Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for elektroniske systemer

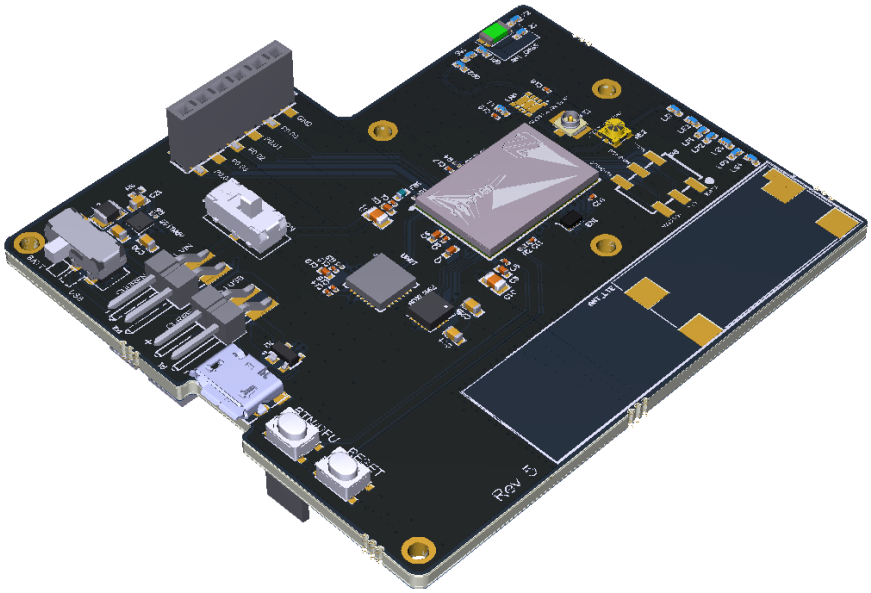
Eirik Ekrheim Bjørklund
Torje Farbu Johansen
Jonas Marelius Saari
Håvard Kalliainen
Andreas Ipsen

E2223 Low power GPS tracker

IELET2900 Bacheloroppgave elektronikk og sensorsystemer

Bacheloroppgave i BIELEKTRO
Veileder: Rolf Kristian Snilsberg, Richard McCrae, Eirik Hovde Skanke

Mai 2022



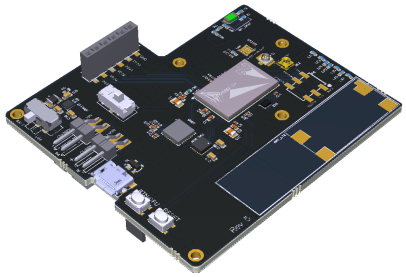
NTNU
Norwegian University of
Science and Technology

NORDIC SEMICONDUCTOR

Eirik Ekrheim Bjørklund
Torje Farbu Johansen
Jonas Marelius Saari
Håvard Kalliainen
Andreas Ipsen

E2223 Low power GPS tracker

IELET2900 Bacheloroppgave elektronikk og sensorsystemer



Bacheloroppgave i BIELEKTRO
Veileder: Rolf Kristian Snilsberg, Richard McCrae, Eirik Hovde Skanke
Mai 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for elektroniske systemer

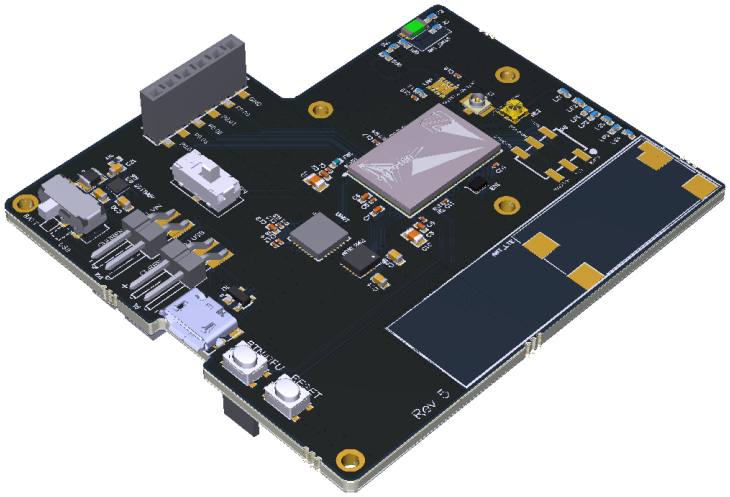
NTNU
Kunnskap for en bedre verden

NTNU

Kunnskap for en bedre verden

E2223 Low power GPS tracker

IELET2900 Bacheloroppgave elektronikk og sensorsystemer



Bachelor thesis 2022

Eirik Ekrheim Bjørklund
Torje Farbu Johansen
Jonas Marelius Saari
Håvard Kalliainen
Andreas Ipsen

Totalt antall sider inkludert forsiden: 87
Trondheim, 20/05/2022

NTNU
PROSJEKTOPPGAVE

NTNU

Tittel:
E2223 Low power GPS tracker

Eirik Ekrheim Bjørklund
Torje Farbu Johansen
Jonas Marelius Saari
Håvard Kalliainen
Andreas Ipsen

Dato:	Emnekode:	Emne:	Dokument tilgang:
20/05/2022	IELET2900	Bacheloroppgave elektronikk og sensorsystemer	Åpent

Studium:	Ant. sider/vedlegg:	Bibl. nr:
BIELEKTRO	87 / 4	

Veileder:
Richard McCrae
Eirik Hovde Skanke
Rolf Kristian Snilsberg

Keywords / Nøkkelord:
nRF9160, GPS, GPS Tracker, GNNS, Nordic Semiconductor, Low Power, Low Power GPS Tracker, Sheep Aqcuisition Unit, Asset Tracker, Zephyr, Zephyr RTOS, sanntidsprogrammering, Zephyr

Denne oppgaven er en besvarelse utført av studenter ved NTNU i Trondheim.

Innholdsfortegnelse

ABSTRACT

Based on Nordic Semiconductors nRF9160 SiP the group developet a PCB outlay with the capability of handling GPS communication, full duplex cloudcomminication with nRF Cloud where postional data and other data is transfered. It was estimated that when the system was running the lowest transmission rate the battery life could sustain the system for 821 days using a 10000mAh battery. The intended use is for tracking sheep on pasture. In this report, we account for the process regarding production of both hardware and software, and resuts of current measurements done to find optimal solutions for our intention. The theory part deals with relevant GPS/LTE theory, presents key concepts for thread programming, how to do battery life analysis and antenna tuning. The method section goes into depth about design philosophy and how to realize GPS module on PCB. It explains how software is implemented on nRF9160SiP. The results section includes measurement results of current draws from different modes such as Cell Tower location, GNSS and own code. Finished PCBs and software are also presented as results. Towards the end, the hardware, software and noise analysis against the finished product are discussed. The conclusion concludes with a finished product that is a prototype of GPS tracker with similar working code.

SAMMENDRAG

Basert på Nordic Semiconductors nRF9160 SiP ble det utviklet et utlegg som er i stand til å håndtere GPS kommunikasjon, full duplex skykommunikasjon med nRF Cloud der posisjon og annen data overføres. Med laveste senderate kunne vi estimere en batterilevetid på opp til 821 dager med et 10000mAh batteri. Den tenkte anvendelsen for produktet er å spore sau på beitemarkene. I denne rapporten redegjør vi for prosessen rundt produksjon av både hardware og software, samt resultat av strømmålinger som er gjennomført for å finne optimale løsninger for vår anvendelse. Teoridelen tar for seg relevant GPS/LTE teori, legger fram nøkkelbegreper for trådproggramering, hvordan gjøre batterilevetidsanalyse og antennetuning. Metodedelen går i dybden om designfilosofi og hvordan realisere GPS modul på PCB. Den forklarer hvordan software er implementert på nRF9160SiP. Resultatdelen inkuderer måleresultater av strømtrekk fra forskjellige moduser som Celletårn plassering, GNSS og egen kode. Ferdig PCB og software blir også presenterit som resultater. Mot slutten drøftes hardware, software og støманalysen mot det ferdige produktet. Konklusjonen konkluderer med et ferdig produkt som er en prototype av GPS-tracker med tilsvarende fungerende kode.

1 INNLEDNING

Denne bacheloroppgaven som ble utlyst av Nordic Semiconductor ASA omhandler utviklingen av en Low Power GPS Tracker basert på deres produkt nRF9160 SiP. Den tenkte anvendelsen for produktet vårt er valgfri, men vi skal gjennomføre nøye analyser av strømforbruk og presisjon. Strømforbruket på en GPS-anvendelse vil være avhengig blant annet av presisjonen og intervallene det sendes data. Mikrokontrolleren støtter ulike modus for presisjon, og vi ønsker en anvendelse hvor det er naturlig å teste de ulike for å studere strømforbruket. Vi har derfor valgt å lage et produkt for å spore sau. Da vil det være naturlig å både benytte både lav- og høy presisjon.

Oppgaven er todelt:

Software: Det skal produseres optimalisert kode for anvendelsen med spesielt hensyn på lav energibruk, og gjennomføre strømmålinger tilknyttet dette.

Hardware: Det skal produseres støttekretser og antennemodul for å realisere et faktisk produkt.

Forutsetningene skal gi grunnlag for enkelte designvalg gjør vi del strømanalyser av ulike teknikker for posisjonering og kommunikasjon. I prosjektet benyttes Nordics Power Profiling Kit II (PPK2) for å gjøre strømtrekksanalyse. Med denne ble det analysert strømtrekk ved ulike posisjoneringsmetoder, samt analyse av vårt eget system.

2 Definisjoner

AT Commands
Et kommandosett som brukes til å kommunisere med modemet, også kjent som Hayes Commands.

A-GPS
Assistert GPS. Enheten får almanakk og ephemeris via internett, som korter ned tiden for å få en GPS fix

Cold Start
Om enhet mangler eller sitter på foreldet informasjon må den iverksette en innhentingprosess. Dette kan være tilfellet om enhet er ny eller har vært avslått over lengre perioder.

CPU-tid
Tiden tildelt til en tråd før den en annen prosess må prioriteres.

Fix
GPS fix inneholder informasjon om enhetens posisjon.

FOTA
Står for Firmware Over the Air og er en teknologi som gjør det mulig å oppgradere fastvare trådløst.

Geofence
Et virtuelt gjerde realisert ved bruk av GPS eller telekommunikasjons-teknologi.

Hot Start
Enhet sitter på korrekt informasjon og kan starte beregning av posisjon. Under denne tilstanden har enhet typisk vært i standby modus. Tiden det tar for å hente posisjonsdata etter en hot start kalles ofte for TTSF som står for Time To Subsequent Fix.

ISR
Interrupt service routine. En rutine som kjører vet interrupt.

JSON
JavaScript Object Notation er en måte formatere datasett for å enkelt utveksle data mellom systemer.

Kernel
En kernel kobler applikasjonsprogramvaren til maskinvaren i en datamaskin.

MCELL
Høyere nøyaktighet enn SCELL ved å se på posisjon til flere mobiltårn. Sparer strøm på å ikke bruke GPS modemet.

MCUBoot
MCUboot er en sikker oppstartslaster for 32-bit mikrokontrollere.

MEMS
MEMS (Mikroelektromekansikesystemer) er en teknologi som dreier seg elektriske og mekaniske komponenter på mikroskala.

nRF9160
Kompakt System in Package (SiP) med lavt effektforbruk. Inkluderer modem som støtter LTE og GNSS.

nRF9160 GPS moduser:

nPM1100
Integrert strømstyringsystem. Består av en lineær-modus batterilader til opplading av litium-ion/litium-polymer batterier, inngangspenning regulator som tar 4,1 til 6,7 V og en ”step-down” spenningsomformer som har utgangspenning 1.8 V, 2.1, 2.7 og 3 V.

NVM
Minne som kan holde lagret informasjon gjennom et brudd med strømstilførsel

P-GPS
Liknende A-GPS der forskjellen er at det er større tidsrom for innhenting av data. Assistansedata er predikert over to uker. P-GPS er et mer strømbesparende alternativ siden innhentingsfrekvens er mindre.

Rammeverk
En abstraksjon av software med det formål å oppnå avansert funksjonalitet ved mindre brukerskrevet kode.

SCELL
Enhet bruker ikke GPS modem, men kan si noe om posisjonen sin ved å vise til nærmeste mobiltårn. Siden GPS ikke er i bruk er strømforbruk lav.

TTFF
Time To First Fix. Hvor lang tid det tar enheten å anskaffe satellittdata for å beregne nøyaktig posisjon.

VS Code
Skrivebordsprogram for å skrive kildekode. Kan kjøres på Mac, Windows og Linux. Kommer med mulighet til å laste ned utvidelser slik som nrf Connect SDK.

WIFI
Enhet skanner minst to WIFI tilkoblingspunkt og sender informasjonen videre til NRF cloud som videre beregner posisjon. Sparer strøm på å ikke bruke GPS modem.

Warm Start

Enhet sitter allerede på informasjon som kan brukes til posisjonsberegning. Dette kan være fart, tidligere posisjon, timing og tidligere almanakk data. Denne tilstanden er typisk når enhet har vært slått av i en kort periode.

3 TEORETISK GRUNNLAG

I dette kapittel vil vi redegjøre for nødvendig teori for å forstå vårt prosjekt.

3.1 Grunnleggende GPS-teori

3.1.1 GPS

GPS er et globalt posisjoneringssystem bestående av 31 satellitter som går i bane rundt jorda. Dersom en enhet på jorda har kontakt med minst fire av disse kan man med høy presisjon beregne både lokasjon, hastighet og tid. Dette skjer ved at satellittene sender ut sin lokasjon og et svært nøyaktig tidspunkt for sending. Det er deretter opp til mottakeren å beregne sin plassering basert på denne dataen. Satellittene sender ut deler av almanakken, som er grov plasseringen til alle satellittene, ved hver sending. Denne brukes til å finne de ulike satellittene, men er ikke presis nok til å beregne posisjonen fra. Hele almanakken er sendt etter 12.5 minutter. Satellittene sender også ut ephemeris. Dette er en svært presis plassering av hver enkelt satellitt, som er god nok til å beregne plasseringen utfra. Dersom bakke-enheten ikke har noe informasjon om hvor og når den er kan den være avhengig av hele almanakken, altså minst 12.5 minutter med mottak, og ephemeris for å kunne få en riktig plassering. [1]

3.1.2 A-GPS

Ved bruk av assistert GPS, eller A-GPS, får man en plassering av alle satellittene samt en grov plassering av eget utstyr via internettet. Dette gjør at man kan få beregnet lokasjonen sin langt raskere, og man blir i teorien kun avhengige av én sending fra minst fire satellitter for å plassere seg. Dette krever en tilbyder av A-GPS data for å fungere. Nordic tilbyr A-GPS data fra sin egen nRF Cloud. Et alternativ til dette kan være SUPL via f.eks. Google.
A-GPS krever aktiv internett tilkobling. [2]

3.1.3 P-GPS

Ved bruk av predikert GPS, eller P-GPS, kan man laste ned data som forutser plasseringen til satellittene i opp til to uker. Når denne dataen er lastet ned har man ikke lengre behov for internett tilkobling. P-GPS kan dermed gi oss raskere GPS fix uten en aktiv internett tilkobling. Dette krever også en tilbyder av tjenesten for å fungere. Også her tilbyr Nordic dataen via nRF Cloud. [3]

3.1.4 GPS på nRF9160

nRF9160 støtter alle de nevnte GPS-modusene. Den støtter også en kombinasjon av A-GPS og P-GPS, hvor man får raskere plassering, og bruker mindre datatrafikk.
I tillegg til dette støtter den to ulike operasjonsmodus. Disse er:

- Kontinuerlig - GPS henter posisjon hvert sekund
- Periodisk - GPS henter posisjon etter gitte intervaller [4]

3.2 Grunnleggende LTE-teori

3.2.1 LTE Moduser

Som en utvidelse av 4G-teknologien, er deler av LTE-nettverket i dag klargjort- og tilsidesatt for M2M-kommunikasjon (Machine-to-Machine). Dette er igjen inndelt i hovedsakelig to ulike kommunikasjonsprotokoller man har tilgjengelig. disse er:

- LTE-M - En høyhastighets, lav-effekt IoT-modus. Denne er raskere og mer stabil.
- NB-IoT - En lavhastighets, og ennå mer strømsparende modus. Den har høyere latency og lavere overføringshastigheter. [5]

3.2.2 LTE strømsparing

Det finnes to energi besparingsmoduser støttet på LTE: PSM og eDRX. Begge disse moduser er støttet uavhengig av om man bruker LTE-M eller NB-IoT.
LTE bruker 1,28 sekunders vindu for kontakt med mobilnettet. Disse vinduene kalles paging cycles og hviletiden mellom hver syklus kalles for hyperframes.

- eDRX - Står for Extended Discontinuous Reception. Enheten hviler for 40+ hyperframes eller 10.24 sekunder mellom hver paging cycle.
- PSM - Står for Power Saving Mode. Enheten sover så lenge som den er programmert til. Dette tidsrommet kan være alt fra timer, dager til uker. [6]

nRF9160 støtter alle kombinasjoner av de nevnte modusene.

