløses termineringen.

Utstyret er i salg hos ELEFUN (www.elefun.no). Her finnes også billigere varianter av fjernstyringsutstyr.

23.Thomas Gansmoe ved NAROM har utviklet utstyret og valgt fjernstyringsenhet.

8.1.3 Reglement for oppsending av forankret ballong

Regler for oppsending av forankrede ballonger finnes i ***BSL F 1-1 Lufttrafikkregler, Kapittel IX. Forankrede ballonger***. Dette regelverket finnes på følgende nettsted: http://www.lovdata.no/cgi-wift/ldles?doc=/sf/sf/sf-20030207-0252.html. Regelverket finnes i vedlegg C. Det viktigste er at nærmeste enhet av Luftfartstilsynet eller NOTAM-kontor (ofte tårnet ved nærmeste flyplass) må ***varsles senest 14 dager før oppsendingen***. Strengere regler gjelder dersom oppsendingen skal skje innen en radius på 10 km fra flyplass.

Meldingen skal inneholde:

a) navn, adresse og eventuelt telefonnummer til den som har ansvaret for oppsendingen,

b) sted/posisjon for forankringen,

c) ballongens maksimale høyde,

d) dato, tid og varighet for oppsendingen,

e) opplysninger om hvordan ballongen med forankringsutstyr er varselmerket.

Dessuten skal ballongen utstyres med en termineringsmekanisme dersom fortøyningen skulle ryke. Ballongen skal dessuten merkes slik at den er lett synlig. Vi brukte oljebasert rød spraymaling.

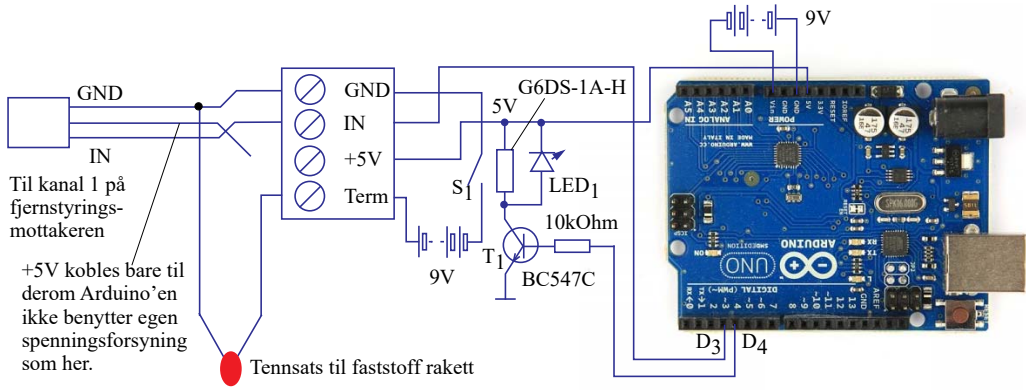
Ved slipp skal tårnet på nærmeste flyplass varsles, likeså når slippet avsluttes. Det er primært helikoptertrafikk som kan bli hindret av forankrede ballonger.

Jeg varslet Svein O. Aasen (Svein.Aasen@caa.no) ved Luftfartstilsynet pr. e-post og fikk et skriftlig tillatelse tilbake med pålegg om å følge reglene i lovverket.

8.1.4 Termineringsmekanisme for forankrede ballonger

Dersom forankringen løsner og ballongen stiger til værs, skal den utstyres med en automatisk eller fjernstyrt termineringsmekanisme.

Thomas Gansmoe (NAROM) har utviklet et termineringssystem basert på en Arduino UNO koblet til en av kanalene (kanal 1) til fjernstyringsmottakeren omtalt foran. På bakgrunn av erfaringene fra denne løsningen har ***Nils Kr. Rossing*** bygget opp følgende utstyr for montasje på toppen av utløsermekanismen:



Figur 8.4 Koblingsskjema for terminering av ballong.

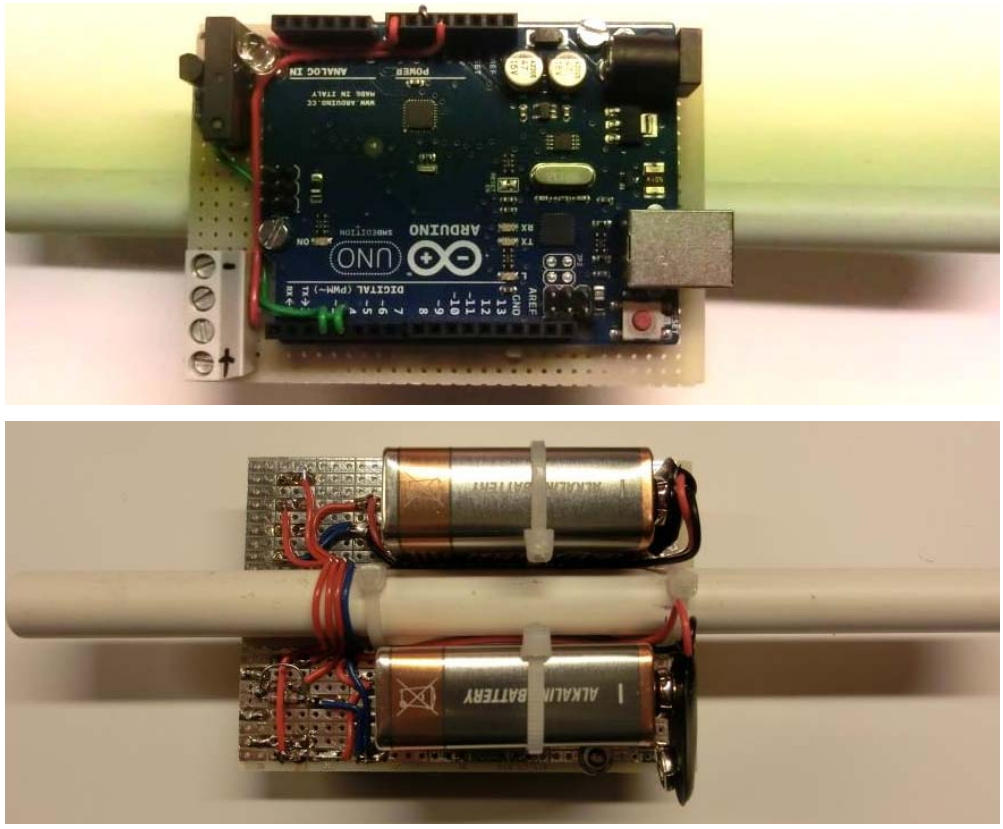
Termineringsanordningen benytter som tidligere en Arduino UNO som dekoder signalet fra fjernstyringsmottakeren via digital inngang D_3. Når programvaren har detektert signal på inngangen i 5 sekunder, går den digitale utgangen D_4 høy slik at transistoren T_1 åpner og reléet S_1 kobler inn og tennsatsen får strøm. En LED er koblet over spolen i reléet for å kortslutte "flyback" spenninger, som ellers ville lunne ødelegge transistoren. I vår løsning har vi valgt å benytte et eget 9V batteri til Arduino'en. Dette kan unngås ved at man benytter 5V fra fjernstyringsmottakeren og kobler den til 5V utgangen på Arduino'en. En slik løsning er noe ureglementert da spenningen går uregulert inn på Arduino-kortet og kan dermed skade kortet om spenningen blir for høy. Vi benytter en egen spenningskilde til tennsatsen da denne belaster batteriet temmelig hardt. På den måten unngår vi at et kraftig spenningsfall i forbindelse med terminering påvirker funksjonen til den øvrige elektronikken.

Figur 8.5 Oppkobling av termineringsmekanismen.

Arduino'en programmeres med programmet: *BallongSquid.ino*, se vedlegg B.6[24].

Termineringskretsen ble festet til et elektrikerrør slik at den kunne smettes inn på snøret til ballongen på oversiden av slippmekanismen som vist på de neste bildene.

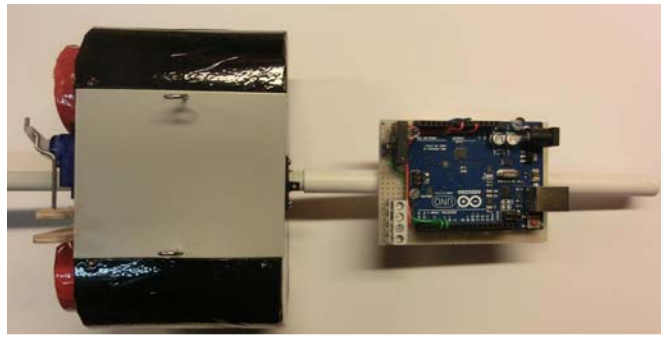
24. BallongSquid.ino er skrevet av Thomas Gansmoe NAROM. Programlistingen kan fås ved henvendelse til Thomas Gansmoe eller Nils Kr. Rossing



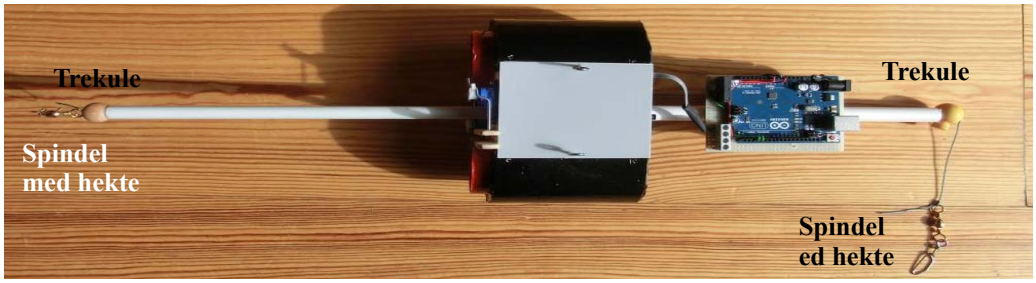
Figur 8.6 Termineringselektronikken montert på et elektrikerrør slik at den kan smettes inn på snora til ballongen.