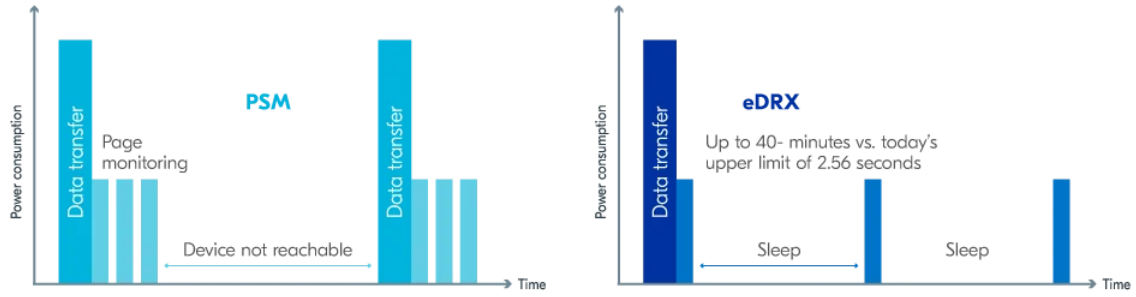


Figure 1: PSM Vs. eDRX [7]

3.3 Antenne, transmisjonslinje og tuning

3.3.1 Grunnleggende Antenneteori:

For å velge riktig antenne etter krav må man forstå noe av den grunnleggende antenneteorien, som vi skal redegjøre for i dette delkapittelet.

Utstrålingsdiagram:
En antennes utstrålingsdiagram er en matematisk eller grafisk funksjon av dens utstrålingsegenskaper i rommet [8]. I datablad finner man disse diagrammene i både 2D og 3D. I 2D er feltstyrken gitt i en polar graf i tre plan: XY, XZ eller YZ. I 3D blir intensiteten i en gitt retning illustrert ved hjelp av en fargeskala som går fra rødt til lilla. I både 2D og 3D diagrammene blir ofte en faktor Azimuth eller Az og Elevation eller EL angitt. Azimuth er radiasjonsmønsteret sett horisontalt (ser ned mot antennen) og Elevation vertikalt (ser på siden av antennen).

Isotropiske, direksjonelle og omnidireksjonelle strålemønster:
En isotropisk antenne har et uniformt strålemønster som betyr at den har lik utstråling i alle retninger. Dette er en ideell form av antenne som ikke er realiserbar. En antenne som viser ulik forsterkning i visse retninger i sitt strålemønster kalles en direksjonell antenne. Omnidireksjonell er en type antenne som har sterkere utstråling i en bestemt retning[8].

Antenneeffektivitet:
En antennes effektivitet består av flere faktorer. Tap i antennen kan oppstå på grunn av refleksjoner i transmisjonslinjen og I^2R tap i dielektrikum eller strømleder.

Hvor:
e_0 = total effektivitet

$$e_0 = e_r e_c e_d \tag{1}$$

1: En antennes totale effektivitet

e_r = refleksjonseffektivitet på grunn av impedans feiltilpasning = $1 - |\Gamma|^2$
e_c = effektiviteten til strømleder
e_d = effektiviteten til dielektrisk materiale
Γ = Refleksjonskoeffisient til antennens inngangsspenning. $\Gamma = \frac{(Z_{in} - Z_0)}{Z_{in} + Z_0}$ Der Z_{in} er inngangsimpedansen til antennen og Z_0 er den karakteristiske impedansen til transmisjonslinjen [8].

Impedansmatching

Impedansmatching er å utligne impedansdifferansen mellom en last og en transmisjonslinje slik at de hele nettverket har samme motstand i alle ledd og reaktansen ideelt sett blir 0Ω. Dette gjøres for å maksimere effektoverføringen og for å minske refleksjonene. Reaktansen blir 0Ω når lasten og transmisjonslinjen er komplekskonjugert.

$$Z_{last} = Z^*{}_{kilde} \tag{2}$$

2: Formel for maksimal effektoverføring mellom kilde og last

VSWR & Returtap
VSWR (Voltage Standing Wave Ratio) er et mål på hvor godt lasten (antennen) er matchet med transmisjonslinjen. En VSWR verdi på 1 vil si at all effekt blir levert til lasten. Desto høyere VSWR desto mer av effekten vil bli reflektert og mindre levert til last[9].
Return tap er det samme som VSWR, men uttrykt i en annen skala. Forskjellen er at return tap er målt i desibel og går fra 0 dB (100 % reflektert effekt) til 20 dB (1 % reflektert effekt). VSWR er enhetsløs og går tilsvarende fra ∞ (100 % reflektert effekt) til 1.2 (1 % reflektert effekt)[10].

Forsterkning:
En antennes forsterkning er definert ut fra gitt utstråling intensitet sammenlignet med en teoretisk ideell derfor tapsløs isotropisk antenne (uniform utstråling). Desto mer direksjonell antennen er desto større forskjell i forsterkning ser man i en antennens strålediagram. Direksjonell forsterkning er ønskelig i noen applikasjoner og uønsket i andre. Hvis man til enhver tid vet hvor signalet kommer fra og enhet skal holdes stasjonært er høy direksjonell forsterkning (topp forsterkning) ønskelig[8].

Smith diagram

Smith diagram er en måte å grafisk framstille impedansverdier på. De brukes ofte som et verktøy for å justere impedansen i en krets. Den rette linjen som deler diagrammet i midten horisontalt representerer det resistive komponentet av impedansen, mens ringene som går gjennom linjen indikerer det reaktive komponenten. I øvre halvdel er reaktanser induktiv mens i nedre halvdel er den kapasitiv.

VNA

VNA (Nettverksanalysator) er et måleinstrument som typisk blir brukt til å måle refleksjon og forsterkning i RF-kretser. Den sender et sweep-sinus signal innen ønsket frekvensområde, måler det returnerte signalet og gir informasjon om kretsens RF egenskaper. Disse blir ofte framstilt på Smith-diagram i tillegg til et diagram som viser VSWR og impedansen i kretsen per frekvens.

3.3.2 Transmisjonslinje:

En transmisjonslinje er nødvendig når lengden av linjen nærmer seg eller er større en bølgelinjen av signalet. Når lengden på signal linjen er større en $\frac{1}{10}$ av signalets bølgelengde bør man begynne å tenke på transmisjonslinje teori [9].

$$Z_0 = \sqrt{\frac{L}{C}} \tag{3}$$

3: Impedans i en tapsfri transmisjonslinje

Refleksjoner:
Ulike konsekvenser gjeldende ulike Z_L:

$$\Gamma_0 = \frac{V_-}{V_+} = \frac{Z_L - Z_0}{Z_L + Z_0} \tag{4}$$

4: Refleksjonskoeffisient

En åpen transmisjonslinje Z_0 gir en refleksjonskoeffisient lik 1 Dette betyr at signalet totalreflekteres med samme polaritet.

$$Z_L = \infty = \Gamma_0 = 1$$

En kortsluttet transmisjonslinje gir en refleksjonskoeffisient lik -1, som betyr at signalet reflekteres med motsatt polaritet.

$$Z_L = \infty = \Gamma_0 = -1$$

Det ideelle tilfellet der lastimpedans Z_L er helt lik transmisjonslinje impedans Z_0 gir en refleksjonskoeffisient lik 0.

Stående Bølgeforhold (SWR):
SWR forholdet brukes til å estimere feilmatching av Z_L.

$$SWR = \frac{V_{max}}{V_{min}} = \frac{I_{max}}{I_{min}} = \frac{1 + |\Gamma_0|}{1 - |\Gamma_0|} \tag{5}$$

5: Stående bølgeforhold SWR

SWR = 1 —— 100 % Matchet
SWR $\rightarrow \infty$ —— 0 % Matchet

Coplanar transmisjonslinje:
En coplanar transmisjonslinje har flere fordeler over microstrip og stripline. En coplanar transmisjonslinje har mindre tap enn striplines fordi den er lagt på toppen og ikke mellom dielektrikum. Videre har den fordelen over microstrip ved at jordlag bidrar til ekstra utstråling isolasjon. Den sistnevnte egenskapen kan videre styrkes ved å plassere skjold viaer nær linjen for å forsikre nær tilkobling til jordlaget[11].

Differensiell transmisjonslinje: Når man plasserer to transmisjonslinjer nær hverandre får man ekstra impedans som må regnes med. Fremgangsmåten er avhengig av om signalene som skal påføres linjene er i fellesmodus (Z_{even}) eller differensialmodus (Z_{odd}). Siden en transmisjonslinje har en induktans L_o får det differensielle paret en felles L_m mellom seg. Det oppstår også ekstra kapasitans fordi lederne sammen med luft og prepreg former en kondensator[12].

Impedansmatching:
For å oppnå maksimal effektoverføring fra antenne til mottakeren, kreves det samme impedans over hele linje. Dette besørger man ved å stemme lasten.
Impedansen er svært vanskelig å beregne på forhånd, så i de fleste anvendelser printer man mønsterkortet og monterer på komponenter først, for så å måle hva impedansen er; og justere dette. en vanlig fremgangsmåte for dette er å gjør klar monteringshull for komponenter; en spole i serie og to avkoblingskondensatorer til jord på hver sin side. Når man vet hvor på det komplekse planet impedansen er, kan vi også beregne komponentverdier for å tune dette til å matche impedansen i systemet forøvrig.

Via stitching og shielding
PCB med flere lag har nesten alltid større kobberområder koblet til jord eller forsyningsnettet.For å forsikre god tilkobling mellom lag er via stitching brukt.Ved å bruke teknikken via stitching sikrer man kortere strømsløyfer og lavere impedans derfor mindre tap og støy. Via shielding blir brukt rundt radiofrekvens linjer for å ”gjerde” inne elektromagnetiske bølger som radierer fra linja. Reduserer derfor crosstalk og EMI[**?**].

3.4 Strømtrekkanalyse

Batterilevetid
Et batteri vil ha en viss mengde ladning. Denne ladningen er angitt som [Ah] og er et mål på batteriets kapasitet til å levere strøm. Når denne kapasiteten er brukt opp er batteriet tomt. Eksempelvis batteriet som brukes til i SAU en kapasitet på 235 mAh. Med konstant strømtrekk på 40mA, forventer man at batteriet skal vare i

$$235mAh/40mA = 5.875timer$$

Mål av strømtrekk
Måling av strømtrekk er et mål av energien som overføres fra kilden. Forbrukt energi blir målt i watt [W], men det er veldig vanlig å måle strømtrekk i ampere[A], eller milliampere [mA] for laveffekt elektriske kretser. Det er to viktige faktorer ved mål av strømtrekk: ”Topp” strømforbruk og energiforbruket.

Strømtopper er viktige i småelektronikk fordi små batterier er som regel svak til topper. Litium-ion mynt batterier er populære i småelektronikk fordi de har høy kapasitet per vekt. Men de er sensitive for strømtopper. Der de billigste ikke tåler større topper enn 5mA uten å bli skadet. [13] En annen grunn til at toppene gjøre at batteriet holder kortere er at når batteriet begynner å bli tomt så øker den indre resistansen i batteriet. Da kan høye strømmer gjøre at spenningen blir for lav til å drive kretsen. Konsekvensen av det er at modulen ikke funker lengre.

Energiforbruket målt i Joule er hva som bestemmer hvor mye energi som tappes fra batteriet under last. For å beregne energiforbruk gjennom strømforbruk må vi ta hensyn til varigheten av utførelsen. Forbruket målt i watt er da gjennomsnittlig strømforbruk Pavg over en gitt tid (1). Men formel (1) funker bare hvis strømforbruket er konstant. Derfor, før man kan beregne det totale energiforbruket, må man kombinere strømdata med respektive tidsintervaller. Dermed er energiforbruket gitt integralet av strømforbruket ut tiden som trengs for å utføre operasjonen (2). Men hvis målefrekvensen er konstant vil den integrerte strømmen over tid (mAh) bli det samme som gjennomsnittet ganget med tiden (3). Dermed får vi følgende relasjon og forenkling [14] [15]

$$(1)\ \ W = P_{avg} \cdot \Delta t \quad \Rightarrow \quad (2)\ \ [Ah] = \int_{t_0}^{t_n} i(t)\,dt \quad \Rightarrow \quad (3)\ \ [Ah] = I_{avg} \cdot t$$

3.4.1 Power Profiler Kit II

Produktbeskrivelse
Power Profiler Kit II er en frittstående enhet utviklet av Nordic Semiconductor for å enkelt måle strømtrekk fra dev-kit eller egenutviklede kretser. Kittet kan kan settes i Ampere mode eller Source mode der sistnevnte modus leverer strøm til device under test (DUT). Strømprofilereren kan gjøre nøyaktige strømforbruk målinger i forbruksområdet som vanligvis sees i innebygde applikasjoner med lav effekt fra µAs til 1A. En slik treffsikkerhet er nødvendig ettersom nRF9160 er en Low Power SiP. Målinger hentes ut fra en ekstern PC-programvare som heter i nRF Connect for Desktop.

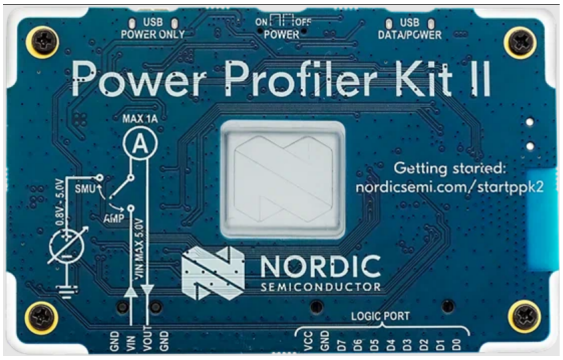


Figure 2: Power Profiler Kit II [16]

Produktspesifikasjoner

- 200nA til 1A strøm måleområde med en oppløsning som varierer mellom 100nA og 1mA
- Source mode og Amperemeter modus
- Source mode inkluderer innebygd programmerbar regulator med en 0,8V til 5V forsyningsområde og opp til 1A strømforsyning. (Ved 1A kreves to USB koblinger til PPK2)
- 100 ksps samplingsrate
- Frittstående enhet
- 8 digitale innganger for low-end logikk analysering
- Øyeblikkelig og gjennomsnittlige strømmålinger på alle nordics Dev-Kit, i tillegg til egenutviklede kretser

- Støttes gjennom nRF Connect for Desktops Power Profiler-app
- Eksporter måledata for etterbehandling

3.5 PMIC

Produktets spenningsforsyning, eller Power Management IC (PMIC), er et viktig komponent for laveffektselektronikk. Valg av PMIC og forsyningsmodus vil ha mye å si for strømforbruket. I dette delkapittelet vil vi redegjøre for noen viktige begreper rundt PMIC og valg av dette.

3.5.1 PWM & PFM omforming

PWM operasjon er en vanlig og effektiv metode for å regulere spenning. Ved bruk av en PWM kontroller, oscillator og et LC filter produseres et firkantsignal med vilkårlig pulsbredde som fungerer som bryter sammen med en eller flere interne mosfets. På filterutgangen likerettes signalet.
PWM operasjon fungerer bra ved høyere laster, men får minket effektivitet under lavere laster. Dette er på grunn av flere årsaker. Dynamiske tap på grunn av energi brukt til å opplade og utlade indre bryter mosfets gate (gate og kanal former en kondensator). Energien tapt øker med bryter frekvens. En annen kilde til tap i mosfet er på grunn av motstand i drain - source kanalen.

Passive komponenter i buck converters ytre krets kan også være en stor kilde til tap. Spolen bidrar til tap på grunn av motstand i kobbertråden og magnetiske tap i kjernen. I kondensatorer er ESR hovedkilden til tap som er betinget av type kondensator, størrelse, frekvens og laststrøm.
Løsningen er å kombinere puls bredde modulasjon (PWM) og puls frekvens modulasjon (PFM) for å få god effektivitet over et bredt lastområde. PFM har en fast pulsbredde, men varierer med frekvensen. Dette er ulikt PWM som har konstant frekvens, men varierer pulsbredden.

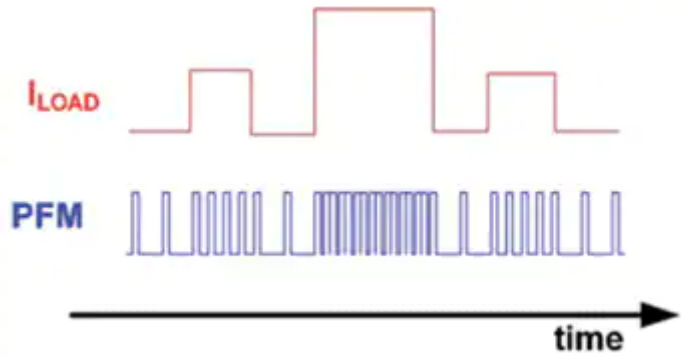


Figure 3: Høyere frekvens gir høyere laststrøm ved bruk av PFM operasjon (Steven Keeping - Digikey)

Som illustrert i figur 3 gir en høyere utgangsfrekvens høyere laststrøm. Det finnes flere typer PFM konverter arkitekturer. Hysteresisk spenningskonverter er et eksempel på en slik arkitektur. Ved bruk av spenning sensing blir en mosfet slått av og på. Ofte er en slik konverter kalt rippel regulator på grunn av mosfetvekslingen skaper en rippel på utgangen. En ulempe med PFM operasjon er en økning i utgangsrippel. Dette er på grunn av at man trenger et toleranse bånd for at kretsen kan merke når den bør slå på strømbryterene. Et kortere toleransebånd betyr lavere rippel, men større tap på grunn av hyppigere brytervekslingen.