Det hele monteres sammen med slippmekanismen som vist på figur 8.7.

For å lette sammenkoblingen av de ulike enhetene, kan man benytte en kort nylonsnor og kraftige spindler med hekter på over- og undersiden av elektronikken slik at de lett kan klipses inn på snora som forankrer ballongen.



Figur 8.7 Termineringselektronikken montert sammen med slippmekanismen.



Figur 8.8 Termineringselektronikken monteres på toppen av utløsningsmekanismen. En snor med trekuler og spindler i begge ender føres gjennom rørene som vist.

Som termineringsanordning ble benyttet tennsatsen til en rakettmotor. Denne ble tapet inntil ballonggummien ved innblåsningsrøret til ballongen, figur 8.9. I tillegg kan en benytte en mindre rakettmotor koblet til tennsatsen og legge det hele inne i ballongen før den fylles med gass. Dette vil sikre at ballongen blir terminert om nødvendig.

Figur 8.9 Tennsatsen tapes inntil ballongen.

Ballongen fylles med helium til den har et løft på mellom 1,2 – 1,5 kg. Dersom det er litt vind er det lurt å øke løfteevnen. Med større løft blir ballongen mindre følsom for avdrift Ballongen merkes med rød farge for lettere å bli sett.

Figur 8.10 CanSat'en er montert til slippmekanismen. En Cola-boks beskytter termineringsenheten.

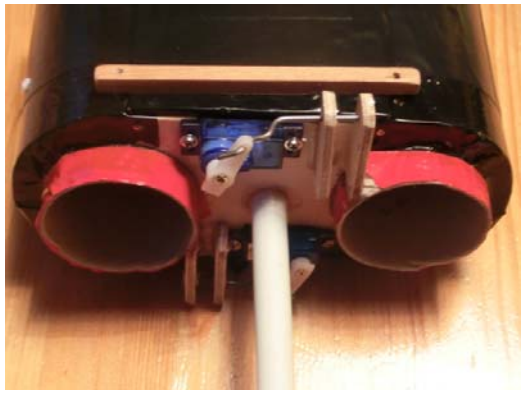
En nylonsnor spunnet opp på et fiskehjul ble montert på en stor trekloss og benyttet som forankring.

Figur 8.11 Ballongen ble forankret i en nylonsnor spunnet opp på et fiskehjul (Hitra). Ballongen ble merket med rød oljebasert spraymaling.

Føringen for nylonsnoren bør monteres tett ved spolen slik at en unngår at snora faller av spolen og kilder seg fast i opplagringspunktet.

8.2 Bygg en egen slippmekanisme

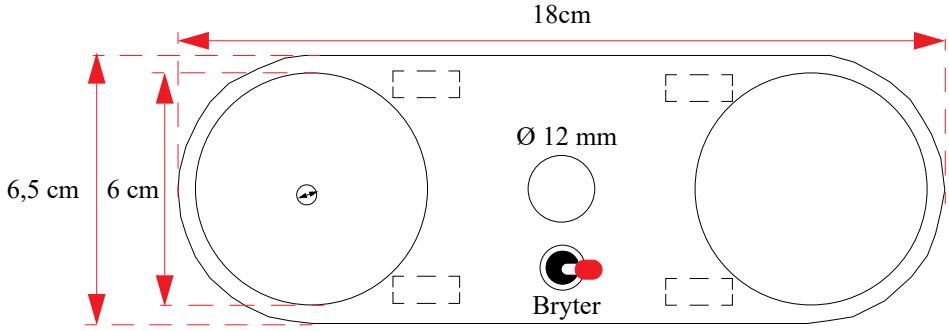
Slippmekanismen består av en spesiallaget boks med plass til to CanSat'er med fallskjerm. Fallskjermene plasseres i to papprør, en på hver side av boksen som vist på figuren under. Øyeskruen hvor fallskjermen er festet, festes i mellomrommet mellom to lister som stikker ut over kanten av boksen.



En servo skyver et stag gjennom to hull gjennom listene somvist til høyre på figuren over. Et elektrikerrør for snøret som holder ballongen går gjennom boksen.

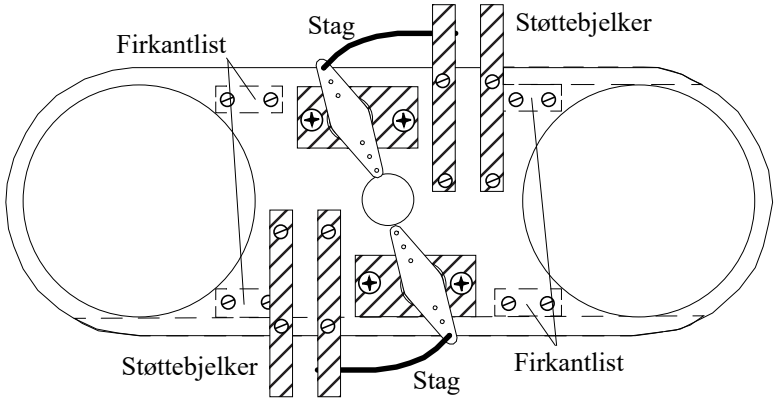
Elektronikken, radiomottakeren og batteriet er plassert i rommet mellom de to papprørene.

Figuren under viser en tegning av *topplata*. Topplata er laget av 3 mm kryssfiner. De to hullene dekkes til slutt med folie slik at de blir tette.

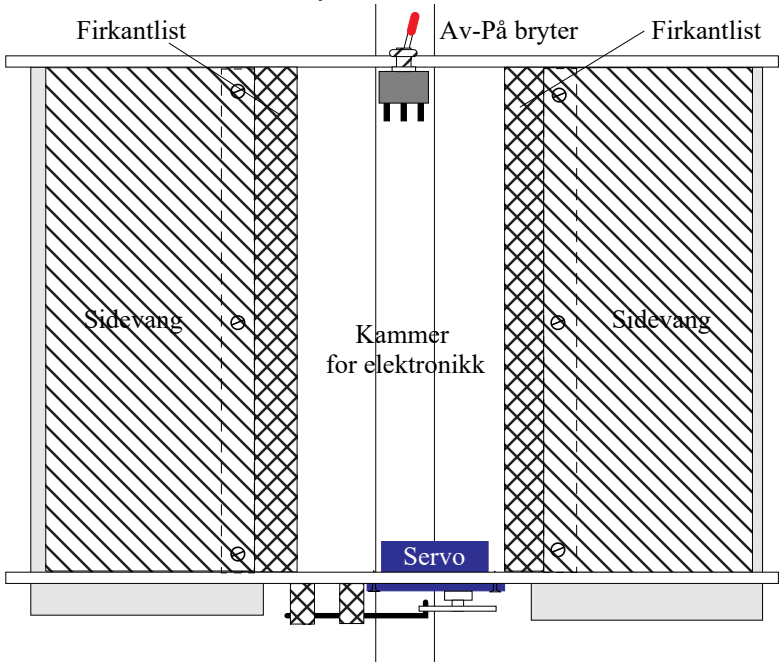


En på-av bryter monteres i topp-plata. Alternativt kan denne monteres i bunnplata, slik at den ikke er så utsatt for fuktighet.

Figuren under viser bunnplata som er laget på tilsvarende måte. I plata er det montert to *servoer* som drar i to *stag* laget av 1,5 mm ståltråd. Stagene går inn gjennom hull i to *støttebjelker* som er skrudd fast i bunnplata.

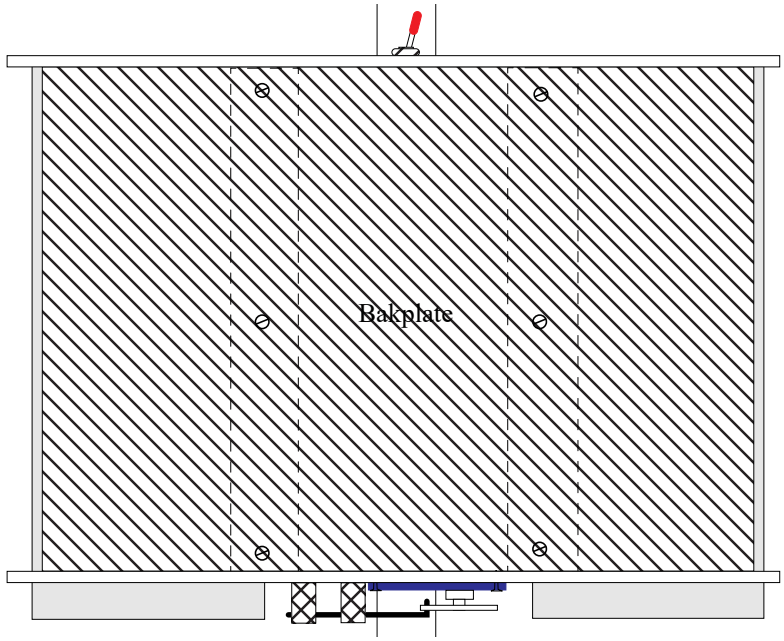


Papprørene går gjennom topp- og bunnplata. De kan også gjerne stikke 1 cm ut under bunnplata. De to platene er skrudd sammen med fire *firkantlister*.