Når en buck converter som veksler mellom PWM og PFM operasjon går fra lav last til høy vil man ofte kunne måle et kortvarig dypp i spenning på utgangen på grunn av konverter løkken tar tid å . Dette dyppet er uønsket og flere kommersielle lav effekt buck konvertere er det implementert mekanismer for å forminske dette dyppet.
Hybrid konvertere bestående av automatisk bytte mellom PWM og PFM blir ofte brukt i mobile enheter. Dette er på grunn av god effektivitet under lav last som drastisk øker batterilevetid. [17]

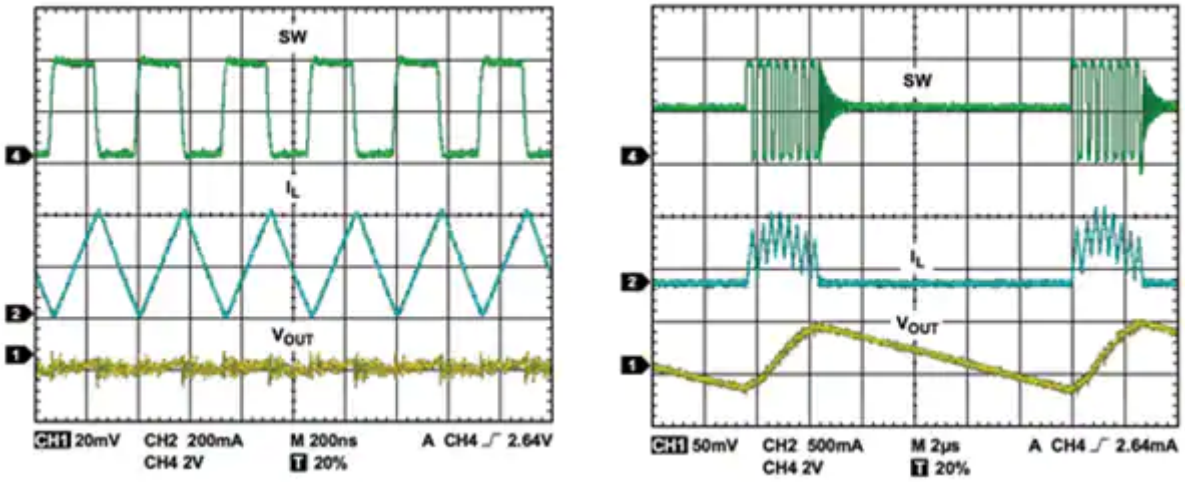


Figure 4: PWM Operasjon [venstre] og hysteresisk PFM operasjon [høyre] (Analog Devices)

3.6 Hardware

3.6.1 nRF9160

nRF9160 er en ”Low power SiP with integrated LTE-M/NB-IoT modem and GNSS” Det betyr at det er en programmerbar mikrokontroller med antenner for LTE og GPS, som er laget for å være spesielt strømeffektiv.
nRF9160 har både en programmerbar enhet, samt et modem som håndterer all kommunikasjon.

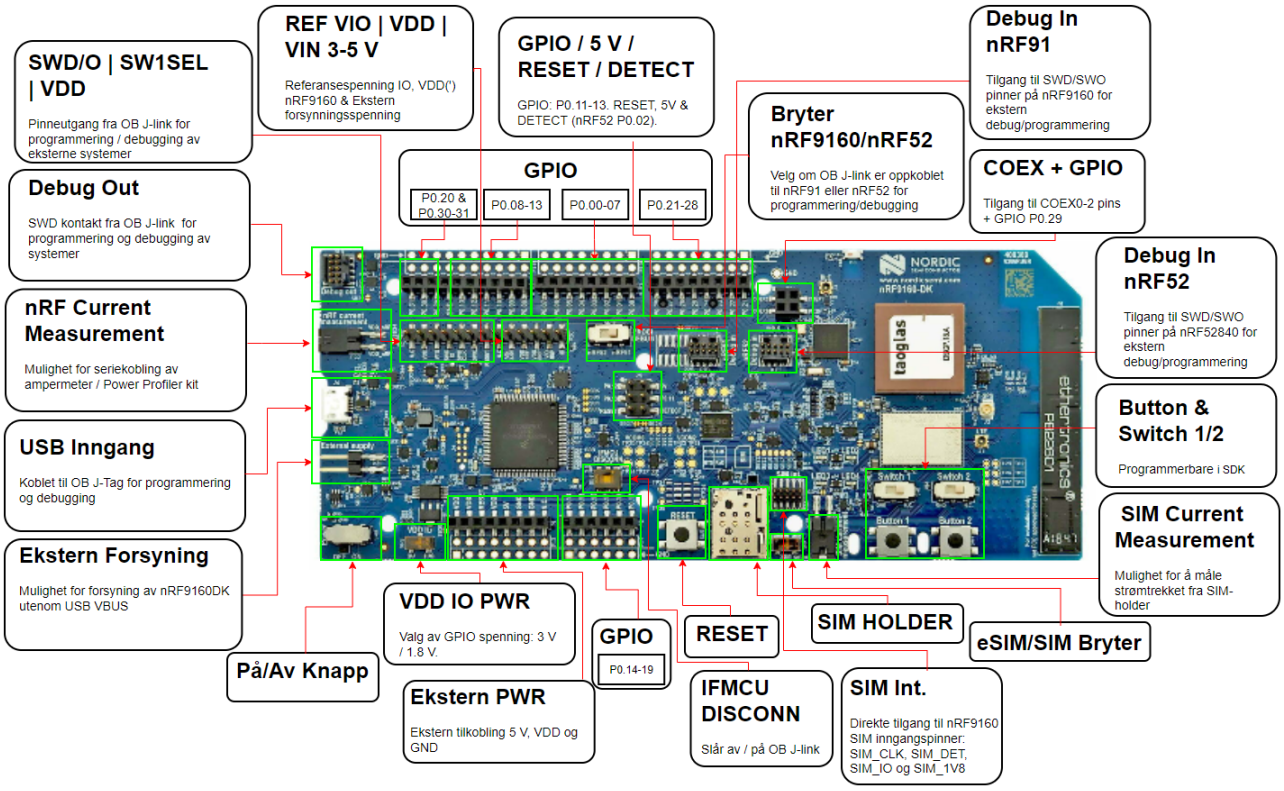


Figure 5: nRF9160DK utganger og funksjoner

3.6.2 Hardware-utvikling rundt nRF9160

Når man utvikler et produkt basert på nRF9160 vil man stå ovenfor en del designvalg og utfordringer, blant annet:

- Antenner - Kombinert eller frittstående, innebygd eller ekstern
- PMIC - Ekstern strømforsyning og dimensjonering av denne
- GPS LNA - Low Noise Amplifier for mottak av GPS
- Transmisjonslinje og tuning av denne
- Kablet kommunikasjonsgrensesnitt (UART etc)
- Grensesnitt for å flashe enhet
- Grensesnitt for strømmåling (spesielt i utvikler fasen)
- Øvrig GPIO

Vi skal i dette delkapittel redegjøre for nødvendig teori rundt de ulike punkter.

3.6.3 Antennevalg

Når man designer produktet sitt står man ovenfor noen valg tilknyttet antenner. Man kan ha eksterne antenner, eller man kan ha antenner på PCB. Man kan ha hver sine antenner for GPS og LTE, eller man kan kombinere det.
Antenner på PCB er plassbesparende, men har typisk lavere gain enn eksterne antenner.
Kombinerte eksterne antenner er plassbesparende, men har også typisk lavere gain enn å ha adskilte antenner.
Den beste løsningen vil derfor i alle tilfeller å ha eksterne, adskilte antenner. Men dette vil kreve et fysisk større produkt. [18]

3.6.4 PMIC

PMIC, Power Management IC, er en krets som regulerer driftsspenning og besørger et stabilt spenningsnivå til kretsen vår. nRF9160-chippen har en innebygd PMIC, men for å få større rekkevidde på tilgjengelige spenninger og for å sikre en mer stabil driftsstrøm kan man velge å bruke en ekstern PMIC istedenfor. Ulempen med dette er at den eksterne PMIC vil i alle tilfeller ha noe effekttap.

Teori rundt dette redegjøres for i delkapittel 3.5 PMIC .

3.6.5 GPS LNA

Siden GPS-signalet er svært svakt, under støygulvet, bør man bruke en Low-Noise Amplifier som filtrerer ut og forsterker GPS-signalet. Denne bør være plassert så nær GPS-antenna som mulig. DK bruker en SKY65943-11, som også er et båndfilter. [19]

3.6.6 Transmisjonslinje

En god transmisjonslinje er viktig både for signalintegritet og for å hindre unødig effektbruk.

Teori rundt dette redegjøres for i 3.3 Antenne, transmisjonslinje og tuning .

3.6.7 Kablet grensesnitt

Både for feilsøking og for å kunne laste programmer opp på enheten kan man tilgjengeliggjøre kommunikasjonsgrensesnitt via pinner eller f.eks USB. Det mest naturlige

kan være å tilgjengeliggjøre UART-kommunikasjon. Det mest praktiske her vil være å inkludere en USB-connector for å enkelt koble til en PC. For dette kreves også en USB til UART bro.

3.6.8 Grensesnitt for strømmåling

For å kunne måle strømtrekken på produktet kan man tilgjengeliggjøre målingspunkter nære strømkilden. Dette kan være to pinner som normalt kortsluttes med en jumper. Denne jumperen kan fjernes for å koble til et amperemeter eller f.eks. en Power Profiler Kit II.

3.6.9 Øvrige GPIO

Avhengig av anvendelsen kan man velge å tilgjengeliggjøre GPIO-pinner for å koble til eksterne sensorer og aktuatorer. Slik kan man åpne for fremtidige utvidelser av produktet sitt. Dette kan også inkludere I2C, SPI etc.

3.7 nRF Cloud

Nordic Semiconductor har sin egen, proprietære skyløsning kalt nRF Cloud. Vi er bedt om å bruke dette som vår front- og backend-løsning. Gjennom nRF Cloud tilbyr Nordic et grafisk brukergrensesnitt, som gir oversikt over brukerens utstyr, tilkoblingsstatus og evt plassering av disse, og som muliggjør kommunikasjon via et terminalvindu. nRF Cloud tilbyr også A-GPS- og P-GPS-data i mikrokontrollerne. Den støtter flere ulike kommunikasjonsprotokoller, bl.a MQTT og REST. Når produktet og SIM-kortet er registrert og provisjonert settes mqtt-topics ol opp automatisk, så det er veldig enkelt for brukeren å kommunisere med skyen. Nordic tilbyr også gode cloud-biblioteker med funksjoner som `cloud_send(cloud_backend, msg)`. Dette abstraherer vekk en stor del av jobben rundt skykommunikasjon. Dette biblioteket støtter også Amazon og Microsoft sine skyløsninger. [20]

3.8 nRF Connect for Desktop

For å hjelpe i utviklingen av applikasjoner til-, samt feilsøking, diagnostikk og generell bruk av Nordic Semiconductor sine SiP-er, så har det blitt samlet en mengde programmer i et verktøy ved navn **nRF Connect for Desktop**. Dette verktøyet er utviklerens inngang inn i Nordic sitt økosystem, og forenkler den ofte lange oppstartsfasen man

finner hos andre SiP-leverandører. I denne pakken finner vi blant annet programmer som *LTE Link Montior*, *Programmer* og *Toolchain Manager*.[21]

3.9 nRF Connect SDK

Selve programmeringen av Nordic Semiconductor sine SiP-er finner vi i deres **Software Development Kit** (SDK). En slik SDK kan også sees på som en type verktøykasse, med innhold bestående av koderammeverk, applikasjonseksempler og dokumentasjon. Det offisielle rammeverket som støttes av Nordic Semiconductor i nRF Connect SDK er Zephyr Project.[22]

3.9.1 Zephyr Project

Zephyr Project er et bredt økosystem som er utviklet i ett stort samarbeid, med åpen-kildekode-prinsippet i bunnen[23]. I kjernen av økosystemet ligger selve operativsystemet Zephyr OS. Dette operativsystemet har helt fra begynnelsen av blitt designet til å kjøre på svært strømeffektive og prosesseringsbegrensede mikrokontrollere. Rundt denne kjernen finner vi komplimenterende bibliotek og pakker, som fritt kan anvendes ved behov, og som alle samspiller godt med Zephyr OS. Mange av disse bibliotekene utvikles og vedlikeholdes av tredjeparter, og omhandler som regel en implementasjon av en etablert standard så som JSON, Wi-Fi, LoRaWAN eller Blutetooth LE, for å nevne noen.

3.9.2 Git

For å holde styr på alle tilleggene, endringene og utvidelsene fra bidragsgiverne i Zephyr Project, så blir det anerkjente verktøyet Git tatt i bruk. Git er et verktøy for versjonskontroll, som kort sagt muliggjør store samarbeid i dataverdenen. Prosjekter deles inn i såkalte *repositories*, og endringer spores og loggføres gjennom handlinger kallet *commits*[24]. Zephyr Project er organisert i et såkalt *monorepository*, som er en samling av flere enkelte repositories. Å dele inn et prosjekt i flere repositories kan i større prosjekt, som Zephyr Project, hjelpe med koordinasjon og innskrenking av ansvarsområder.

3.9.3 West

Siden Zephyr OS og omliggende biblioteker er i stadig utvikling, så kan det fort oppstå problemer med kompatibilitet. Løsningen på dette, er at hvert repository innad i Zephyr

Project anvender semantisk versjonering. Dette går ut på at man markerer en commit som en utgivelse (release), og gir den et versjonstall. En utgivelse vil aldri bli endret på, og man kan derfor være sikker på at alle med samme versjonstall sitter med samme kode. På denne måten, så kan man spesifikt velge hvilken versjon av et bibliotek man ønsker å bruke, og sikre seg at det ikke oppstår kompatibilitetsproblemer. Zephyr Project har da sitt eget kommandolinjeverktøy ved navn West, som automatisk henter inn kompatible versjoner av repositories direkte fra monorepositoriet.[25]

3.9.4 Semantisk Versjonering

Semantisk versjonering bygger på at hver utgivelse får tildelt et sammensatt tall. Dette tallet har tre komponenter; major-, minor- og patch-tallet. En endring i major-tallet betyr at det har blitt lagt til funksjonalitet, og at det oftere enn ikke har skjedd endringer som vil forårsake kompatibilitetsproblemer. Man må derfor gjennom en oppgraderingsprosess. En endring i minor-tallet kan ofte inneholde ny funksjonalitet, men da vil i så fall dette alltid være baklengskompatibelt. En endring i patch-tallet vil kun inneholde mindre endringer, såkalte ”patches/fixes”, og har derfor ingen innvirkning på kompatibilitet. For å velge ut en versjon man ønsker å jobbe med, kan man da si at man ønsker å jobbe med `^1.2.0`, der hatten betyr ”høyeste versjon uten en endring i major-tallet”, eller `~1.2.0` der tilde betyr ”høyeste versjon uten endring i minor-tallet”. Det finnes også tagger som `X.X.X-RC1`, som betyr *”Release Candidate 1”*, men disse er ikke viktige for dette avsnitts formål.[26]

3.9.5 CMAKE

Programmeringsspråket C er et gammelt språk, og ble utviklet lenge før det var aktuelt med frivillige utvidelsespakker og bibliotek. Av denne årsaken, så er det veldig tungvint å inkludere bibliotek eller flere filer uten ekstra verktøy, da det kan oppstå alle mulige problemer som dobbelt deklarasjon og feil inkluderingsrekkefølge, for å nevne noen. CMake er én av en håndfull løsninger på dette problemet, og blir brukt i mange større C-prosjekt. CMake fungerer ved at man inkluderer CMakeList-fil i hvert prosjekt eller modul, som beskriver hvilke filer som skal med og i hvilken rekkefølge de bør inkluderes. I tillegg så løser CMake problematikken rundt å kompilere C-programmer på tvers av blant annet Linux, Windows or MacOS. [27]

3.9.6 Zephyr OS

Zephyr OS er et relativt nytt RTOS (Real-Time Operating System), som er tilpasset for kjøring på svært ressursbegrenset maskinvare uten å ofre mye av funksjonalitet. Operativsystemet støtter en et bredt utvalg av mikrokontrollere, deriblant nRF9160.

3.9.7 Device Tree

Siden Zephyr kan kjøres på et så bredt utvalg av mikrokontrollere, så inkluderes det en teknologi kallet **Device Tree**[28]. Det er naturlig at blant annet pinnekonfigurasjon, minne, lagringspartisjoner mm. vil variere etter gjeldende maskinvare. Uten noe form for verktøy, så må hvert program tilpasses til utstyret som det kjøres på. Kortsiktig, så kan man kanskje nøyes med å definere variabler for maskinvare i kildekoden, men når mengden arkitekturer som støttes er såpass omfattende og voksende som den er i Zephyr's tilfelle, så kan dette utvikle seg til et problem. Formålet med Device Tree er at man for eksempel definerer et alias som `UART_RX` eller `UART_TX`, der hver støttet mikrokontroller tilordner nevnte alias til rette pinne hos seg selv. Per versjon `v3.0.0` støtter Zephyr Project **317** forskjellige brett, og trenger derfor et slikt system. For å få et tall på dette, laget vi et script som tellet de for oss. [1]

Listing 1: nRF9160DK Device Tree eksempel

```
{
    model = "Nordic nRF9160 DK NRF9160";
    compatible = "nordic,nrf9160-dk-nrf9160";

        leds {
                compatible = "gpio-leds";
                led0: led_0 {
                        gpios = <&gpio0 2 0>;
                        label = "Green LED 1";
                };
    ...
```

3.9.8 KConfig

Det er ofte hensiktsmessig å legge til funksjonalitet i applikasjoner for blant annet feilsøking, loggføring eller annen ekstra funksjonalitet. Videre, så trenger man en mekanisme for å utvelge eller skru av og på nevnte funksjonalitet. I mindre applikasjoner, for eksempel i et enkelt Arduino-script, kan man nøyes med å definere variabler i kildekoden. Zephyr er derimot et helt rammeverk, og inneholder en såpass bred mengde funksjonalitet og konfigurering at det ville strittet imot hele hensikten til et rammeverk å fylle kildekoden med konfigureringsvariabler. Av denne grunnen alene, så inkluderes det er verktøy kallet **KConfig**.[29]

KConfig fungerer ved at hvert prosjekt, modul eller bibliotek inkluderer deres egen KConfig-fil. KConfig er da et eget steg i byggeprosessen, der variablene blir inkludert i C-kildekoden, som om man skulle definert dem som faktiske C-variabler. I figuren un-

[1] https://github.com/IELET2900-E2223/dt-counter

der ser man blant annet på linje 3 den boolske konfigurasjonsvariablen `POWER_SAVING_MODE_ENABLE` og på linje 10 at konfigurering for Zephyr sitt loggføringsbibliotek er lagt inn. Zephyr Kernel har fått sin egen meny på linje 12 og nedover, da denne er såpass massiv.

Listing 2: Utdrag fra Cloud Client Sample konfigurasjon

```
1  menu "Cloud Client Sample"
2
3  config POWER_SAVING_MODE_ENABLE
4          bool "Requests PSM from cellular network"
5  endmenu
6
7  module = CLOUD_CLIENT
8  module-str = Cloud client
9  source "${ZEPHYR_BASE}/subsys/logging/Kconfig.template.log_config"
10
11 menu "Zephyr Kernel"
12 source "Kconfig.zephyr"
13 endmenu
```

Listing 3: Bruk av KConfig i kildekoden

```
1 #if defined(CONFIG_POWER_SAVING_MODE_ENABLE)
2     ....
3 #endif
```

3.9.9 Toolchain

Kildekoden må til slutt kunne kjøres på den mikrokontrolleren som anvendes. Zephyr kan som tidligere nevnt kjøres på et bredt utvalg av mikrokontrollere, som hver for seg er basert på forskjellige prosessorarkitekturer, der iblant `x86`, `POSIX`, `RISCV` og en masse forskjellige ARM-arkitekturer. nRF9160 bruker en `ARM Cortex-M33`-prosessor, som bygger på `ARMv8-M` arkitekturen. Vi trenger dermed et verktøy som kan oversette koden vår til de instruksene som finnes i `ARMv8-M`. nRF Connect SDK bruker i denne sammenhengen toolchain-en `GNU Arm Embedded`. [30]

3.10 Om sanntidsprogrammering

En fersk ingeniørs første kontakt med mikrokontrollerverden er ofte via Arduino. Arduino-applikasjoner avgrenses ofte til kun ett formål, og kildekoden holdes derav ganske simpel. *Programarkitekturen* som brukes i Arduino eller tilsvarende simple applikasjoner

heter *Super Loop*. Koden er satt opp som én stor funksjon som går kronologisk fram og gjentar seg selv uendelig, herav navnet. [31]

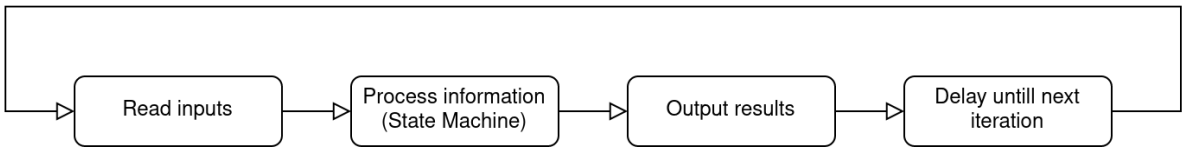


Figure 6: Super Loop programarkitektur

Nå om dagen blir mikrokontrollere tildelt langt flere oppgaver enn før. IoT-industrien krever et hav av forskjellig konnektivitet som LTE, Bluetooth, Wi-Fi eller lignende, i tillegg til at applikasjonen skal være så strømeffektiv som mulig. Uten å gå for mye inn i de nevnte standardene for konnektivitet, så må man være oppmerksom på at hver enkel standard krever at det regelmessig sendes en pakke for å opprettholde forbindelsen. Lengden på denne intervallen kan variere, men ignoreres den, så vil man miste forbindelse. Dette er er i de fleste sammenhenger er uønsket. Videre, så kan være vanskelig å tilpasse et simpelt Super Loop til dette, spesielt oppå all den bestående funksjonaliteten, så industrien beveger seg imot en annen type programarkitektur ved navn *sanntidsprogrammering*. I sanntidsprogrammering bruker man et RTOS, som introduserer mange nye konsepter som sammen muliggjør parallel prosessering.

3.10.1 Tråder

Det første konsept man må lære seg innen sanntidsprogrammering er tråder. Man kan anse en tråd som en frittstående og uavhengig prosess, som ofte er dedikert til ett enkelt formål. Ideelt så ønsker tråder å få kjøre så mye som den vil uten avbrytelser, men i realiteten er ikke dette mulig. Prosessorer i mikrokontrollere har ofte ikke mye mer enn én kjerne å rutte med, så tråder må dele CPU-tiden. Mange RTOS har innebygd funksjonalitet for å kunne gi hver enkelt tråd en prioritering. Prioriteringsrekkefølge håndteres forskjellig etter hvilket RTOS man benytter, men vanligvis blir tråder tildelt et tall, der enten høyere eller lavere tall bestemme prioriteten.[32]

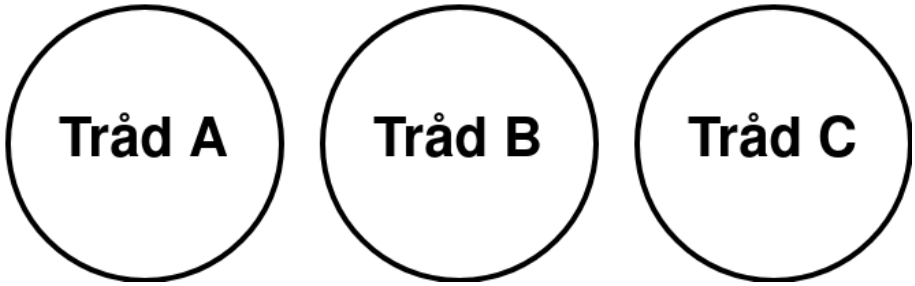


Figure 7: Tråder

3.10.2 Ressurser

Tråder trenger ofte tilgang til ressurser for å kunne gjøre sin jobb. Dette kan være alt fra sensordata og minne til terminalutganger og flash. Det er viktig at disse ressursene behandles riktig, ellers så kan uventede feil skje. I scenariet under, der to tråder begge er avhengige av å lese og skrive til samme fil, ser vi for eksempel at informasjonen som Tråd B skriver til filen aldri vil bli lagret. Begge tråder henter nyeste versjon av fila, men siden Tråd A tar litt lengre tid for å bli ferdig, så overskrives Tråd B sin informasjon. For å komme rundt dette problemet trenger vi altså en mekanisme som enten kan låse en ressurs, eller i verste fall signalere når den er klar til å bli behandlet.

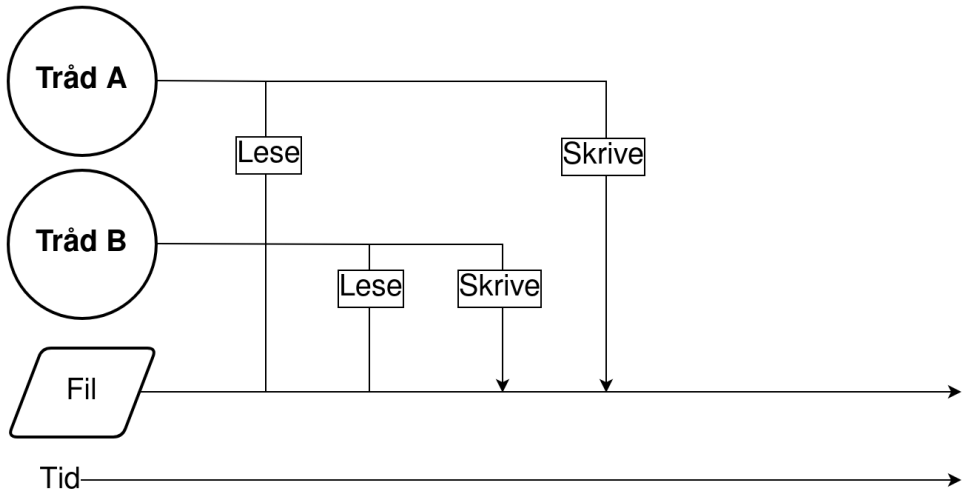


Figure 8: Scenario – Feilhåndtering av ressurser

3.10.3 Mutex

En mutex er en mekanisme innen sanntidsprogrammering som brukes til å låse ressurser. Det finnes to handlinger som kan påvirke tilstanden til en mutex; `lock` og `unlock`. For å få tilgang til en ressurs som er beskyttet av en mutex, så må den være ulåst. Når en tråd får tilgang til en ressurs og dermed låser den, tar tråden eierskap av mutex-en, og kun den samme tråd kan deretter låse ressursen opp igjen. Alle andre tråder som i mellomtiden ønsker tilgang til ressursen, må stille seg i kø og vente. Står det flere tråder i kø, blir prioriteringsmekanismen anvendt. [33]

En mer virkelighetsnær måte å forklare funksjonen til en mutex på, er å bruke *baderom-sanalogien*. Tenk dere et helt vanlig hus med ett bad. Badet er en ressurs, og låsen sitter på innsiden av døren. Hvis noen forsøker å bruke badet mens det er opptatt, blir de møtt av en låst dør og må derfor stå og vente.

3.10.4 Semaphore

En semaphore brukes innen sanntidsprogrammering til å vente på og signalere at en ressurs er klar. Det finnes to typer semaphores; *binary sempahore* og *counting semaphore*. Alle semaphores har to deler, en teller og en liste med tråder i kø. I en binary semaphore vil telleren kun gå mellom 0 og 1, mens den i en tellende semaphore kan telle til et forutbestemt tall. En binary semaphore brukes i all hovedsak til synkronisering av tråder, mens en counting semaphore brukes til et fenomen ved navn *mututal exclusion*, som innebærer det å kun la N tråder få tilgang på en ressurs til en gitt tid. [34]

Når en tråd bruker kommandoen `wait` på en semaphore, så vil den minke sin teller med én, gitt at det er flere igjen. Hvis ikke, så vil tråden bli plassert i køen inntil det blir ledig. Forskjellen fra mutex er at ingen tråd holder eierskap semaphore-en, og at dermed en hvilken som helst tråd kan kalle kommandoen `signal`. Et eksempel på god bruk av denne funksjonaliteten er når én tråd driver å fyller opp en buffer med data, mens en annen tråd tømmer den.

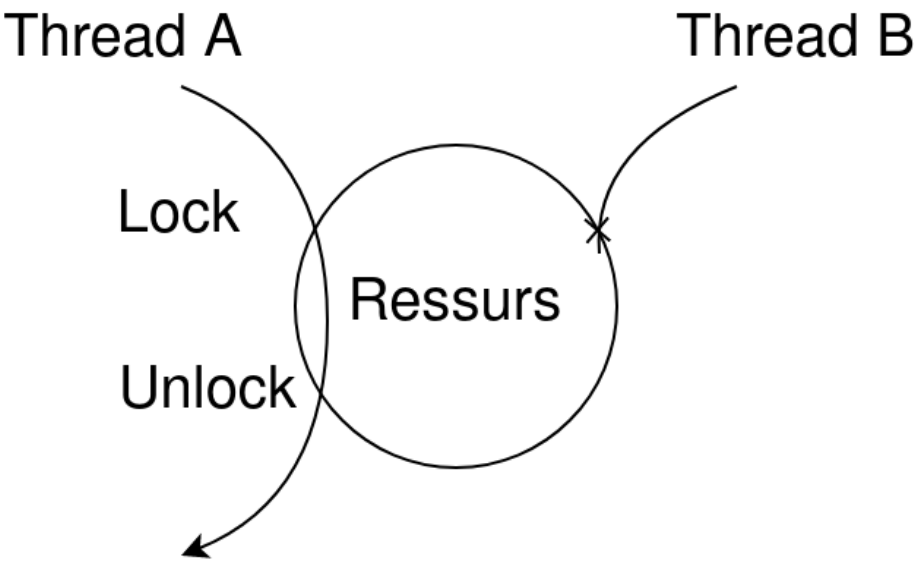


Figure 9: Ressurs beskyttet av mutex

3.10.5 Deadlocks

Når man jobber med tråder og låsing av ressurser, så kan det oppstå et fenomen kallet *deadlocks*. Dette er når systemet går helt i stå og står uten noe måte å starte opp igjen[35]. En deadlock kan for eksempel oppstå i et scenario der to tråder deler på to ressurser. Begge trådene må ha tak i begge ressursene for å gjøre seg ferdig. Hva skjer da hvis begge trådene tar én ressurs hver samtidig? Da ender man med en

deadlock. Et annet godt dokumentert eksempel på nettopp denne typen deadlock er *Dining Philosophers Problem*[36]

3.11 nRF9160 modem og intern kommunikasjon

nRF9160 har et modem som håndterer svært mye logikken og rutinene rundt LTE, GPS, datahåndtering etc. Den har en egen prosessor og eget minne som utviklere ikke rører. Modemets oppgaver inkluderer, men er ikke begrenset til:

- Håndtere kommunikasjon med LTE nettverk
- Innhente og lagre informasjon om LTE-celler
- Innhente, omregne og formatere GPS-data
- Behandle, sende og motta skydata
- Behandle og lagre A-GPS og P-GPS data

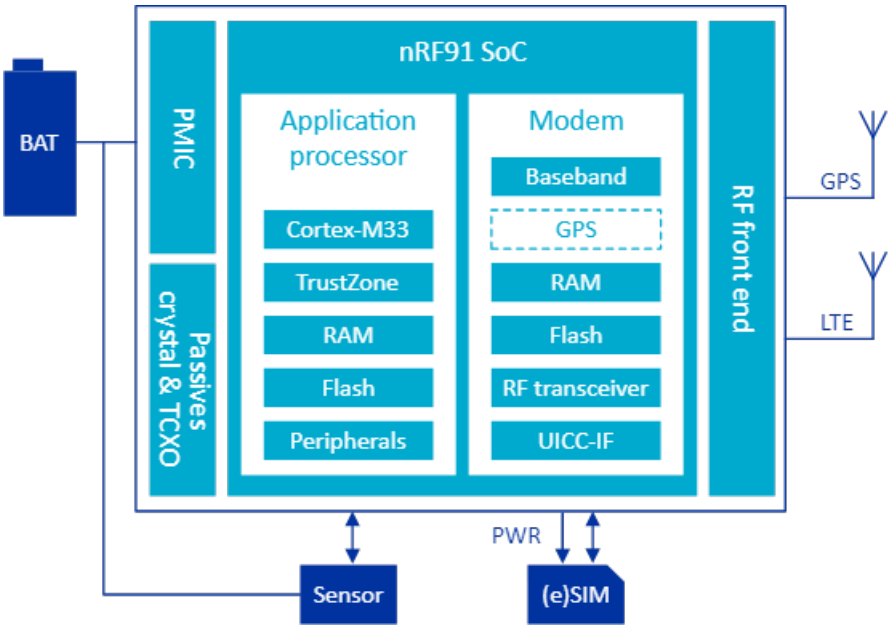


Figure 10: nRF9160 blokkdiagram

3.11.1 AT commands

All kommunikasjon mellom nRF9160 og modemet gjøres ved hjelp av AT Commands (Hayes Commands). Syntaksen til AT commands er standardisert og bruken av AT

Commands er utbredt i kommunikasjon med modemer. Propritære Nordic Semiconductor kommandoer starter med ”%” mens standard telefon- og sms-relaterte kommandoer starter med ”+”.

Utviklere slipper i praksis også å håndtere AT-kommandoer direkte, da alle ønsket funksjonalitet har ferdige funksjonswrappers gjennom biblioteker som Nordic tilbyr. Hvis man f.eks ønsker å slå på LTE PSM-modus kan man enkelt kalle rutinen `lte_lc_psm_req(true)` som sender følgende AT-commands til modem:

```
nrf_modem_at_printf("AT+CPSMS=1,,,\"%s\",\"%s\"",psm_param_rptau, psm_
                                    param_rat);
```

På denne måten kan utviklere slippe å forholde seg til AT-commands, og i stedet fokusere på sin egen kode. Det kan dog i noen tilfeller være gunstig å kommunisere direkte med modemet, for eksempel ved feilsøking eller testing av enkeltfunksjonalitet, som å fremtvinge enten NB-IoT eller LTE-M. Nordic har av denne grunnen utviklet et verktøy ved navn *LTE Link Monitor*.

3.11.2 Handlers

Når ulike hendelser fra modemet skal kommuniseres til programmet vårt, krever dette et gitt oppsett for ulike hendelser. Her bruker man en switch...case-struktur som kalles fra modemet. Case’ene er predefinerte og må ha samme navn som modemet kaller for å fungere, men innholdet i dem kan utviklere velge selv. Under følger et eksempel på en GPS-handler:

```
1  static struct nrf_modem_gnss_pvt_data_frame pvt_data;
2
3  static void gnss_event_handler(int event_id)
4  {
5      int err;
6
7      /* Process event */
8      switch (event_id) {
9      case NRF_MODEM_GNSS_EVT_PVT:
10         /* Read PVT data */
11         err = nrf_modem_gnss_read(&pvt_data, sizeof(pvt_data), NRF_MODEM_GNSS_DATA_PVT);
12     ...
13 }
```

Casen `NRF_MODEM_GNSS_EVT_PVT` kalles hvis det finnes ny PVT-data tilgjengelig, og vi kan velge å lese denne dataen i den tilhørende casen.