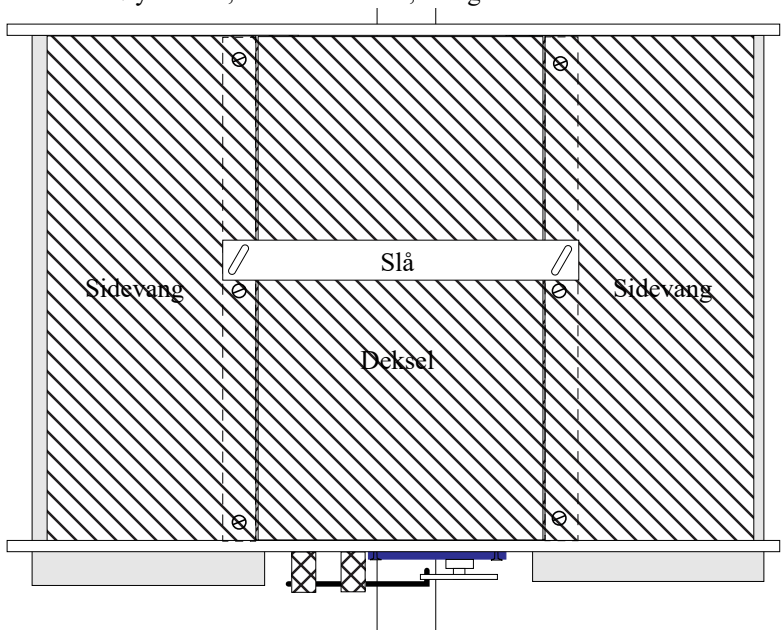


De to *sidevangene* er 3 mm kryssfiner som monteres en på hver side av åpningen inn til kammeret for elektronikken.

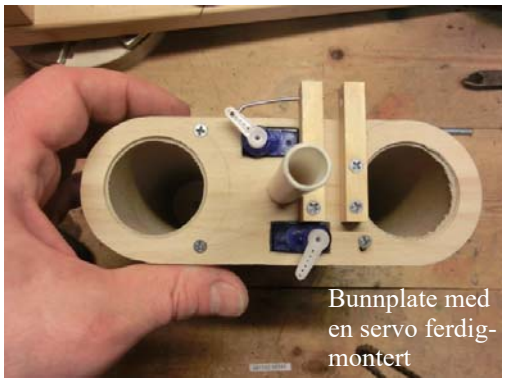
På baksiden er det montert et heldekkene *bakplate* som går ut til sidene der kromningen begynner, se figuren under.

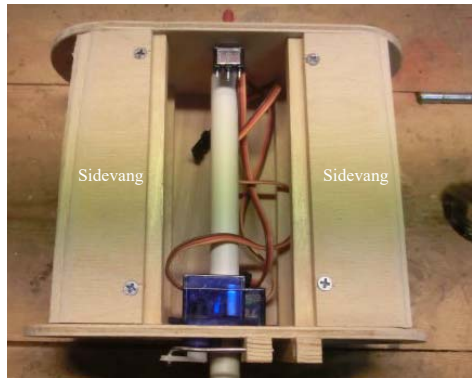


Et deksel i 3 mm finer dekker åpningen inn til kammeret med elektronikken. Dekselet kan enten festes med to eller fire øyeskruer, eller med en slå, se figuren under.



Bildene under viser slippmekanismen framstilt i tre.


Bunnplate med en servo ferdig-montert



Dekselet monter foran kammer for elektronikk

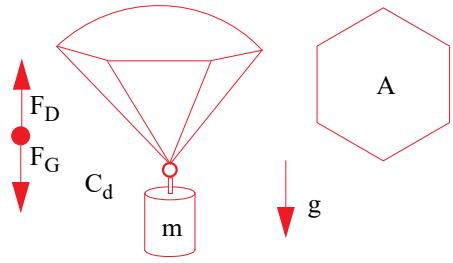
Tilslutt kles boksen utvendig med plastfolie eller laminert papir som stiftes fast i topp og bunnplatene.

Ellers brukes en standard fjernstyringsenhet med mottaker, mottakeren legges inn i kammeret og tilkobles batteriet. Servoene kobles til to av kanalene til mottakeren.

8.3 Beregning av fallhastighet

Fallhastigheten bestemmes av en rekke ulike parametere.

Vi antar at sonden inntar terminalhastigheten etter svært kort tid, slik at den faller med konstant fart omtrent fra slipptidspunktet. Når dette inntreffer vil *drag'et*, F_D, på grunn av luftmotstanden, være lik tyngdekrafta, F_G:



$$F_D = F_G \qquad (8.1)$$

$$F_G = m \cdot g \qquad (8.2)$$

Hvor:
g = Tyngde akselerasjonen – 9,81 m/s^2
m = Massen av CanSat – maks. 0,350 kg

En kan også finne en formel for *drag'et*, F_D:

$$F_D = 1/2 \cdot C_d \cdot \rho \cdot v_t^2 \cdot A \quad (8.3)$$

Hvor:
C_d = *Drag*-faktor varierer blant annet med formen på fallskjermen (kule-kalott $C_d = 1{,}5$)
ρ = Tettheten til luft (typisk $\rho = 1{,}22$ kg/m^3) varierer med lufttrykket
v_t = Terminalhastigheten
A = Tverrsnittsarealet av fallskjermen projisert ned på horisontalplanet.

Setter ligning (8.2) og (8.3) inn i ligning (8.1) og løser denne mht. terminalhastigheten, får vi:

$$v = \sqrt{\frac{2mg}{C_d \rho A}} \quad (8.4)$$

Drag-faktoren er avhengig av formen på fallskjermen. Dersom den kan settes tilnærmet lik en kuleflate vil den i teorien ha en *drag*-faktor nær $C_d = 1{,}5$. Dersom fallskjermen tilnærmet er en sirkulær flate med samme tverrsnitt som kula, vil en C_d nær 0,75 være nærmere sannheten. En fallskjerm som vist på figuren over, vil ha en *drag*-faktor som ligge et sted mellom disse verdiene[25].

Bestem *drag*-faktoren:

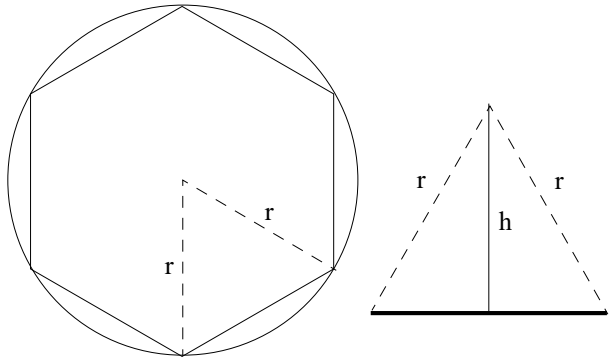
Drag-faktoren for en gitt fallskjerm kan måles ved følgende metode:

1. Mål og beregn arealet, A, til fallskjermen (projisert areal)
2. CanSat'en henges under fallskjermen og finn massen, m
3. Slipp CanSat'en i fallskjerm fra en avsats fra 5–10 meter over bakken
4. Mål tida til fallet over en gitt fallhøyde. Eller film slippet med et kamera med et kjent antall bilder i sekundet. Plasser en målestav i bildet for å bestemme dimensjonene.
5. Løs ligningen (8.4) mht C_d. Sett inn måleverdier og beregn C_d.

25.http://my.execpc.com/~culp/rockets/descent.html

Beregning av fallskjermens areal:

For å finne det totale arealet for en fallskjerm, kan vi bruke illustrasjonen til høyre som viser en sekskantet fallskjerm innskrevet i en sirkel.



Av figuren ser vi at fallskjermen består av 6 like trekanter. Det betyr at det totale arealet for denne fallskjermen er $A = 6 \cdot A_T$ hvor A_T er arealet til hver av trekantene.

For å finne arealet for den enkelte trekanten, A_T, kan vi benytte formelen:

$$A_T = \frac{sh}{2} \tag{8.5}$$

Vi kan måle lengdene *s* og *h* direkte på fallskjermen. Det totale arealet av den 6-kantede fallskjermen blir dermed:

$$A = \frac{6sh}{2} \tag{8.6}$$

Det er mulig å finne et mer generelt utrykk for beregning av arealet for en fallskjerm formet som en regulær mangekant med et vilkårlig antall kanter. For mer informasjon om beregning av arealet til fallskjermer, se:

http://www.sunward1.com/imagespara/The%20Mathematics%20of%20Parachutes%28Rev2%29.pdf

Når fallskjermens totale areal A er bestemt, kan fallhastigheten bestemmes av ligning: (8.4)

På kurset vil det bli gjennomført slipp av CanSat med fallskjerm uten modifikasjoner. Fallhastigheten kan dermed bestemmes dersom slipphøyden er kjent (for eksempel ved hjelp av data fra trykksensoren) og tiden CanSat'en bruker på fallet til bakken (hvordan kan falltiden bestemmes?). Ved å anta konstant fart, kan formelen v = s/t brukes for å beregne fallhastigheten.

Sammenlign målingene av fallhastigheten med beregningene.

9 Referanser

[1] Gunnar Stette, *Romtekologi - Del av faget Teknologi og Forskningslære (CanSat)*, NTNU juli 2011.