Når man har satt opp en struktur for å håndtere meldinger fra modemet, må man også iverksette den ved følgende rutine, hvor argumentet er navnet man ga til handleren:

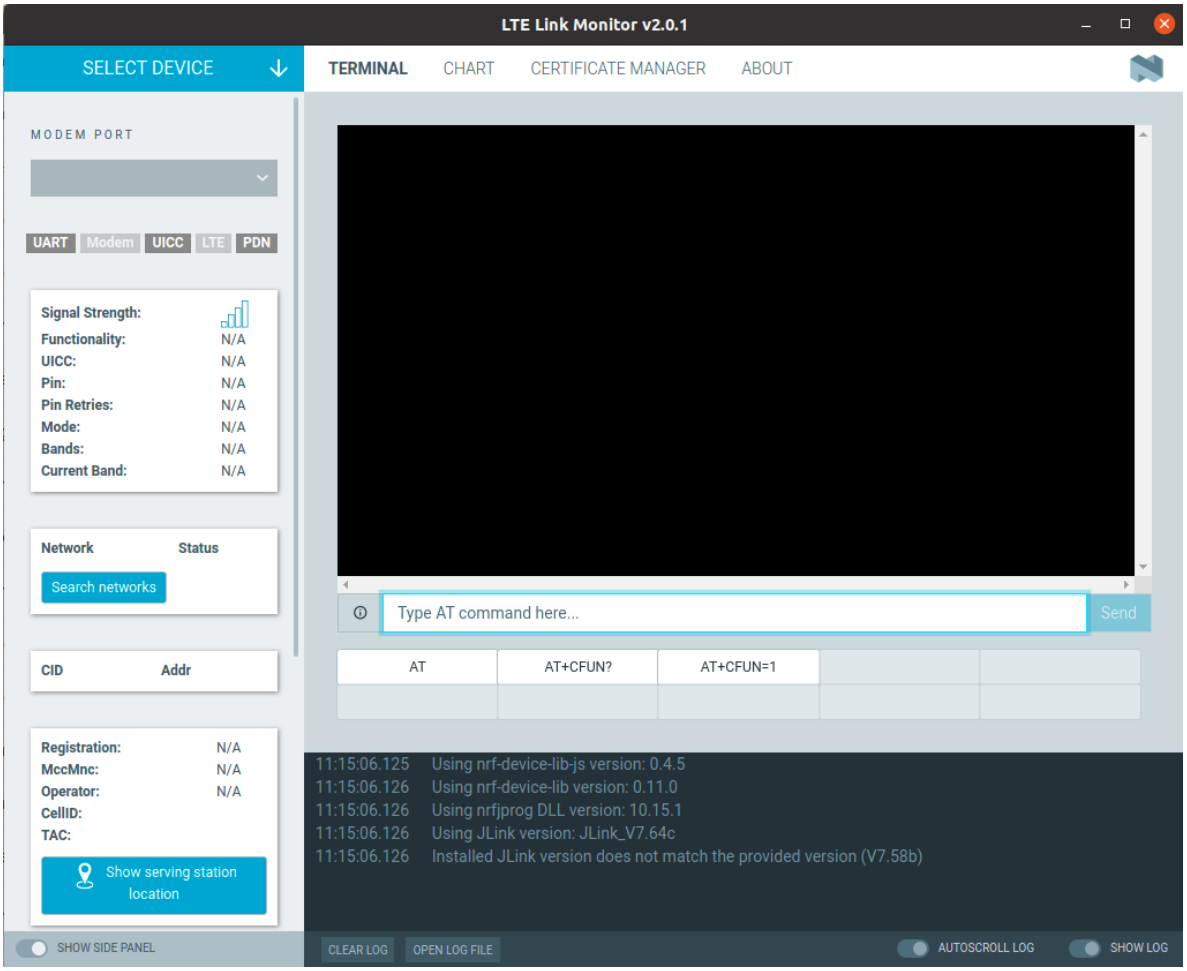


Figure 11: LTE Link Monitor

```
1 err = nrf_modem_gnss_event_handler_set(gnss_event_handler);
```

Hvor funksjonens returnverdi er ulik null dersom det ikke ble gjennomført.

Denne logikken og strukturen brukes også av LTE-hendelser, Cloud-hendelse etc.

4 METODE OG MATERIALER

4.1 Hardware

4.1.1 Designfilosofi

Den produserte prototypen hadde som hensikt å legge grunnlaget for et endelig produkt. Siden dette var gruppens første møte med RF PCB design var et av målene å holde kretsen enkel og plass romslig. Første prioritet var å få plass skjemategning av det absolutt mest nødvendige for å realisere produktets hensikt. For dette trengtes selvfølgelig nRF9160, en LTE og GNSS antenne, LNA filter, SIM-kortholder, SIM EMI filter, SWD debugger/flashing kontakt og strømforsyning. Siden vårt produkts hensikt var å trådløst spore kveg måtte kretsen forsynes ved hjelp av en batteriløsning. Akkurat hvilken batteriløsning som var best passende i en virkelig scenario var bestemt som en mindre viktig sak under prototype utlegg. Det viktigste var at løsningen skulle gi trådløs funksjonalitet til testing ut i åpne områder.

4.1.2 Skjemadesign og valg av komponenter

Krets-skjemategningene bruker standardiserte europeiske symboler og er laget for å være oversiktlig og lesbare. Kretsen ble delt opp i 7 deler:

- Main
- nRF9160
- Power og UART
- SIM-kort
- GPS / LTE-antenner
- nPM1100
- Akselerometer

Skjemaene er koblet sammen ved hjelp av porter og signal harness og koblet sammen med sheet symboler. Denne løsningen var gunstig og gjorde det enkelt å samarbeide med verktøyet Altium 365 som er basert på Git.

Det ble valgt å bruke 0201 og 0402 størrelse på de fleste av de passive komponentene i kretsen fordi vi hadde mulighet til å få kortet produsert på produksjonslinjen til Nordic Semiconductors AS. Vi valgte å unngå for små komponenter for å holde kortet enkelt,

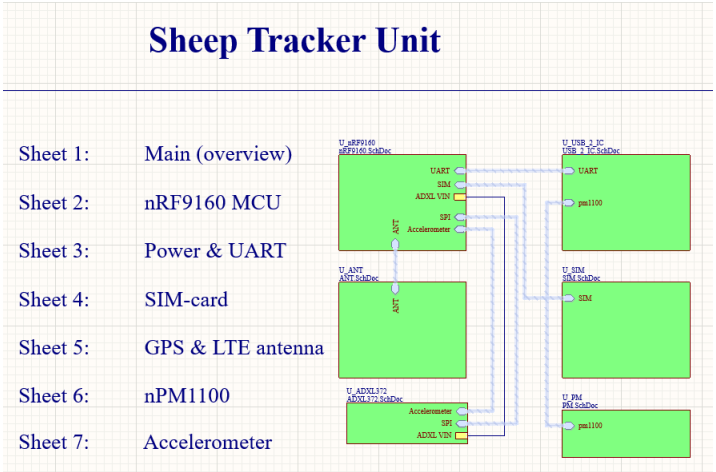


Figure 12: Signalflyt mellom kretsskjemategninger

billig og oversiktlig å feilsøke på. På de resterende komponentvalgene ble valgt å holde seg til komponenter som ble brukt på nRF9160-DK eller Thingy:52 på grunnlag av tilgjengelighet og leveringstid som var en kritisk parameter.

Basisdesign:

Vårt design tok utgangspunkt i et basis i Hardware Design Guidelines [37] som gir veiledende informasjon om krets-og pcb design basert rundt nRF9160. I maskinvareretningslinjene ligger det informasjon om anbefalte komponentvalg og komponentplassering. Som basis for videre skjemategning er det lagt ved et startpunkt design sett i figur (13). Nordic Semiconductor tilbyr også nedlasting av Altium designer prosjekt filer med alle sine utviklingssett.

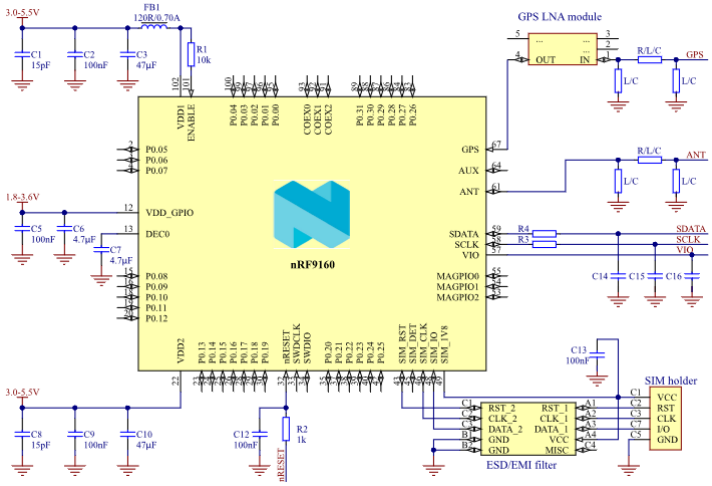


Figure 13: Anbefalt utgangspunkt for skjemategning [37]

USB til UART bridge:

Det ble tenkt hensiktsmessig å inkludere en USB til UART bridge først å fremst for debugging, men også for flashing av applikasjoner. Zephyr har støtte for oppstart-slasteren MCUBoot som gjør det mulig å oppdatere fastvaren via UART, Bluetooth eller over nettet. For opplasting via UART kan man bruke verktøyet mcumgr. Dette er et separat verktøy som ikke kommer standard ved nedlasting av nrf sdk.
I planleggingsfasen ble det gjort et valg av komponent: Silicon Labs CP210. Denne ble valgt fordi den også hadde en indre 5 V til 3 V spenningsregulator. Hele kretsen kunne da bli forsynt via USB kabel under testing. Ulike konfigurasjoner er vedlagt i CP2012 sitt datablad [38]. Konfigurasjon 1 gir VBUS forsyning for UART funksjonalitet samt regulator.CP2012 er internt stemt til å ligge rundt det riktige impedansnivået 45 Ohm (38 Ohm typisk) og trenger derfor ikke ytre impedans stemming.

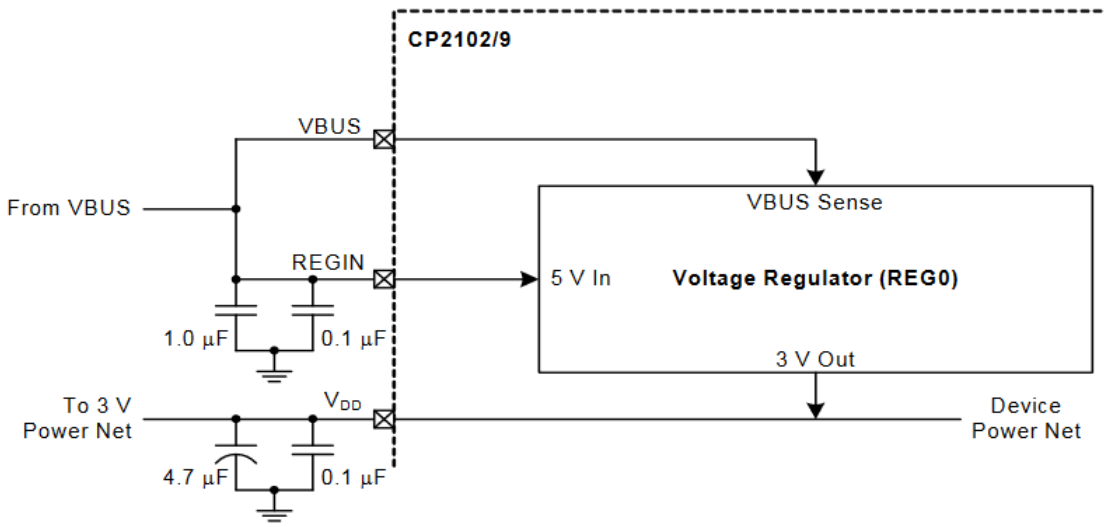
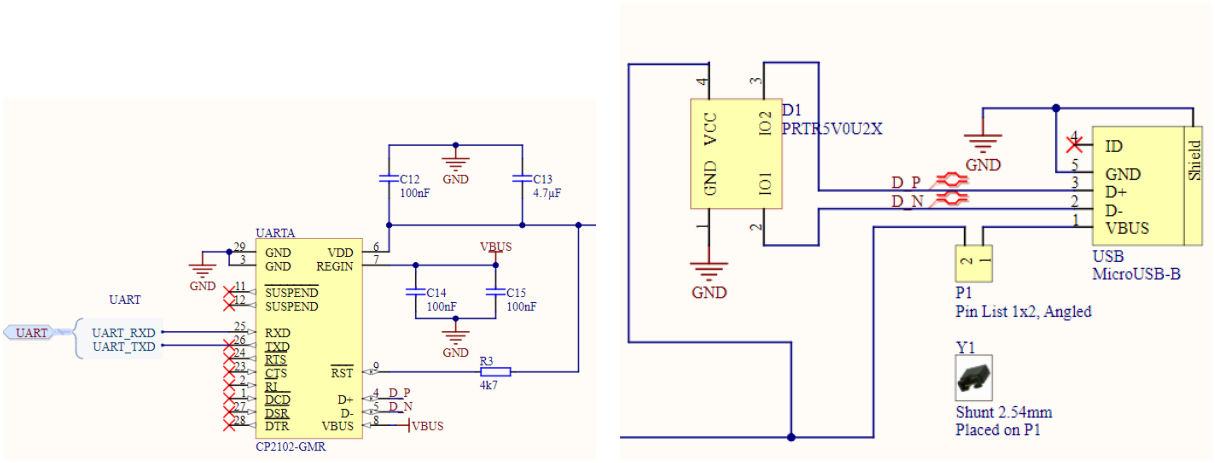


Figure 14: VBUS forsynt CP2102 med 3 V regulator utgang

Det ble anbefalt å ha en ESD beskyttelse diode på USB inngangen som er en tre veis diode koblet til jord. Denne skulle kobles til utgangene D +/- og VBUS. Dette for å unngå at store strømmer som oppstår på grunn av statisk elektrisitet å gå videre inn i kretsen. Videre er det også anbefalt en ekstra avkoplingsmotstand på 4.7 μf mellom VDD og jord hvis 3 V regulatoren skal gi strøm til andre enheter.
Dette ga følgende skjemakonfigurasjon:



(a) CP2012 skjemakonfigurasjon

(b) USB-inngang med ESD-diode og pinner til strømtrekk

Figure 15: CP2012 og USB konfigurasjon i prototype skjema

PMIC - nPM1100 Nordic Semiconductors nPM1100 er en power management IC som inneholder en batterilader, en systemregulator og en steg ned DC/DC konverter. Dette systemet tar opp svært lite PCB areal og er ment for å drive lav effekt kretser. Den ble inkludert i prototype som kuriositet, men var ikke tenkt til være en del av det endelige produkt konseptet. Dette fordi produktet har som mål å være mest mulig energieffektiv og det vil derfor gå lange perioder før utskifting eller ladning av batteri er nødvendig.
Spesifikasjoner:

Batterilader	
Forskrift overholdet	JEITA
Termineringsspenning	4.1 V or 4.2 V valgbart
Ladestrøm	20 mA to 400 mA
Inngangsregulator	
Inngangsspenning	4.1 to 6.7 V
Utgangsspenning	3.0 to 5.5 V uregulert
Overspenningsbeskyttelse	20 V transient
USB strøm grense	100 mA or 500 mA
Buck regulator	
Utgangsspenning	1.8, 2.1, 2.7 or 3.0 V
Strømgrense	150 mA utgang
Stillegående strømmer (quiescent current)	800 nA typisk, 460 nA i ship mode
Batterispenning	2.3 V til 4.35 V
Operasonstemperatur	-40 °C to 85 °C

Blokkdiagram:
SYSREG:
Systemregulatoren er en linear drop out regulator (LDO) som er forsynt av VBUS. SYSREG regulerer spenningen til 5 V fra en inngangsspenning som kan ligge mellom 4,1