[2] Hovedressurskilde for Teensy
https://www.pjrc.com

[3] Teensy koblet opp mot GPS-modul NEO-6M
https://www.pjrc.com/teensy/td_libs_TinyGPS.html

[4] Datablad NTC: http://www.vishay.com/docs/29051/ntclesb.pdf

[5] Datablad BMP280: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMP280-DS001-19.pdf

[6] Datablad MPU-9250: https://store.invensense.com/datasheets/invensense/MPU9250REV1.0.pdf

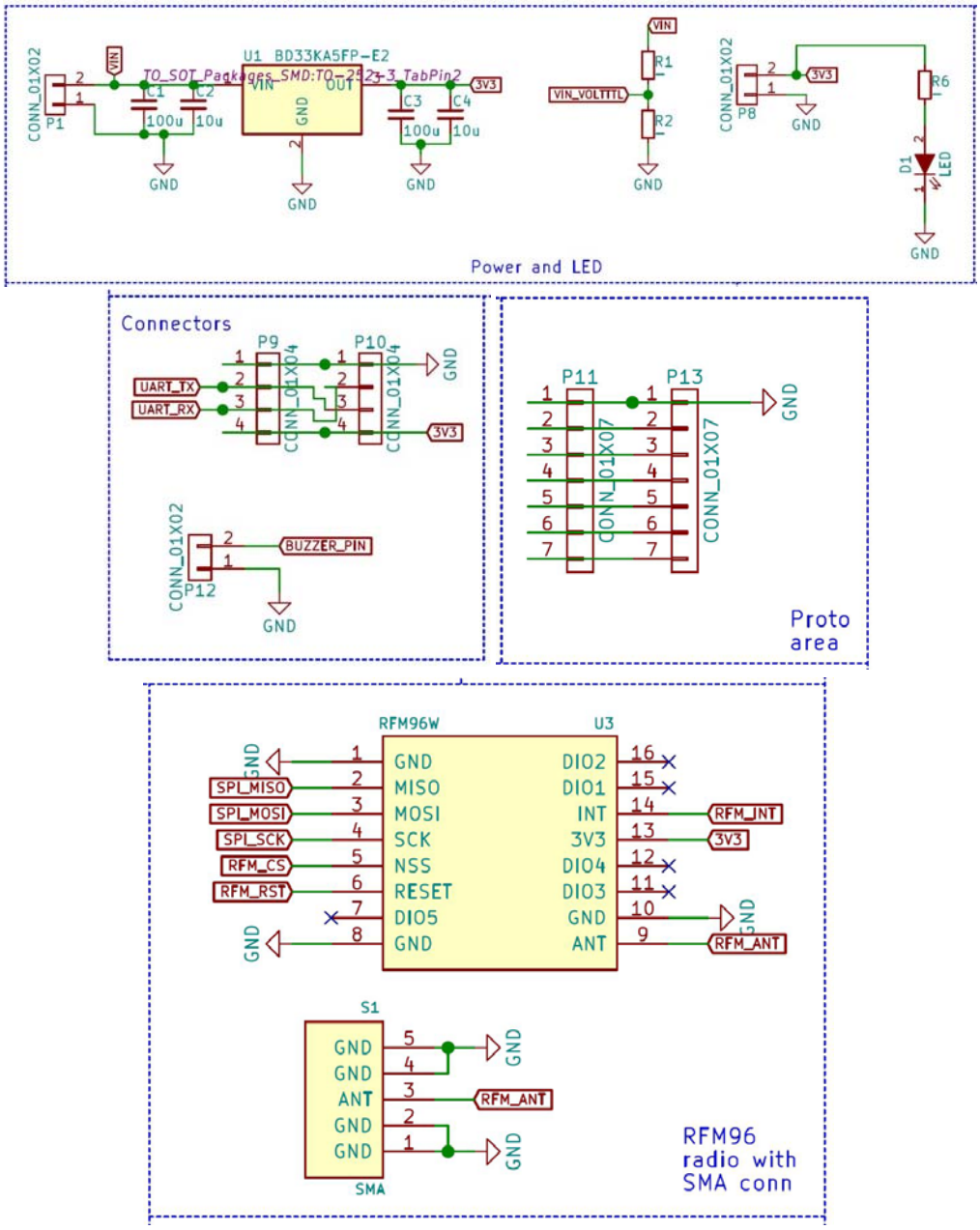
[7] Datablad AK8963: https://www.akm.com/akm/en/file/datasheet/AK8963C.pdf

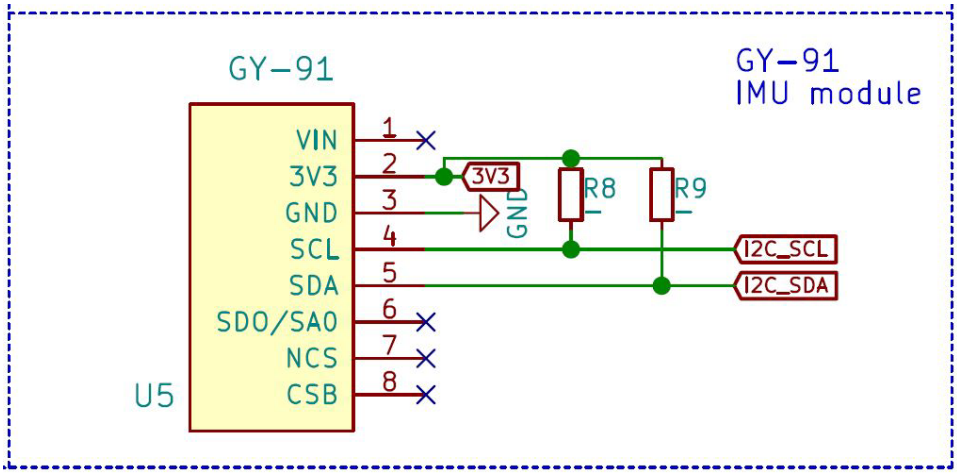
[8] Datablad NTCLE100E3103B: https://www.vishay.com/docs/29049/ntcle100.pdf