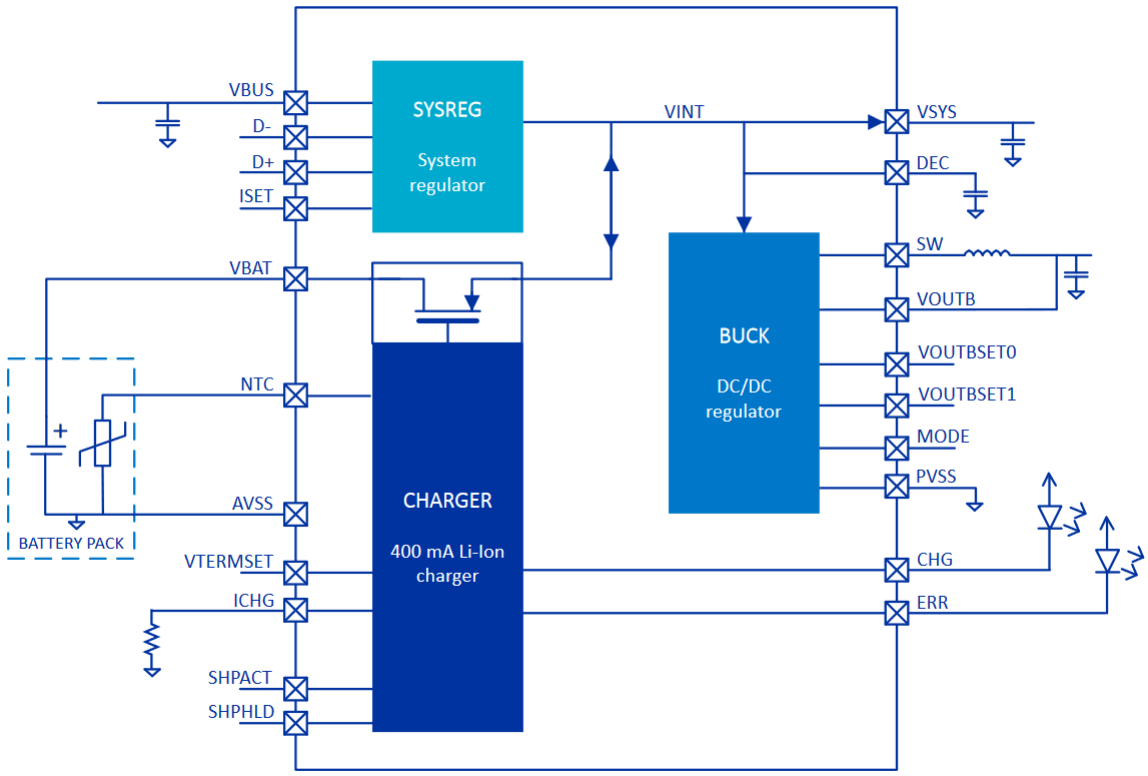


Figure 16: Blokkdiagram som viser systemfunksjon i generell konfigurasjon

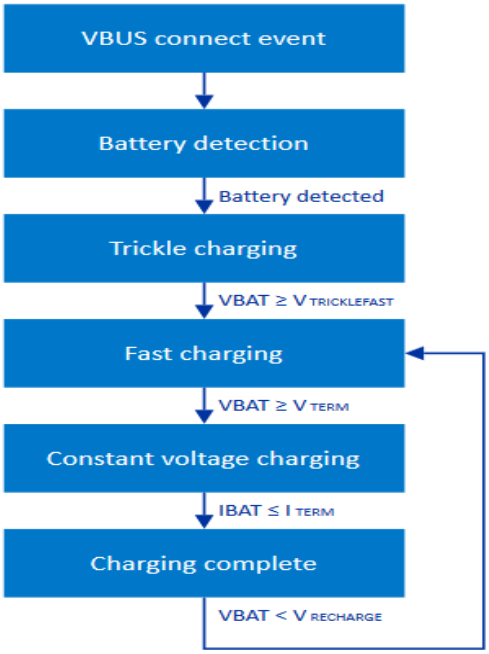
V til 6.7 V. Den har også en overspenning beskyttelse på 20 V, USB port detection og VBUS strøm begrenser (100 mA og 500 mA).

CHARGER:
Systemets lader er ment for batterier med Lithium-Ion / Lithium-Polymer kjemi. Laderen er linear og har en konfigurerbar ladestrøm (20 mA til 400 mA). Terminering spenning er valgbar gjennom VTERMSET pinnen (4,1 V eller 4,2 V). Laderens syklus er illustrert i figur 17a og 17b.

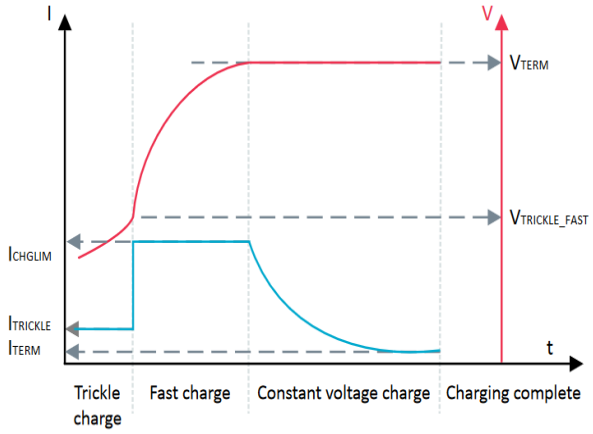
Lading skjer først om VBUS er tilkoblet og batteri er oppdaget på VBAT pinnen. Figur18 viser nPM1100s ladesyklus. Ladestrøm er konfigurerbar ved å knytte en motstand mellom ICHHG og jord. Formel 6 viser hvordan man beregner motstand for ønsket ladestrøm.

$$R_{ICHG} = \frac{625}{I_{CHGLIM}} - 1562.5 \tag{6}$$

nPM1100 har mulighet for termisk batteribeskyttelse. Dette kan oppnås ved å enten koble til en ekstern NTC thermistor eller koble til en batteripakke med en integrert. Selve thermistor må kobles til pinne NTC.



(a) Stegvis ladesyklus



(b) Ladesyklus strøm I over tid t

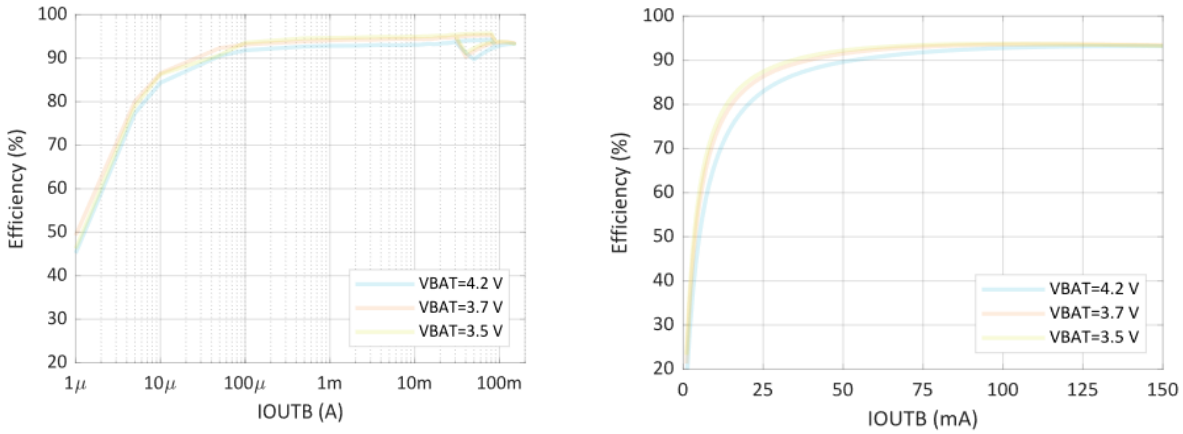
Figure 17: nPM1100s ladesyklus

BUCK:
Buck converteren har to forskjellige operasjonsmoduser: hysterese som er tilgjengelig med AUTO konfigurering og PWM som er tilgjengelig med PWN og AUTO modus. Hysterese modus gjør at man får god effektivitet under lav last. PWM modus gir stabil operasjon ved å sette buckconverter til en fast bryter frekvens F_{buck} (typ 3.6 Mhz). Man setter modus ved å sette MODE pinnen lav eller høy. Buck converteren har konfigurerbar utgangspenning på 1.8 V til 3 V. I vår konfigurasjon er det valgt 3 V.

Ship mode:
Ship mode reduserer stillegående strømmer ved å isolere batteriet (460 nA vs 800 nA). For å slå på må SHPACT pinnen være høy, VBUS koblet i fra og SHPACT holdes høy for minst 200 ms. Siden SHIPACT har en indre pull up motstand kan den kobles til enn GPIO. For å gå ut av ship mode kan man enten koble til VBUS eller holde SHPLD lav for minst 200 ms.

Effektivitet:
System effektivitet er avhengig av utgangsspenning og modus. Hysterese operasjon (auto modus) gir en svært mye bedre effektivitet under lave laststrømmer.

Feil under modus bytte:
nPM1100 revisjon 1 har en feil ved seg. Denne feilen gjør at buck converter slår seg plutselig av under modusbytte fra PWM til hysterese modus. Det er flere måter man



(a) Effektivitet i auto modus (hysterese)

(b) Effektivitet i pwm modus

Figure 18: Effektivitet i auto modus vs fixed frequency modus

kan arbeide rundt denne feilen. For å unngå at konverteren slår seg av må man enten operere med AUTO modus bare med last under 65 mA eller så kan man tvinge konverteren i PWM modus ved å sette MODE pinnen høy. Sistnevnte øker den stillegående strømmen med flere mA.

Evalueringskort
Nordic Semiconductor selger et evalueringsbrett for testing av nPM1100s ulike funksjoner. Brettets applikasjonsnotat viser blant annet hvordan man gjør måling av forsyning av DK serien ved hjelp av Power Profiler 2.

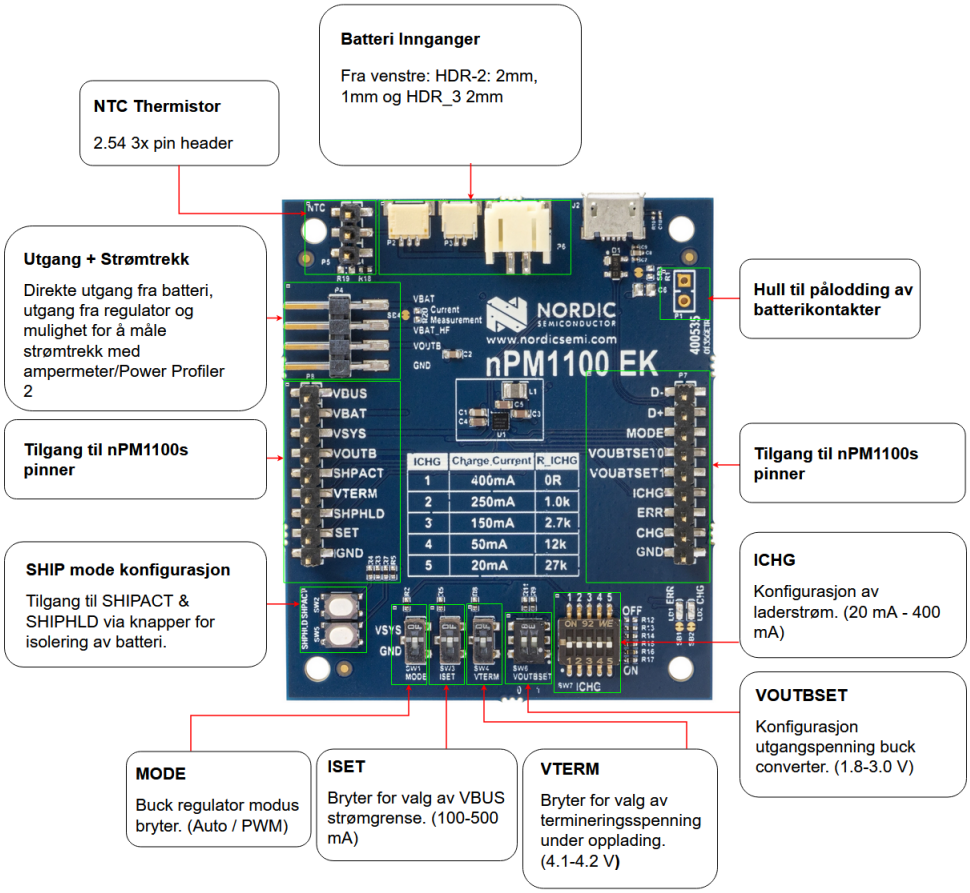
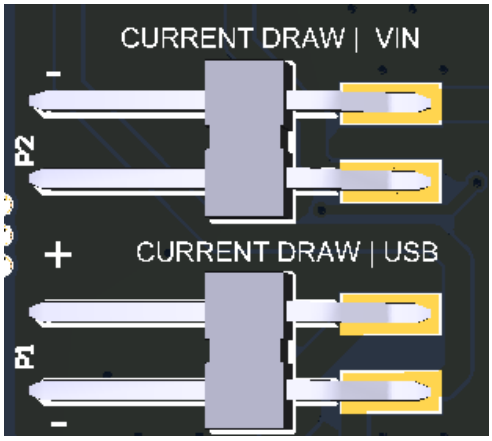


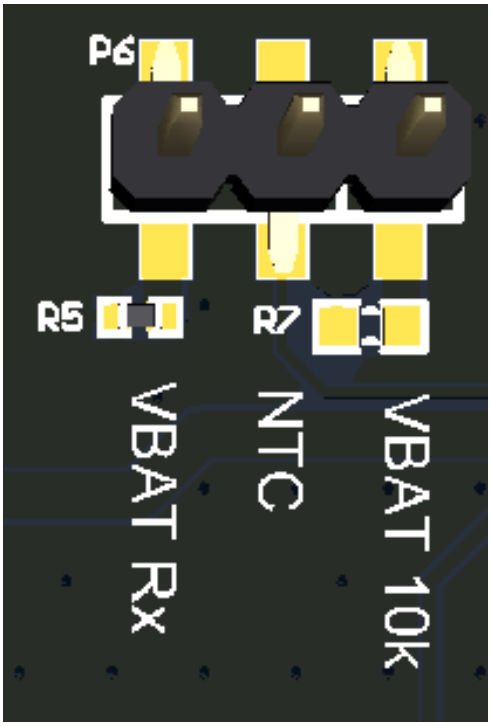
Figure 19

Strømtilførsel og testpunkt til strømtrekk:

Prototypen ble designet til å kunne tilføres strøm enten ved hjelp av batteri eller direkte via USB. For å måle strømtrekk ble det lagt til to stk 2.54 mm headere i serie. En direkte tilkoblet den universale strømtilførselen (direkte til nRF9160 VDD) og en mellom VBUS. Sistnevnte ga mulighet å differensiere eventuelle effekt tap mellom komponenter og forsyning til nRF9160. For tilkobling av batteri kan man enten koble batteriet direkte på nRF9160 eller gjennom nPM1100. Hvis man kobler til batteriet direkte må batteriets spenning ligge mellom 3 til 3.6 V for å dekke VDD1/VDD2s og VDD_GPIO forsyningsbehov. Gjennom nPM110s buck converter kan batterispenningen ligge mellom 4.1 til 6.7 V. nPM1100 kan også lade batterier med satt ladestrøm på 200 mA. Denne ladestrømmen kan manuelt konfigureres på prototypen. Den har et ledig footprint for pålodding av motstand.



(a) Headere for måling av strømtrekk



(b) Headere for konfigurasjon av ladestrøm

Figure 20: PCB headere for strømtrekk og ladestrøm konfigurasjon

4.1.3 Hensyn ved design av PCB

Antenneplassering og klareringsområde
En antennes plassering er kritisk for god ytelse som ligner spesifikasjoner gitt i datablad. Plassering og klareringsområde avhenger helt av typen antenne man velger. For vårt utlegg bruker vi en chip antenne og en keramisk antenne. Disse trenger begge et klareringsområde og har best effekt om de posisjoneres spesifikke plasser på PCB. Det er viktig at klareringsområdet er fri for komponenter, hull og kobber. For å gjerde inne utstråling fra antenne kan man bruke via stiching rundt dette klareringsområdet. Prototypen vår vil huse to typer antenner: en GNSS antenne og en LTE antenne.

Valg av antenner:

For valg av antenner har Nordic Semiconductor vedlagt egne retningslinjer: nRF9160 Antenna and RF Interface Guidelines [39]. Her må visse anbefalinger følges som sett i tabell 1. Valg av antenne må gjøres ut fra parametere: spenning standbølge forhold (VSWR), returtap, effektivitet, maks forsterkning og maks inngangseffekt. Ved mange parametere VSWR, retur tap og effektivitet.

Parameter	Krav	Notis
VSWR	$\leq 2:1$ $\leq 3:1$	Anbefalt Minste mulige
Retur tap	$\geq 9.5dB$ $\geq 6.0dB$	Anbefalt Minst mulige
Effektivitet	$\leq 75\%$ $\leq 50\%$	Anbefaling Minste mulige
Maks inngangseffekt	1 W	

Table 1: Spesifikasjonskrav til antenner

Etter undersøkelse av ulike type antenner landet vi på å bruke Ignion NN02-220 for LTE-antenne og Taoglas GGBLA.01 for GNSS-antenne.

GPS antennesammenlikning
Målt ved 1575MHz

	Prototype	**nRF9160-DK**
Modell	GGBLA.01.A	DSGP.1575.18.4.A.02
Effektivitet	83.43%	83.33%
Forsterkning	3.29dBi	4.20dBi
Størrelse [mm]	3.2x1.6x0.5	18x18x4

Table 2: Fra datablad GGBLA.01.A [**?**] og DSGP.1575.18.4.A.02 [40]

LTE antennesammenlikning Målt ved 1710-2690MHz

	Prototype	**nRF9160-DK**	**Thingy:91**
Modell	NN02-220	P822601	FR01-S4-210
Effektivitet	over 75%	76%	over 65%
Forsterkning	3.1dBi	4.4dBi	2.4dBi
Størrelse [mm]	24x12x2	50x8x3	30x3x1

Table 3: Fra datablad NN02-220 [41] og P822601 [42] og FR01-S4-210 [43]

Ignion NN02-220
NN02-220 er en antenne med god ytelse som kan oppfylle de satte krav. Den er også ganske liten sammenlignet med LTE antenner med liknende båndbredde og ytelse på 24 mm x 12 mm. Den dekker to frekvens bånd: 698-960 MHz og 1710-2690 MHz. I antennens datablad er det vedlagt flere målinger gjort på kort av ulik størrelse.

Båndbredde	698-960 MHz	1710-2690 MHz
Gjen. effektivitet	$\geq 55\%$	$\geq 75\%$
Peak Gain	2.3 dBi	3.1 dBi
VSWR	$\leq 3:1$	$\leq 3:1$

Table 4: Målte antenneparametere for Ignion NN02-220 (2.1 LTE Solution)

Medfølgende antennens applikasjonsnotat er det lagt ved parametere angående design av et testet evalueringskort. Matching nettverket og klareringsområdet har blitt brukt som referanse for prototype utlegg. Matching nettverket består av tre parallell og 4 serie 0201 smd footprints. Klareringsområdet er på 5 mm x 12. mm (bredde ganger lengde).

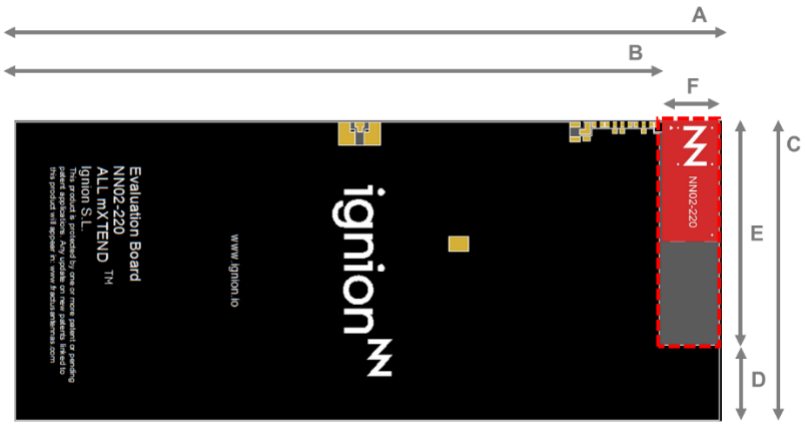


Figure 21: Evalueringskort brukt som referanse under prototype design

Taoglas GGBLA.01

Taoglas GGBLA.01 er en svært kompakt keramisk antenne i SMD format. Den er stemt til å innhente ulike frekvensbånd fra GPS, GLONASS, Galileo og Beidou. Det er følgende frekvensbånd 1575.42MHz, 1602 MHz og 1561 MHz. Den er unik ved at den har en forsterkning som kan sammenlignes med mye større keramiske antenner. Effektiviteten som kan oppnås ligger mellom 64% til 85%. Antennen bør plasseres ved enden av kortet midt på jordplanet for best effekt, men kan også plasseres i hjørnet med god ytelse. GGBLA.01 har et omnidireksjonelt strålemønster og har typisk VSWR lik eller mindre enn 2. Følgende målinger er gjort av produsent (80x40mm jordplan):

Frekvensbånd	1561 MHz	1575.42 MHz	1602 MHz
VSWR (max)	2.0:1 max	2.0:1 max	2.0:1 max
Effektivitet	74.32 %	83.43 %	71.98 %
Forsterkning	2.9	3.29	2.58

Table 5: Målte antenneparametere fra Taoglas GGBLA.01 datablad [**?**]

I databladet er det stilt krav til et 5mmx5mm klareringsområdet sentrert om antennen. Matching nettverket er av typen pi (parallell serie parallell) og det er også to footprints koblet innen for antennens klareringsområde. Disse er koblet til FTE pinnen og er for finjustering av resonansfrekvens.

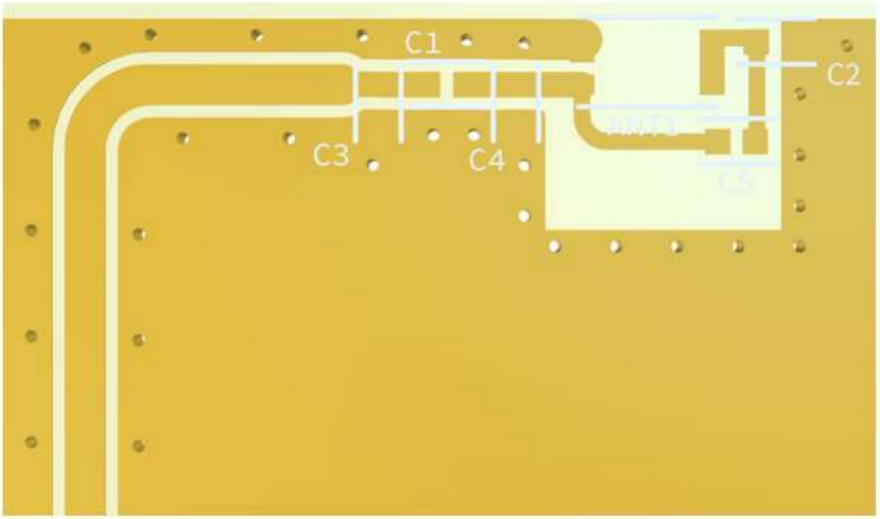


Figure 22: Anbefalt klareringsområde fra datablad [**?**]

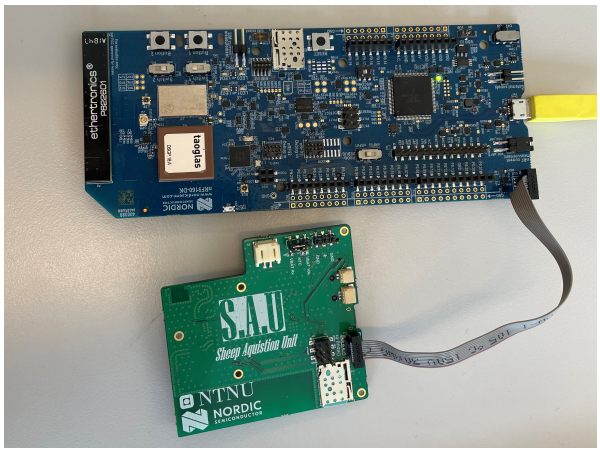
4.1.4 Akselerometer

Prototypen har et akselerometer kommuniserer med nRF9160 over SPI. Den har og muligheten til brukes som kilde til avbrudd. Den bruker MEMS kapasitiv teknologi som gjør formfaktoren veldig kompakt.

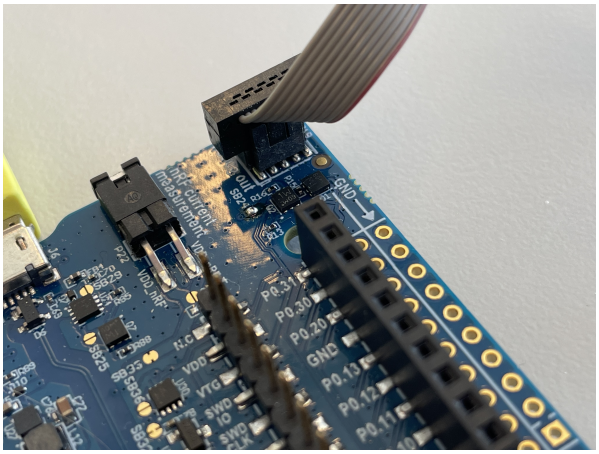
4.1.5 Debugging av prototypekort

For debugging av egendefinerte kort med nRF9160 kan man enten bruke en nRF9160DK/nRF53DK eller en selvstending J-Link programmer.

nRF9160DK finner automatisk ut om et eksternt kort er tilkoblet Debug Out terminalen. Om man ønsker å bruke DK til hovedsaklig debugging kan man lage en loddebro rett ved kontakten (SB24). Dette gjør også at man kan forsyne prototypen gjennom nRF9160 DK, om ikke må man forsyne prototypen med akkurat likt spenningsnivå som enten må være 1.8 V eller 3 V. Utgangspenningen (nivå på GPIO) kan justeres på bryter SW9 på DK.



(a) Debugging av prototypekortet gjennom nRF9160DK



(b) Loddebro for å låse OB Jlink til Debug Out pinnen

4.1.6 Layer stack

En PCB's lagstruktur kan i Altium bli endret gjennom verktøyet layer stack manager. Her kan organisering, bredde og mengde på korets lagstruktur fritt modifiseres. Siden prototypen er ment for å operere i et frekvensområde rundt 2 Ghz er ikke valg av dielektrikum viktig[**?**]. Det ble derfor valgt en ofte brukt firelagstruktur med type 1080 prepreg. Tallet referer til fiberstrukturen til materialet og har en typisk tykkelse på 60 - 70μm, ofte en dielektrisk konstant mellom 3.2-3.7 og resin mengde på rundt 60 % [44]. I vår konfigurasjon ligger materialets dielektriske konstant på 4 og høyde på

sammenlagt 0.17 mm. Topp/bunn-lag og jordlag er av kobber med tykkelse 0.035mm og tyngde 1 oz. I midten ligger det en kjerne på 1.3 mm som isolerer 1/2 fra 3/4.

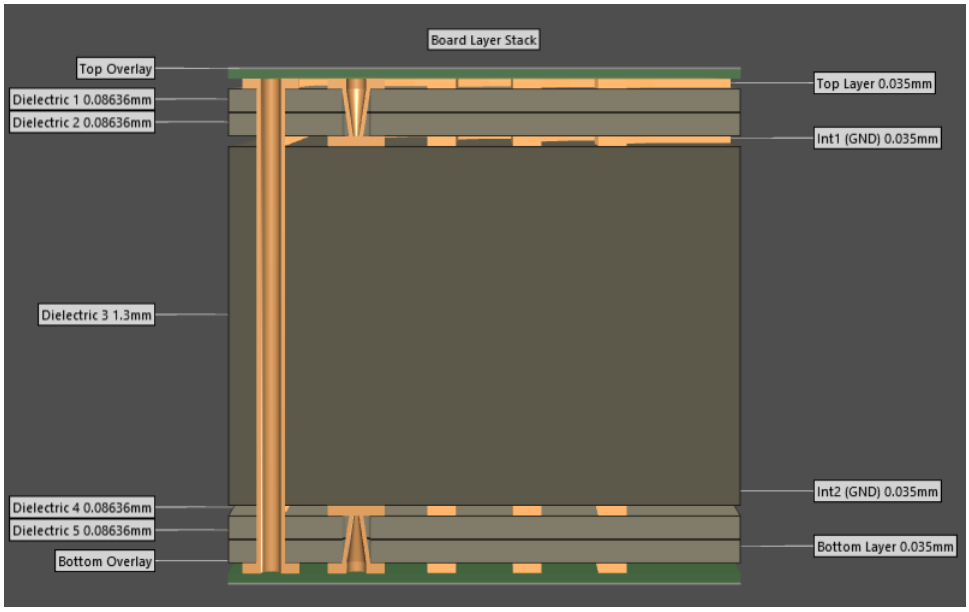


Figure 24: Prototype kortets lagstruktur

4.1.7 Via stiching/sheilding, transmisjonslinje og impedansmatching

Impedansmatching:

I gruppens forsøk på gjøre antennenettverket godt nok til å en fix valgte å bruke en kondensator i serie på 2.2pF og en i parallell på 3.8pF. Målingen nordic gjorde av transmisjonslinjen når matchenettverket kun hadde en 0Ω motstand ble utgangspunktet for kalkylen, se figur 44. Disse verdiene ble kalkulert ved hjelp av et nettverktøy som vist i figur 26. Impedansverdiene fra målingene på 11.310Ω + j16.208Ω ble lagt inn i verktøyet. Det ble valgt å bruke to kondensatorer på grunnlag av at det ville oppstått effekttap i en reell motstand. I følge modellen får vi en VSWR på 1.83:1 som er innen for Nordics anbefalinger på $<$ 2:1 og absolutte krav på $<$ 3:1 som står oppført i "Antenna and RF Interface Guidelines" [39]

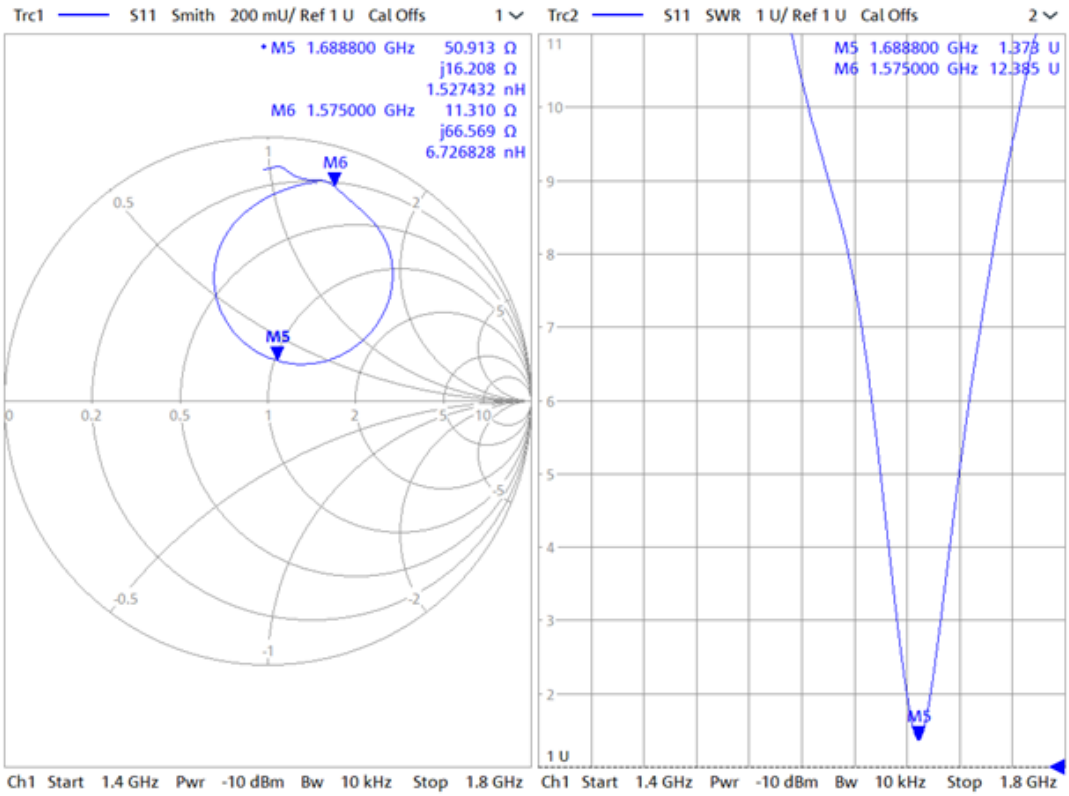


Figure 25: Spektralmåling med kun 0Ω motstand i serie

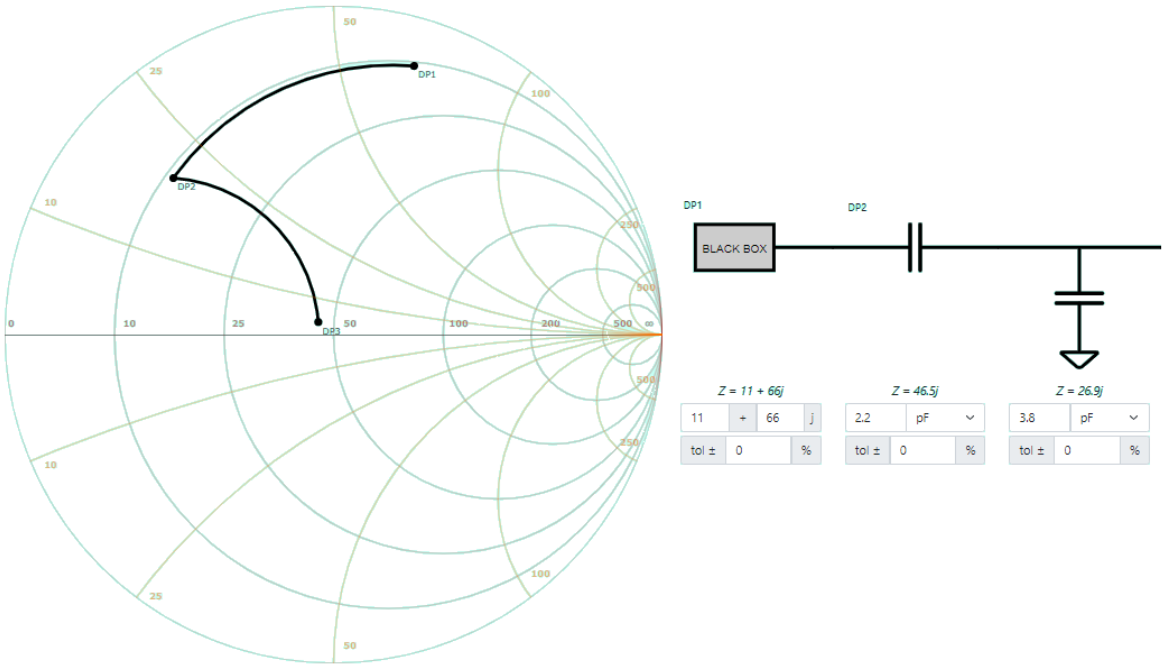


Figure 26: Modellert impedansmatch fra https://www.will-kelsey.com/smith_chart/

Transmisjonslinje: En transmisjonslinje kan defineres i Altium layerstack manager. Dette er en enkel prosess man trenger bare å definere ønsket impedans, type transmisjonslinje og toleranse.