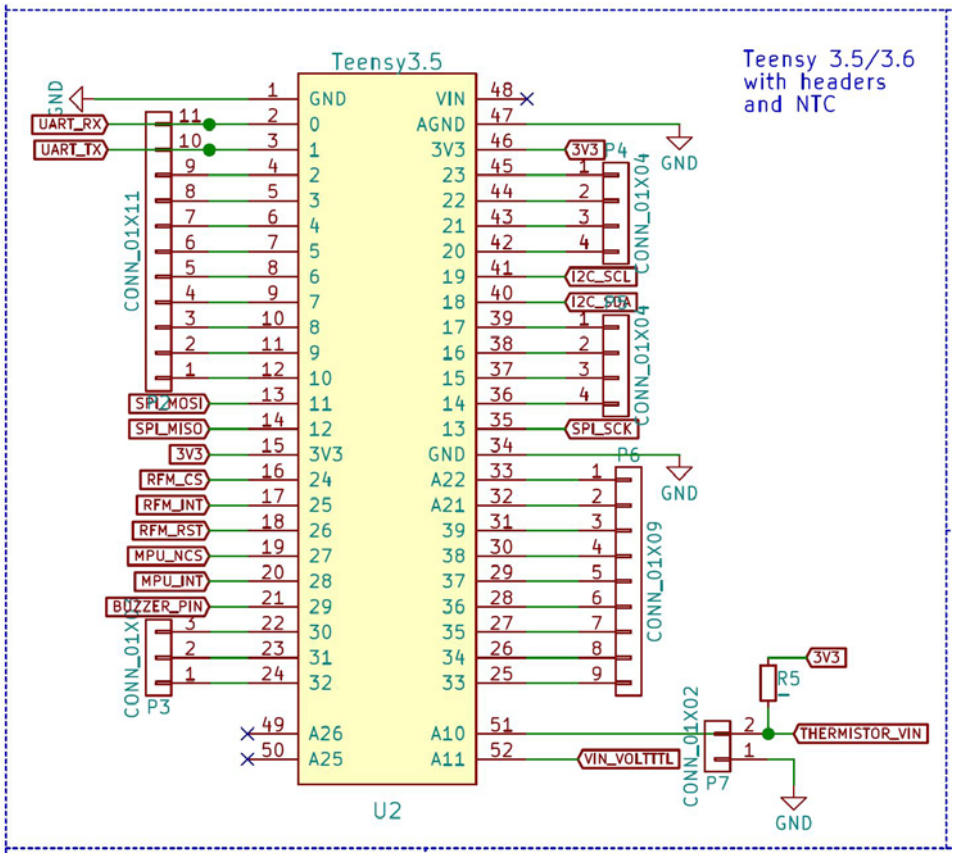
Vedlegg A Kretsskjema

A.1 Kretsskjema CanSat-kortet med Teensy 3.5

Figuren under viser kretsskjemaet for CanSat-kortet med Teensy 3.5

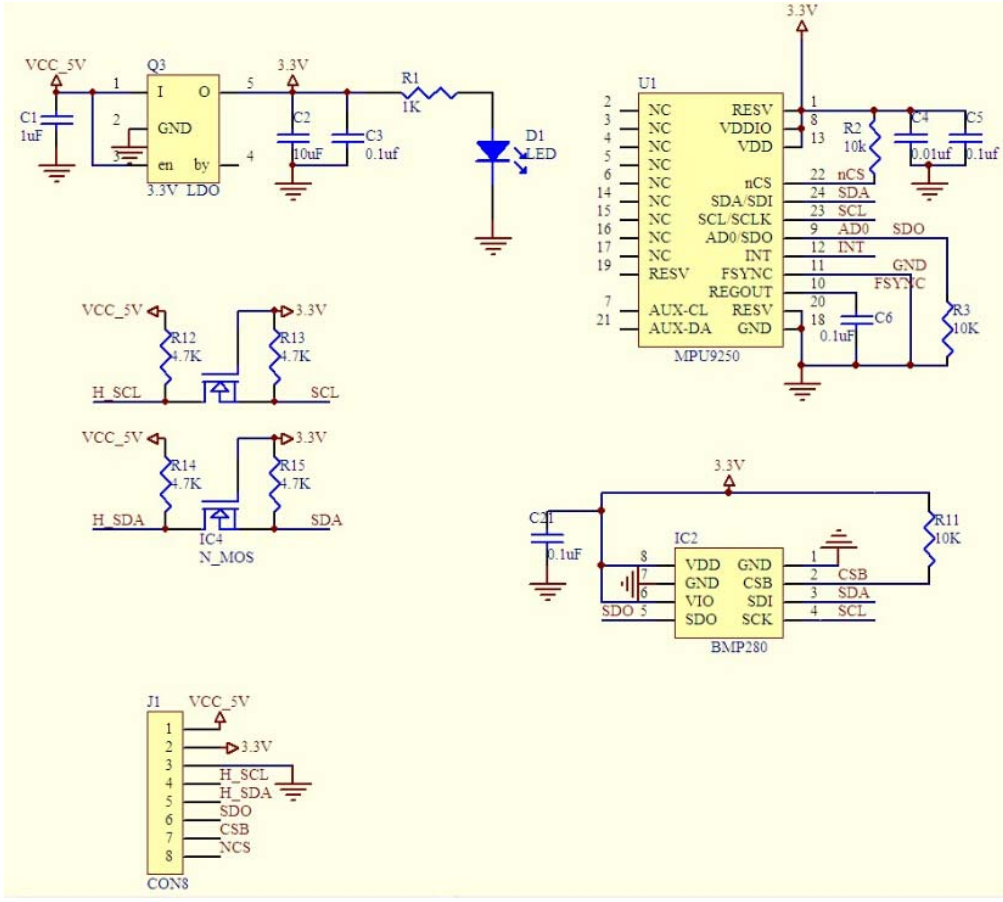



GY-91
GY-91
IMU module
VIN
3V3
GND
SCL
SDA
SDO/SA0
NCS
CSB
U5
1
2
3
4
5
6
7
8
3V3
GND
R8
R9
I2C_SCL
I2C_SDA


Teensy3.5
Teensy 3.5/3.6
with headers
and NTC
GND
UART_RX
UART_TX
CONN_01X11
SPI_MOSI
SPI_MISO
3V3
RFM_CS
RFM_INT
RFM_RST
MPU_NCS
MPU_INT
BUZZER_PIN
CONN_01X
P3
GND
VIN
AGND
3V3
GND
A22
A21
A26
A25
A10
A11
U2
P4
CONN_01X04
I2C_SCL
I2C_SDA
SPI_SCK
P6
CONN_01X09
VIN_VOLTTTL
R5
THERMISTOR_VIN
CONN_01X02
P7

A.2 Kretsskjema GY-91

Figuren viser kretsskjema for kortet GY-91



Vedlegg B Eksempler på kode

Disse testprogrammene er skrevet av Christoffer Stausland ved NAROM, og er ev. kommentert eller lettere modifisert av Nils Kr. Rossing.

B.1 Testprogram GY91, NTC, batteri og buzzer

```
#include <math.h>
#include <GY91.h>

#define USE_BUZZER            1
#define OUTPUT_TEMP           1
#define OUTPUT_VIN            1
#define OUTPUT_ACC            0
#define OUTPUT_GYRO           0
#define OUTPUT_MAG            0
#define OUTPUT_PRESSURE       1

#define BUZZER_PIN            29
#define BUZZER_TEMP           21.0

#define _A1   3.354016E-3
#define _B1   2.569850E-4
#define _C1   2.620131E-6
#define _D1   6.383091E-8

unsigned long _time=0;
double ax, ay, az, gx, gy, gz, mx, my, mz, pressure;
GY91 gy91;

void setup() {
  Serial.begin(9600);
  while(!Serial);

  analogReadResolution(12);
  analogWriteResolution(16);
```

```
  digitalWriteFast(BUZZER_PIN, LOW);
  pinMode(BUZZER_PIN, OUTPUT);

  if (!gy91.init()) {
    Serial.println("Could not initiate GY91");
    while(1);
  }
}

void loop() {
#if USE_BUZZER
  if (read_temp_direct() > BUZZER_TEMP)
    //analogWrite(BUZZER_PIN, (read_temp_direct()-BUZZER_TEMP)*1000);
    tone(BUZZER_PIN,440);
  else
    noTone(BUZZER_PIN);
#endif

  if (millis()-_time > 100) { // Makes sure that we do not send data too
often
    _time = millis();

    gy91.read_acc();
    gy91.read_gyro();
    gy91.read_mag();

    pressure = gy91.readPressure();

    ax = gy91.ax;
    ay = gy91.ay;
    az = gy91.az;

    gx = gy91.gx;
    gy = gy91.gy;
    gz = gy91.gz;
```

```
    mx = gy91.mx;
    my = gy91.my;
    mz = gy91.mz;

#if OUTPUT_VIN == 1
    Serial.print(analogRead(A11)*3.3*(3.6+6.2)/(4095*3.6));
    Serial.print("\t");
#endif

#if OUTPUT_TEMP == 1
    Serial.print(read_temp_direct());
    Serial.print("\t");
#endif

#if OUTPUT_ACC == 1
    Serial.print(ax);
    Serial.print("\t");
    Serial.print(ay);
    Serial.print("\t");
    Serial.print(az);
    Serial.print("\t");
#endif

#if OUTPUT_GYRO == 1
    Serial.print(gx);
    Serial.print("\t");
    Serial.print(gy);
    Serial.print("\t");
    Serial.print(gz);
    Serial.print("\t");
#endif

#if OUTPUT_MAG == 1
    Serial.print(mx);
    Serial.print("\t");
    Serial.print(my);
```

```
    Serial.print("\t");
    Serial.print(mz);
    Serial.print("\t");
#endif

#if OUTPUT_PRESSURE == 1
    Serial.print(pressure/1000,4); // it is in Pascals, but we want it
in centi bar to make
                                   // it easier to plot, which is one tenth
of hPa/mbar
#endif

    Serial.println();
  }
}

double read_NTC(double V_NTC) {
  double R_NTC, log_NTC;
  R_NTC = V_NTC*4700/(3.3-V_NTC);
  log_NTC = log(R_NTC/10000);
  return 1/(_A1 + _B1*log_NTC + _C1*log_NTC*log_NTC +
_D1*log_NTC*log_NTC*log_NTC)-273.15;
}

double read_temp_direct() {
  double R_NTC, log_NTC;
  uint16_t ARead = analogRead(A10);
  R_NTC = 4700*ARead/(4095.0-ARead);
  log_NTC = log(R_NTC/10000);
  return 1/(_A1 + _B1*log_NTC + _C1*log_NTC*log_NTC +
_D1*log_NTC*log_NTC*log_NTC)-273.15;
}
```

B.2 Program for senderen for testing av radiokommunikasjon

```
#include <Cansat_RFM96.h>
#define USE_SD 0
```

```
Cansat_RFM96 rfm96(433500, USE_SD);
unsigned long time_counter=0, time_to_send=0, loop_counter=0,
  counter=0;

void setup() {
  Serial.begin(9600);
  //while(!Serial); // Wait for the user to open Serial Monitor

  Serial.println("Starting setup");

  if (!rfm96.init()) {
    Serial.println("Init of radio failed, stopping");
    while(1);
  }

  Serial.println("Found RFM96 radio, and it is working as expected");

  rfm96.setTxPower(5); // +5 dBm, approx 3 mW, which is quite low

  // This is the first we will send.
  //rfm96.printlnToBuffer("Ready to send...");

  // This function will write the buffer to the file
  // if we have started SD storage and then send the
  // data if TX is ready. If SD storage is disabled,
  // it will just send it if TX is ready
  rfm96.sendAndWriteToFile();

  Serial.println("End of setup");
  Serial.println();
}

void loop() {
  if (rfm96.isTxReady() && !time_to_send) {
    time_to_send = millis()-time_counter;
  }
```

```
if (millis()-loop_counter > 500) {
  loop_counter = millis();

  // We choose something simple to send: first the number of
  // milliseconds that has elapsed, then a comma, then a one-
  // incrementing counter, then a comma and then the time it
  // took to send the last buffer in milliseconds
  rfm96.printToBuffer(millis());
  rfm96.printToBuffer(", ");
  rfm96.printToBuffer(counter++);
  rfm96.printToBuffer(", ");
  rfm96.printToBuffer(time_to_send);
  rfm96.printlnToBuffer(", And then we have a very long sentence");

  // Again, no need to check if Tx is ready, since this
  // function will check if Tx is ready, and if not it will
  // only write to file (if SD is enabled). If neither is
  // ready/enabled, it will do nothing
  rfm96.sendAndWriteToFile();
  time_counter = millis();

  // To debug, we also write approx the same (millis are
  // different)
  Serial.print(millis());
  Serial.print(", ");
  Serial.print(counter);
  Serial.print(", ");
  Serial.print(time_to_send);
  Serial.println(", And then we have a very long sentence");

  // We set it to 0 to let the if in the top of loop
  // know that we have started a new transmission. This
  // to avoid it trigger only one time.
  time_to_send = 0;
}
```

```
}
```

B.3 Program for mottakeren for testing av radiokommunikasjon

Dette programmet lastes over i det forenklede mottakerkortet som plugges i PC'en.

```
#include <Cansat_RFM96.h>
#define USE_SD 0

Cansat_RFM96 rfm96(433500, USE_SD);
unsigned long time_counter=0;

void setup() {
  Serial.begin(9600);
  while(!Serial);

  Serial.println("Starting setup");

  if (!rfm96.init()) {
    Serial.println("Init of radio failed, stopping");
    while(1);
  }

  Serial.println("Found RFM96 radio, and it is working as expected");

  rfm96.setTxPower(5); // +5 dBm, approx 3 mW, which is quite low

  Serial.println("End of setup");
  Serial.println();
}

void loop() {
  uint8_t read_value;

  // We check if there it something in the buffer
  while (rfm96.available()) {
    // We keep track of the time since we last received something,
    // so the user can get feedback if the radio does not receive
```

```
    // something
    time_counter = millis();

    // Read it into a variable. Here we could just directly use:
    // Serial.write(rfm96.read());
    read_value = rfm96.read();

    // Write it to file. We do not use Serial.print, because the
    // conversion to readable ASCII has already been done when
    // we send it
    Serial.write(read_value);
  }

  if (millis()-time_counter > 5000) {
    time_counter = millis();
    Serial.println("We have not received anything in 5 seconds");
  }
}
```

B.4 Testprogram for lesing fra analog inngang

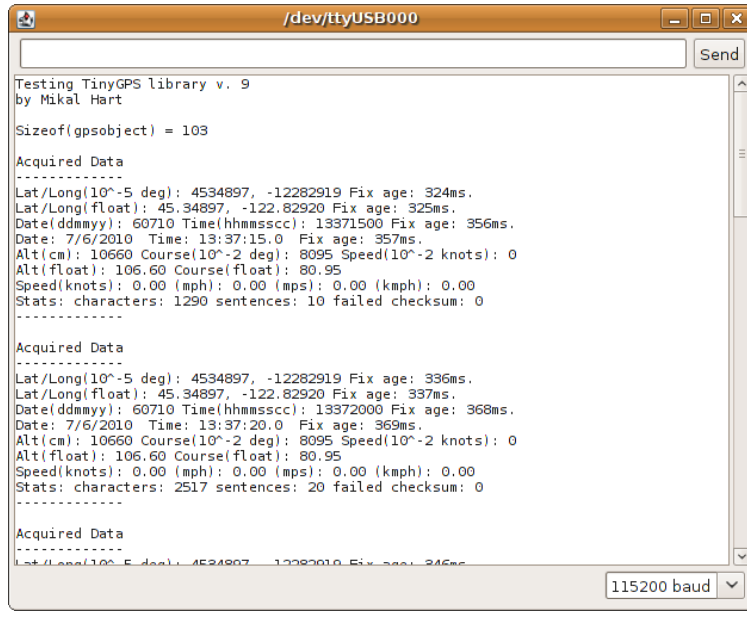
```
/*
 * Teensy 3.5 test code.
 * This code is intended as a very first introduction to reading analog
and digital values.
 *
 * The code reads in two digital inputs and two analog inputs and display
this on the Serial Monitor once every second.
 */


void setup() {
  Serial.begin(9600); // Baud rate doesn't matter, so just use 9600
  while(!Serial);

  pinMode(3, INPUT_PULLUP);
  analogReadResolution(12); // 12 bit, meaning maxium value is 4095 (2^12-
1)
```

```
}

void loop() {
  Serial.print("Time: ");
  Serial.print(millis());
  Serial.print(" ms\t\tDigital value pin 2: ");
  Serial.print(digitalReadFast(2));
  Serial.print("\t\tDigital value pin 3: ");
  Serial.print(digitalReadFast(3));
  Serial.print("\t\tAnalog value pin A0: ");
  Serial.print(analogRead(A0)*3.3/4095);
  Serial.print(" volts\t\tAnalog value pin A1: ");
  Serial.print(analogRead(A1)*3.3/4095);
  Serial.print(" volts");
  Serial.println();

  delay(1000);
}
```

B.5 Komplett testprogram for mottak av GPS data fra NEO-6M

Programmet er hentet fra Mikal Hart's beskrivelse av hvordan GPS-modulen NEO-6M kan kobles opp mot Teensy 3.0 (3.5). For detaljer se: https://www.pjrc.com/teensy/td_libs_TinyGPS.html

Dataene fra programmet skrives ut på formen vist i figuren til høyre.



Vi har brukt koden direkte for kommunikasjon mellom Teensy 3.5 og NEO-6M med en endring:

```
Uart.begin(4800);
```

er endret til:

```
Uart.begin(9600);
```

Kode:

```
#include <TinyGPS.h>

/* This sample code demonstrates the normal use of a TinyGPS object. */
TinyGPS gps;

/* On Teensy, the UART (real serial port) is always best to use. */
/* Unlike Arduino, there's no need to use NewSoftSerial because */
/* the "Serial" object uses the USB port, leaving the UART free. */
HardwareSerial Uart = HardwareSerial();

void gpsdump(TinyGPS &gps);
void printFloat(double f, int digits = 2);

void setup()
{
  Serial.begin(115200);
  Uart.begin(4800);

  delay(1000);
  Serial.print("Testing TinyGPS library v. ");
Serial.println(TinyGPS::library_version());
  Serial.println("by Mikal Hart");
  Serial.println();
  Serial.print("Sizeof(gpsobject) = "); Serial.println(sizeof(TinyGPS));
  Serial.println();
}

void loop()
{
  bool newdata = false;
  unsigned long start = millis();

  // Every 5 seconds we print an update
  while (millis() - start < 5000) {
    if (Uart.available()) {
      char c = Uart.read();
      // Serial.print(c);  // uncomment to see raw GPS data
      if (gps.encode(c)) {
```

```
        newdata = true;
        // break;  // uncomment to print new data immediately!
      }
    }
  }

  if (newdata) {
    Serial.println("Acquired Data");
    Serial.println("-------------");
    gpsdump(gps);
    Serial.println("-------------");
    Serial.println();
  }
}

void gpsdump(TinyGPS &gps)
{
  long lat, lon;
  float flat, flon;
  unsigned long age, date, time, chars;
  int year;
  byte month, day, hour, minute, second, hundredths;
  unsigned short sentences, failed;

  gps.get_position(&lat, &lon, &age);
  Serial.print("Lat/Long(10^-5 deg): "); Serial.print(lat); Serial.print(", ");
Serial.print(lon);
  Serial.print(" Fix age: "); Serial.print(age); Serial.println("ms.");

  // On Arduino, GPS characters may be lost during lengthy Serial.print()
  // On Teensy, Serial prints to USB, which has large output buffering and
  //   runs very fast, so it's not necessary to worry about missing 4800
  //   baud GPS characters.

  gps.f_get_position(&flat, &flon, &age);
  Serial.print("Lat/Long(float): "); printFloat(flat, 5); Serial.print(", ");
printFloat(flon, 5);
  Serial.print(" Fix age: "); Serial.print(age); Serial.println("ms.");

  gps.get_datetime(&date, &time, &age);
  Serial.print("Date(ddmmyy): "); Serial.print(date); Serial.print(" Time(hhmms-
scc): ");
```

```
    Serial.print(time);
  Serial.print(" Fix age: "); Serial.print(age); Serial.println("ms.");


  gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths,
&age);
  Serial.print("Date: "); Serial.print(static_cast<int>(month)); Serial.print("/
");
    Serial.print(static_cast<int>(day)); Serial.print("/"); Serial.print(year);
  Serial.print("  Time: "); Serial.print(static_cast<int>(hour));
Serial.print(":");
    Serial.print(static_cast<int>(minute)); Serial.print(":");
Serial.print(static_cast<int>(second));
    Serial.print("."); Serial.print(static_cast<int>(hundredths));
  Serial.print("  Fix age: ");  Serial.print(age); Serial.println("ms.");


  Serial.print("Alt(cm): "); Serial.print(gps.altitude()); Serial.print("
Course(10^-2 deg): ");
    Serial.print(gps.course()); Serial.print(" Speed(10^-2 knots): ");
Serial.println(gps.speed());
  Serial.print("Alt(float): "); printFloat(gps.f_altitude()); Serial.print("
Course(float): ");
    printFloat(gps.f_course()); Serial.println();
 Serial.print("Speed(knots): "); printFloat(gps.f_speed_knots()); Serial.print("
(mph): ");
    printFloat(gps.f_speed_mph());
 Serial.print(" (mps): "); printFloat(gps.f_speed_mps()); Serial.print(" (kmph):
");
    printFloat(gps.f_speed_kmph()); Serial.println();


  gps.stats(&chars, &sentences, &failed);
  Serial.print("Stats: characters: "); Serial.print(chars); Serial.print(" sen-
tences: ");
    Serial.print(sentences); Serial.print(" failed checksum: ");
Serial.println(failed);
}

void printFloat(double number, int digits)
{
  // Handle negative numbers
  if (number < 0.0) {
     Serial.print('-');
     number = -number;
  }
```

```
  // Round correctly so that print(1.999, 2) prints as "2.00"
  double rounding = 0.5;
  for (uint8_t i=0; i<digits; ++i)
    rounding /= 10.0;

  number += rounding;

  // Extract the integer part of the number and print it
  unsigned long int_part = (unsigned long)number;
  double remainder = number - (double)int_part;
  Serial.print(int_part);

  // Print the decimal point, but only if there are digits beyond
  if (digits > 0)
    Serial.print(".");

  // Extract digits from the remainder one at a time
  while (digits-- > 0) {
    remainder *= 10.0;
    int toPrint = int(remainder);
    Serial.print(toPrint);
    remainder -= toPrint;
  }
}
```

B.6 Program for styring av terminering (BallongSquid.ino)

Skrevet av Thomas Gansmoe, NAROM)

```
int pwmInn = 3;
int ledPin = 13;
int outPin = 4;
int outTestPin = 7;
int pulsewidth;
long int timer1, blinkTime=0;
boolean trigger = LOW, flag1=LOW;
void setup()
{
 Serial.begin(9600);       // Starts serial port
 pinMode(ledPin, OUTPUT);
 pinMode(pwmInn, INPUT);    // sets the pin as input
 pinMode(outPin, OUTPUT);
```

```
  pinMode(outTestPin, OUTPUT);
  digitalWrite(outPin,LOW);
  digitalWrite(outTestPin,LOW);
}
void loop()
{
  trigger=LOW;
  flag1=LOW;
  pulsewidth=pulseIn(pwmInn,HIGH);
  while (pulsewidth>1700&&pulsewidth<2000){
    if (flag1==LOW){
      flag1=HIGH;
      timer1=millis();
    }
    if (millis()-timer1>5000){
      trigger=HIGH;
      digitalWrite(outPin,trigger);
    }
    Serial.print(pulsewidth);
    Serial.print("  - Trigger: ");
    Serial.print(trigger);
    Serial.print("  - Flag: ");
    Serial.print(flag1);
    Serial.print("  - Timer: ");
    Serial.print(millis()-timer1);
    Serial.println(" --InLoop");
    if ((millis()-blinkTime)>100){
      if (digitalRead(ledPin)==LOW){digitalWrite(ledPin,HIGH);}
      else {digitalWrite(ledPin,LOW);}
      blinkTime=millis();
    }
    if (trigger==HIGH) digitalWrite(ledPin,LOW);
    pulsewidth=pulseIn(pwmInn,HIGH);
  }
  Serial.print(pulsewidth);
  Serial.print("  - Trigger: ");
  Serial.print(trigger);
  Serial.print("  - Flag: ");
```

```
  Serial.print(flag1);
  Serial.print("  - Timer: ");
  Serial.print(millis()-blinkTime);
  Serial.print("  -  ");
  Serial.print(digitalRead(7));
  Serial.print("  -  ");
  Serial.println(millis());

  if ((millis()-blinkTime)>500){
    if (digitalRead(ledPin)==LOW){digitalWrite(ledPin,HIGH);}
    else {digitalWrite(ledPin,LOW);}
    blinkTime=millis();
  }
digitalWrite(outPin,trigger);
  if (millis()>60000) digitalWrite(outTestPin,HIGH);
}
```

Vedlegg C Regler for forankrede ballonger

Kapittel IX. Forankrede ballonger

§ 9-1. Kapittelets virkeområde

Dette kapittel gjelder forankrede ballonger når de på grunn av størrelse, brennbarhet, høyde over bakken mv. kan medføre fare eller ulempe for luftfarten.

§ 9-2. Krav i forbindelse med oppsending av forankrede ballonger

1. Forankrede ballonger må bare sendes opp, og holdes oppe, når bakkesikten er minst 8 km. De tillates heller ikke holdt oppe når de er nærmere skyer enn 300 m vertikalt og/eller 1.500 m horisontalt.
2. Oppsending av forankrede ballonger i kontrollert luftrom krever tillatelse fra Luftfartstilsynet. Søknad om tillatelse må være Luftfartstilsynet i hende minst 14 dager før oppsendingen skal finne sted.
3. Utenfor kontrollert luftrom gjelder bestemmelsen i annet ledd tilsvarende når oppsendingen skjer nærmere en flyplass enn 10 km eller skjer innenfor et av Luftforsvarets lavflygingsområder.
4. Hvis avstanden fra en flyplass er 10 km eller mer, kan oppsendingen til høyder over 45 m bare finne sted såfremt melding er gitt minst 14 dager forut til nærmeste enhet av lufttrafikktjenesten eller til NOTAM-kontoret.
5. Melding etter fjerde ledd skal inneholde:

   a) navn, adresse og eventuelt telefonnummer til den som har ansvaret for oppsendingen,

   b) sted/posisjon for forankringen,

   c) ballongens maksimale høyde,

   d) dato, tid og varighet for oppsendingen,

   e) opplysninger om hvordan ballongen med forankringsutstyr er varselmerket.
6. Forankrede ballonger skal være utstyrt med en anordning for punktering av ballongen. Anordningen må kunne utløses automatisk dersom ballongen kommer i drift. Hvis eier eller bruker av en forankret ballong blir kjent med at ballongen er kommet i drift uten at punktering har skjedd, plikter han straks å underrette nærmeste enhet av lufttrafikktjenesten eller NOTAM-kontoret.

Vedlegg D Leverandører

D.1 Oversikt over komponenter til CanSat kit'et

Vedlegg E Læreplaner

E.1 Teknologi og Forskningslære 1

Fargekodene forsøker å antyde hva som ***lett kan*** knyttes til CanSat og romteknologi (rødt/kursiv), hva som er litt på siden (blått) og det som vanskelig lar seg oppfylle innen prosjektet (sort).

Den unge ingeniøren - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne:

- *planlegge og bygge en konstruksjon som er fast eller bevegelig, og som har en definert funksjon*
- bruke tredimensjonale tegninger eller skisser i utvikling av konstruksjoner
- bruke forskjellige materialer og former for sammenføyninger og begrunne valg av materialer og byggemåte ut fra materialenes egenskaper og konstruksjonens funksjon
- *bruke sensorer og styringssystemer i forbindelse med forsøk og konstruksjoner*
- *dokumentere og vurdere konstruksjoners fysiske egenskaper og funksjonalitet ved hjelp av målinger og enkle beregninger*

Den unge forskeren - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne:

- *gjøre rede for hvordan et naturvitenskapelig prosjekt planlegges, gjennomføres og etterarbeides før det blir publisert (planlegge målinger utført i løpet av droppet)*
- *planlegge, gjennomføre, analysere og dokumentere systematiske målinger*
- om støy, *luftforurensning*, inneklima og vannkvalitet, og drøfte virkninger på helse og miljø

Teknologi, naturvitenskap og samfunn - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- drøfte etiske, miljømessige, kulturelle og politiske sider ved teknologisk utvikling
- *beskrive den historiske utviklingen av en teknologisk innretning, forklare virkemåten og drøfte anvendelser i samfunnet*
- gjøre rede for utvikling og produksjon av et teknologisk produkt og vurdere produktets brukervennlighet, utviklingsmuligheter og miljøpåvirkning
- *beskrive prinsipper og virkemåte for noen moderne instrumenter i industri, helsevesen eller forskning, og gjøre rede for nytten og eventuelle skadevirkninger*
- kartlegge og presentere praktisk bruk av realfag i en lokal bedrift eller institusjon

Design og produktutvikling - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- *gjøre rede for funksjonen til vanlige komponenter i elektroniske kretser, og gjenkjenne komponentene i en krets*
- *lage elektroniske kretser ved å lodde komponenter og simulere og teste kretsene*
- *forme og utvikle produkter som har en definert funksjon og inneholder elektronikk*
- *dokumentere og presentere designprosesser fra idé til ferdig produkt*
- *begrunne valg av materialer i produkter og vurdere produktenes form og funksjon, miljømessige konsekvenser*, estetikk og forbedringsmuligheter
- *utføre målinger med eller teste et eget produkt, og vurdere kvaliteten på produktet med tanke på funksjonalitet*

E.2 Teknologi og forskningslære 2

Den unge forskeren - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- gjøre rede for et forskningsprosjekt i en bedrift eller institusjon, og beskrive problemstillinger, organisering, måleutstyr, resultater og finansiering
- *planlegge og gjennomføre naturvitenskapelige undersøkelser basert på egne ideer, og presentere arbeidet i en vitenskapelig form*
- *drøfte resultater fra egne undersøkelser i forhold til relevant kunnskap på området, og vurdere hvordan kontroll av variabler og reproduserbarhet er ivaretatt*

Naturvitenskapelige arbeidsmetoder - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- *forklare hva som menes med modell, teori og hypotese, og gjøre rede for hvordan de brukes og utvikles i forskning*
- drøfte ved å bruke eksempler hvordan empiriske data kan styrke eller forkaste en hypotese
- *gjøre rede for hvordan forskning utvikles og kvalitetssikres gjennom samarbeid, kritisk vurdering og argumentasjon*
- *gjøre rede for strukturen i en vitenskapelig publikasjon eller presentasjon*

Forskning, teknologi og samfunn - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- beskrive kjennetegn ved grunnforskning, anvendt forskning og utviklingsarbeid og gjøre rede for hovedtrekk ved finansiering og styring
- gjøre rede for betydningen av naturvitenskapelig forskning og teknologiutvikling for næringsliv og samfunn

- drøfte økonomiske, miljømessige og etiske spørsmål i forbindelse med naturvitenskapelig forskning og teknologiutvikling
- *drøfte og gi eksempler på hvordan forskningsresultater og ny teknologi formidles og brukes av forskningsinstitusjoner, medier, bedrifter, interessegrupper og myndigheter*

Vitenskapsfilosofi og vitenskapsteori - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- beskrive hovedtrekk i den historiske utviklingen av vitenskapelige tenkemåter og drøfte teknologiens rolle i denne utviklingen
- gjøre rede for hovedideene til noen sentrale vitenskapsteoretikere og vitenskapsfilosofer vurdere hvordan argumentasjon i aktuelle naturvitenskapelige debatter bygger på empiriske resultater, teoretisk kunnskap og ideologisk ståsted

E.3 Teknologi i Praksis (TiP) – ungdomsskolen

Formål

Valfaga skal medverke til at elevane, kvar for seg og i fellesskap, styrker lysta til å lære og opplever meistring gjennom praktisk og variert arbeid. Valfaga er tverrfaglege og skal medverke til heilskap og samanheng i opplæringa.

Teknologi handlar om den menneskeskapte verda og om innretningar og system som kan gjere kvardagen betre. Opp gjennom tidene har menneska brukt kreativitet og skaparevner til å utvikle reiskapar, maskinar og andre teknologiske produkt og løysingar. Teknologien grip inn på mange område, og har gjeve og kan gje både moglegheiter og utfordringar, både for den einskilde og for samfunnet. Innanfor teknologien finn vi dei enklaste verktøy og produkt og den mest avanserte elektronikken. Erfaring med og innsikt i teknologi kan fremje personleg utvikling, demokratisk deltaking og medverke til eit aktivt forhold til ein teknologisk kvardag.

Valfaget teknologi i praksis skal motivere elevane til å utvikle teknologiske produkt med utgangspunkt i lokale behov og problemstillingar. Prosessen frå idé til eit ferdig produkt kan medverke til skaparglede og meistringsoppleving. Gjennom eige arbeid og i samarbeid med andre kan elevane utvikle ferdigheiter og innsikt. Det inneber å prøve ut eigne talent og moglegheiter på ulike steg i prosessen, vurdere prosessar og produkt og få tilbakemeldingar frå andre.

Valfaget handlar om å planleggje, konstruere og framstille gjenstandar og produkt med varierte materiale og teknologiske løysingar. Kunnskap om teknologiske produkt som blir brukte i dagleglivet, gjev eit godt grunnlag for å forbetre produkt og utvikle nye produkt.

Valfaget hentar hovudelement frå matematikk, naturfag og kunst og handverk/duodji. Element frå norsk/samisk, RLE og samfunnsfag kan også inngå.

Hovedområder

Undersøkingar

Hovedområdet handlar om korleis teknologiske produkt er konstruerte og verkar, kva for prosessar som inngår i utvikling og bruk, og kva for behov produkta dekkjer. Utvikling, konstruksjon og produksjon av teknologi inngår i hovudområdet, i tillegg til helse, miljø og sikkerheit (HMS). Kunnskap om korleis teknologien byggjer på nokre grunnleggjande prinsipp, og korleis ny teknologi byggjer på tidlegare erfaringar, høyrer også med til hovudområdet.

Idéutvikling og produksjon

Hovudområdet omfattar planlegging, framstilling og utprøving av eigne produkt og konstruksjonar. Planar for framstilling og utprøving av eigne produkt og konstruksjonar byggjer på kravspesifikasjon.

I utviklingsfasen er kjennskap til design og verkemåte til andre produkt viktig. Diskusjon omkring ulike sider ved produkta er viktig i alle fasar av produktutviklinga og kan også medverke til å forbetre prosessar og produkt.

Kompetansemål

Undersøkingar

- undersøkje teknologiske produkt og dei vala som er gjorde med omsyn til bruk, tekniske løysingar, funksjonalitet og design
- *demonstrere riktig bruk av utvalde verktøy*
- vurdere teknologiske produkt ut frå brukartilpassing, HMS-krav og miljøtilpassing

Idéutvikling og produksjon

- *utvikle ein realistisk kravspesifikasjon for eit teknologisk produkt og beskrive kva behov produktet skal dekkje*
- *framstille produktet med eigna materiale, komponentar, og funksjonelle teknologiske løysingar*
- *bruke kunnskap om andre produkt i arbeidet med eige produkt*
- *teste eigne produkt og foreslå moglege forbetringar*

E.4 Forskning i Praksis (FiP) – ungdomsskolen

Formål

Valgfagene skal bidra til at elevene, hver for seg og i fellesskap, styrker lysten til å lære og opplever mestring gjennom praktisk og variert arbeid. Valgfagene er tverrfaglige og skal bidra til helhet og sammenheng i opplæringen.

Forskning handler om å utvikle ny kunnskap og innsikt. Formuleringer som ”forskning viser at” eller ”vitenskapelige undersøkelser har vist at” brukes ofte og viser at tilliten til forskning er stor. Forskning skal også være kritisk, prøve ut om forskningsresultater er riktige og bidra til debatt. Erfaring med utforskning kan derfor danne grunnlag for egne meninger og kritisk tenkning, og gi elevene bedre mulighet til å forholde seg til debatt om forskning på en hensiktsmessig måte.

Opplæringen i valgfaget forskning i praksis skal bidra til at elevene får erfaring med vitenskapelige metoder og arbeidsmåter. Valgfaget skal stimulere til undring, aktiv handling for å teste ut løsningerog utvikle evnen til å stille nye spørsmål. Opplæringen skal bidra til å finne forklaringer på det som er observert, og gjennom kildegransking, eksperiment og observasjon kontrollere om forklaringene holder.

I valgfaget forskning i praksis blir elevene utfordret til å undersøke aktuelle spørsmål relatert til natur, miljø og klima, samt kultur-, samfunns- og arbeidsliv. Dette innebærer at de skal finne problemstillinger de ønsker å undersøke, vurdere mulige forklaringer, planlegge og gjennomføre undersøkelser, bruke utstyr og teknikker for datainnsamling, bearbeide data, og vurdere og formidle resultatene. Slik bidrar valgfaget med erfaringer og ferdigheter om de praktiske sidene ved forskning. Valgfaget stimulerer til nysgjerrighet og bruk av fantasi for å finne forklaringer. Det bidrar også til innsikt i hvordan etablert kunnskap og andres forskning kan trekkes inn i egen utforskning. Opplæringen skal legge til rette for at elevene får erfare hvordan det å gi og få tilbakemeldinger underveis i en utforskende prosess fremmer kvaliteten på resultatene.

Valgfaget henter hovedelementer fra naturfag, matematikk og samfunnsfag, men tema kan også hentes fra ungdomstrinnets øvrige fag.

Hovedområder

Idéutvikling

Hovedområdet omfatter de kreative sidene som inngår i alle stadier av forskningsprosessen. Utgangspunktet er en problemstilling som man lurer på og vil finne svar på. Det omfatter videre å lage gode forskningsspørsmål, få innspill fra andre, reformulere i lys av innspillene og, planlegge undersøkelser. I dette hovedområdet utvikler elevene mulige hypoteser, henter inn andre relevante data og sammenlikner og diskuterer ideer med andre elever.

Praktisk utforskning

Hovedområdet omfatter arbeid med ulike prosesser ved praktisk gjennomføring av de planlagte undersøkelsene. Det er viktig at elevene prøver ut og bruker forskjellige metoder, gjør seg kjent med og tar i bruk ulike typer utstyr og samler inn og systematiserer data. I tillegg skal elevene prøve ut tolkninger og sammenlikne egne funn med andres samt formidle resultatet.

Kompetansemål

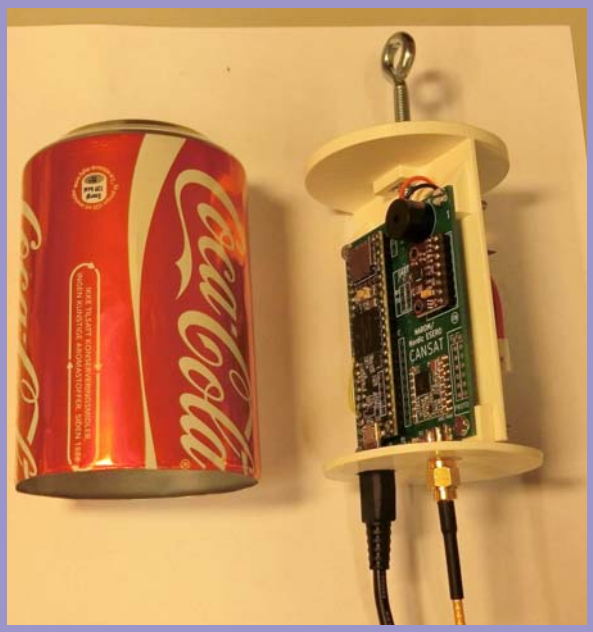
Idéutvikling

- *finne problemstillinger, formulere forskbare spørsmål og forslag til hypoteser*
- *planlegge undersøkelser basert på egne eller gruppas forskningsspørsmål og hypoteser*
- *delta i samtaler om egen og andres utforskning*
- *argumentere for egne hypoteser i lys av funn og andre relevante undersøkelser*

Praktisk utforskning

- *gjennomføre planlagte undersøkelser og foreta relevante justeringer underveis*
- *bruke relevante metoder og utstyr for innsamling og analyse av data*

- *demonstrere og forklare metoder, virkemåten til utstyr og prosedyrer for datainnsamling i gjennomførte forskningsprosjekter*
- *systematisere data slik at mønstre kommer tydelig fram og vurdere usikkerheter*
- *formidle resultater fra egne prosjekter*



Hensikten med heftet er å samle erfaringer fra utprøving av det nye CanSat-kittet basert på Teensy 3.5. Kittet er utviklet ved NAROM av Christoffer Stausland og Bente Jensen. Deretter er det gjennomgått og bearbeidet av Nils Kr. Rossing som har brukt heftet til å samle og systematisere data. Det er ikke meningen at heftet skal konkurrere med den offisielle CanSat-håndboken som ligger på nett, men ev. være et supplement.

Foreløpig beskriver heftet de sensorene som er inkludert i standardutgaven av CanSat. Deretter kan man etter eget ønske bygge ut data- og sensordelen etter eget behov og interesser. Heftet kan ev. brukes som tilleggsstoff til Teknologi og Forskningslære 1 og Fysikk 1 i videregående skole, men også av lærere som underviser de nye valgfagene i ungdomsskolen – Teknologi i Praksis og Forskning i Praksis.

Nils Kr. Rossing
Dosent ved Skolelaboratoriet,
Institutt for fysikk, NTNU
og prosjektleder ved Vitensenteret
E-post: nils.rossing@ntnu.no

Christoffer Stausland
NAROM
E-post: christoffer@narom.no

Bente Jensen
NAROM
E-post: bente@narom.no

Trykk: NTNU-trykk

NTNU
Trondheim

Institutt for fysikk

Skolelaboratoriet
for matematikk, naturfag og teknologi

Tlf. 73 55 11 43
http://www.ntnu.no/skolelab