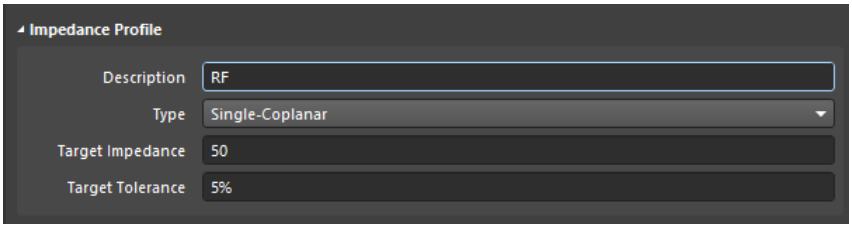


Figure 27: Definering av impedanseprofil

I Altium kan man velge mellom microstrip (standard) eller coplanar. For både USB og RF linjene ble det valgt å bruk et coplanar transmisjonslinje design. Dette fordi sine fordeler over microstrip beskrevet tidligere i teori kapitellet. Alt av komplisert matematikk blir gjort automatisk av Altium.

Via stiching & shielding

Via stiching og shielding kan gjøres automatisk gjennom verktøy i Altium. Man tar i bruk via stiching for å knytte større kobber områder som jord eller power sammen på flerlagskort. I verktøyet velger man nett, viastørrelse og mellomrom mellom hvert hull. Via sheilding verktøyet fungerer på liknende måte som stiching verktøyet. Et nettverk av enkle viaer knyttes rundt transmisjonslinja og gjerder inne strålingen som emitterer.

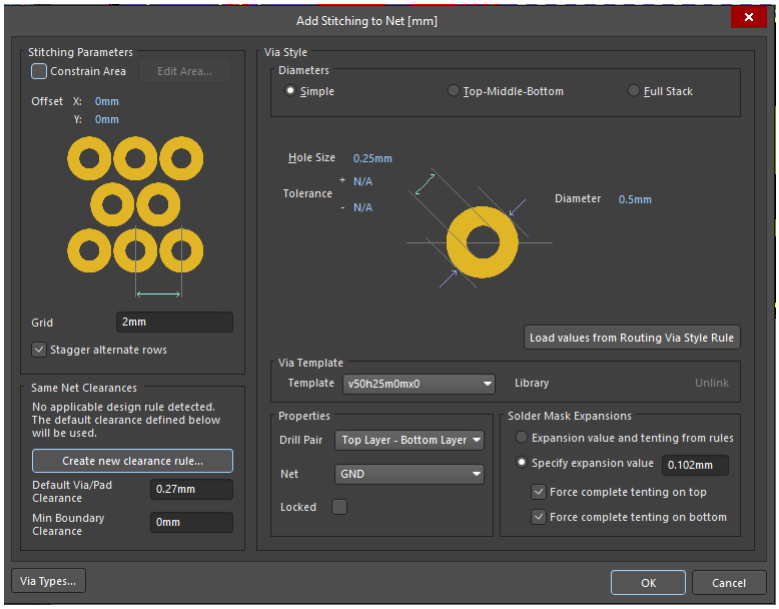


Figure 28: Via stiching i Altium

Sheilding er derfor også kalt via fencing. Det som separerer de to verktøyene er at man kan velge ønsket distanse fra linja, mellomrom mellom i både vertikal og horisontal retning.

4.1.8 LNA

Det ble valgt å bruke en LNA av typen Skyworks SKY65943-11 da denne er den samme som ble brukt i utviklingsbrettet nRF9160-DK og er da enkel for oppdragsgiver å oppdrive og det er stor sannsynlighet for at de har den på produksjonslinjen. Denne enheten er ideell for bruk på GPS/GLONASS/Galileo og er optimalisert for å operere på mellom 1559 og- 1606MHz. Enheten er impedansematchet internt til 50Ω. Strømforbruket når LNA er operativ er 2.9mA ved 1.8V [19].

4.1.9 MAGPIO & COEX

MAGPIO er utganger som er styrt av modemet og kan brukes til å styre eksterne komponenter basert på hvilken frekvens modemet operer på. MAGPIO funksjonaliteten blir satt av AT commands og blir skrevet til enhetens RAM og lagret i NVM. Typisk bruksområde til MAGPIO er å tune antenner basert på hvilken frekvens modemet opererer på.

```
1 \%XMAGPIO=<gpio_0>,<gpio_1>,<gpio_2>,<num_ranges>,<state_0>,<flo_0>,<fhi_0><state_1
      >,<flo_1>, <fhi_1>,...
```

num_ranges settes antall frekvensområder. States setter antall tilstander og utgangsverdien for frekvensområdene basert på binær logikk som vist i tabellen under der tilstand 2 blir representert som 010 binært.

State	MAGPIO1	MAGPIO2	MAGPIO3
1	0	0	1
2	0	1	0
3	0	1	1
etc..			

COEX0 (pin 93) er Nordic Semiconductors proprietære utgang som har samme funksjonalitet som MAGPIO men styrer kun en utgang. Tilsiktet funksjonalitet er kontrollering av LNA til GPS antennen. Utgangen COEX0 er kontrollert av nRF9160s interne modem og kan bli konfigurert til å kontrollere ekstern GPS LNA slik at forsterkeren ikke er aktiv når GPS ikke er i bruk.

Konfigurasjonen for COEX0 settes i Kconfig som AT commands og bruker følgende syntaks:

```
1 \%XCOEX0=<Antall>,<Tilstand>,<Nedre\_frekvens>,< vre\_frekvens>
```

$< Antall >$	Antall frekvenseområder (1-4)
$< Tilstand >$	Tilstanden til COEX0 i frekvensområdene (0-1)
$< Nedre\_frekvens >$	Nedre frekvensgrense
$< Øvre\_frekvens >$	Øvre frekvensegrense

```
1 AT%XCOEX0=1,1,1565,1586
```

Denne AT commandoen konfigurerer COEX0 til å være høy når modemet er i frekvensområdet 1565-1586MHz slik at LNA blir operativ.

Ved bruk av ekstern antenne må dette spesifiseres i Kconfig slik LNA er inaktiv. LNA brukes ikke ved bruk av ekstern antenne.

4.1.10 Konfigurering i Device Tree

I VSCode er det mulig å konfigurere et eget kort gjennom nRF SDK utvidelsen. Hvor man lagrer disse filene er valgfritt, men den beste plasseringen er sammen med alle andre arm baserte konfigurasjoner:

```
1 C:\Users\#\ncs\v1.9.1\zephyr\
```

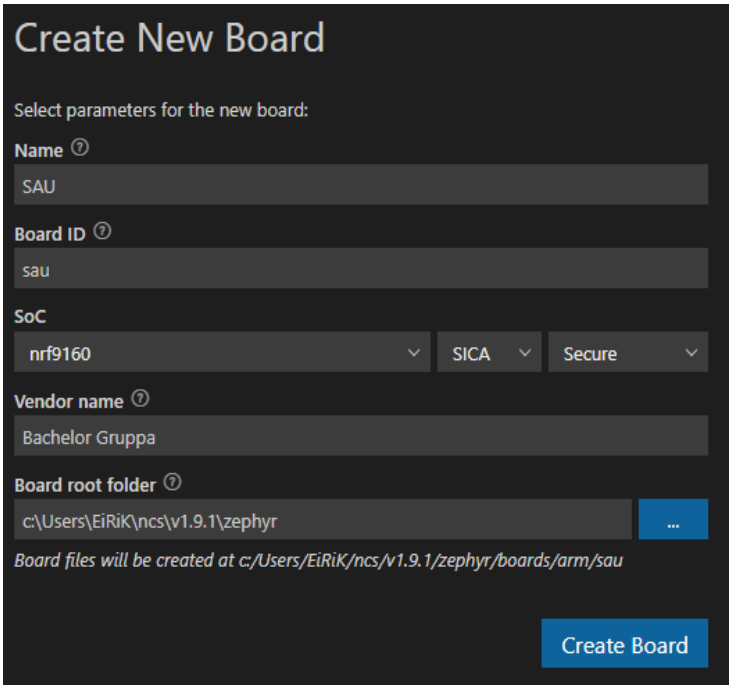


Figure 29: Hvordan legge til egendefinert maskinvare konfigurasjon i nRF SDK

Generer man filer for et egendefinert nRF9160 kort blir minnet automatisk partisjonert til å støtte det interne minnet alene. Siden prototype utlegget ikke bruker ekstern flash kunne dette stå uendret. Andre grensesnitt som knapp, GPIO utganger, UART og SPI må defineres til å passe maskinvare konfigurasjonen.

4.1.11 UART kommunikasjon med nRF9160

For direkte UART kommunikasjon til PC via USB kan man ta i bruk en USB til UART bro. UART må først konfigureres i kortets Device Tree. Videre er det også mulighet for fastvare oppdatering gjennom oppstartslasteren MCUBoot. Det er flere kort på markedet som har en slik funksjonalitet som Spark Fun: Thing Plus. Man konfigurerer UART pinnene i kortets Device Tree slik:
Først innen for treet (/):

```
chosen {
  zephyr,console = &uart0;
  zephyr,shell-uart = &uart0;
  zephyr,uart-mcumgr = &uart0;
};
```

Så legger man ved type pinner og ønsket baud rate:

```
&uart0 {
  status = "okay";
  current-speed = <115200>;
```

```
  tx-pin = <6>;
  rx-pin = <5>;
};
```

4.1.12 SPI kommunikasjon med nRF9160

Akselerometeret kommuniserer med nRF9160 med bruk av SPI protokollen. I kortets maskinvarekonfigurasjon må man definere valgte pinner og sette maks SPI frekvens.

```
&spi1 {
  status = "okay";
  sck-pin = <18>;
  mosi-pin = <16>;
  miso-pin = <15>;
  cs-gpios = <&gpio0 12 GPIO_ACTIVE_LOW>;
  adxl362: spi-dev-adxl362@0 {
    compatible = "adi,adxl362";
    label = "ADXL362";
    spi-max-frequency = <8000000>;
    reg = <0>;
    int1-gpios = <&gpio0 11 0>;
};
```

4.1.13 Definering av knapper i Device Tree

Definisjon av knapper gjøres i hovedtreet. Prototypen har en enkel knapp som har som funksjon og sette enheten i DFU modus. Den er her definert som en GPIO pull up som er aktiv lav.

```
  buttons {
    compatible = "gpio-keys";
    button0: button_0 {
      gpios = <&gpio0 13 (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>;
      label = "DFU";
    };
  };
```

4.1.14 J-link

SEGGER J-link er et verktøy for debugging og oppdatering av fastvare av ulike CPUer og arkitekturer. J-link kommuniserer direkte med målenheten via SPI protokollen. nRF9160 DK har en fast montert J-link OB MCU for programmering/debugging av både nRF9160 og nRF52840. DK har også en SWD samt pinne kontakter for tilkobling av

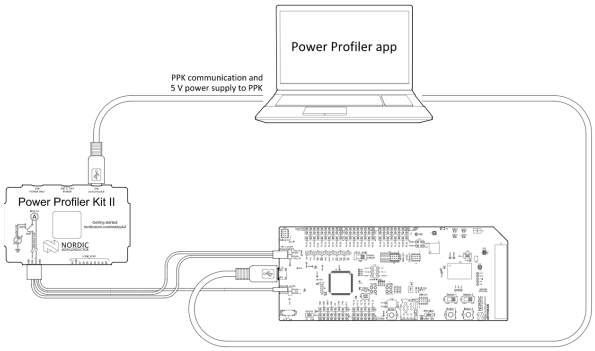
OB J-link til andre kort.

4.2 Strømtrekkanalyse

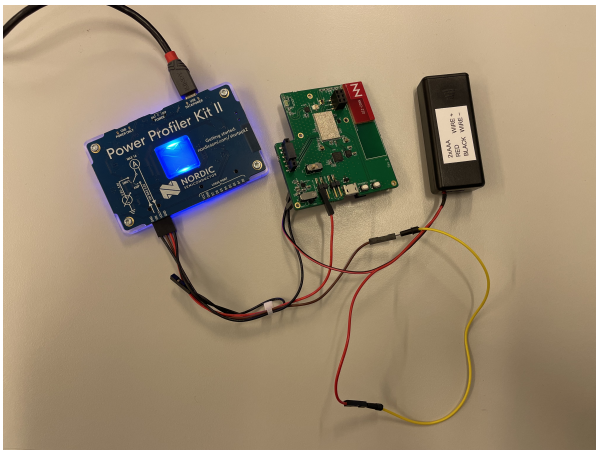
For å gjennomføre strømtrekkanalyser har vi benyttet følgende utstyr og oppsett:

Utstyr
All måledata er innhentet med PPK2 v1.0.1 serienummer: PCA63100*. nRF Connect v3.10.0 med official Power Profiler app v3.4.3. PPK2 brukes i Amperemeter modus ettersom dette vil gi de mest realistiske målingene fra når modulen trekker strøm fra et batteri. Samplingsalternativene tillater maksimalt varighet fra 500 sek med 100 000 samples per sekund til 578 dager med 10 samples per sekund. Datainnsamling er utført med 10 000 samples per sekund.

Oppsett
Målinger er gjort med utstyret koblet som på figur 15. For å unngå støy fra kraftnettet, som PPK2 plukker opp, har vi gjort det uten strømforsyning i den aktuelle laptop.



(a) PPK2 med DK [45]



(b) PPK2 med prototype

Figure 30: Oppsett for strømanalyse

Målinger
Vi ønsker å måle strømbruk på følgende. Vi vurderer også presisjonen på disse der det er aktuelt:
1. Single- vs. multicell celletårnposisjonering
2. Nedlasting av assistansedata for P-GPS vs. A-GPS
3. Ren GPS-posisjonering
4. Vårt eget system med gitte intervaller på utviklingskort og på egen prototype

Grunnet begrensningen hos nettleverandøren iBasis får vi ikke testet LTE-M vs NB-IoT, som vi gjerne skulle gjort.

Alle målinger bortsett fra den på eget system er gjort ved å bruke Modem Shell-samplet. Dette gir oss mulighet til å isolert teste enkelte metoder og teknikker.

4.3 Software

4.3.1 Realisering og konsept

For å realisere vårt system tok vi utgangspunkt i flere av Nordic Semiconductor sine samples for nRF9160, spesielt da GNSS- og Cloud Client-samplet. Vi vil her redegjøre for de valg og metoder som kan anvendes for et slikt program:

LTE: Enheten støtter NB-IoT og LTE-M modus. For større datapakker og raskere trafikk ble LTE-M valg. Dette var en fordel når vi anvender A-GPS, som er av noe størrelse. Uten bruk av dette kan NB-IoT være et mer effektivt alternativ, gitt at tjenesteleverandøren støtter dette.
Dette settes i prosjektet konfigurasjonsfil slik:

```
CONFIG_LTE_NETWORK_MODE_LTE_M_GPS=y
# CONFIG_LTE_NETWORK_MODE_NBIOT_GPS
```

For lange intervaller uten datatrafikk er PSM å foretrekke foran eDRX som strømsparemodus på LTE. Vi setter LTE PSM-modus før vi kobler oss til nettverket, slik at dette etterspørres som en del av nettverksforhandlingen ved tilkobling.

GPS: Prosjektet vårt støtter både ren GPS eller assistert GPS. Dette kan velger i prosjektets kconfig. Dersom man ikke velger assistert GPS kompileres ikke de tråder, workqueues og funksjoner som er tilknyttet dette. Slik sparer vi minne dersom man ikke ønsker denne funksjonaliteten. Slik definerer man valget i kconfig:

```
choice
  default SAU_ASSISTANCE_NONE
  prompt "Select whether GNSS assistance is used or not"
```

```
4
5 config SAU_ASSISTANCE_NONE
6   bool "Assistance not used"
```

Deretter kan man i koden definere hva som skal kompileres eller ikke ut fra gitte konfigurasjoner, eksempelvis slik:

```
1  #if !defined(CONFIG_SAU_ASSISTANCE_NONE) // This only happens if we want to use AGPS
2    struct k_work_queue_config cfg = {
3      .name = "gnss_work_q",
4      .no_yield = false};
5
6    k_work_queue_start(
7      &gnss_work_q,
8      gnss_workq_stack_area,
9      K_THREAD_STACK_SIZEOF(gnss_workq_stack_area),
10     GNSS_WORKQ_THREAD_PRIORITY,
11     &cfg);
12
13   k_work_init(&agps_data_get_work, agps_data_get_work_fn);
14   err = assistance_init(&gnss_work_q);
15 #endif /* !CONFIG_SAU_ASSISTANCE_NONE */
```

I vår anvendelse ser vi ikke at P-GPS er et fornuftig alternativ, da vi stadig har LTE-tilkobling og sender med nokså hyppig intervall.

Kommunikasjon: For å kommunisere med nRF Cloud benytter vi MQTT. For denne skyløsningen er Nordics biblioteker svært gode, og kommunikasjonsprotokoll kan defineres likt som LTE-modus i prosjektets konfigurasjonsfiler. Broker og topics settes opp i bakgrunnen, og vi trenger bare forholde oss til de innebygde funksjoner for å skrive til- og lese fra skyen. MQTT gir oss fordelen med full duplex i sanntid. Alternativt kan man også benytte REST. Dette ville vært ennå mer strømsparende, men vi kunne bare kommunisert til enheten når den selv tok initiativ til det.

Handlers: For å agere på hendelser fra GPS-modul, LTE-modul og fra skyen setter vi opp handlers for dette. Ulike hendelser blir agert på utifra dens natur, eksempelvis vil en "disconnect from cloud"-hendelse føre til at vi forsøker å etablere en ny tilkobling, og en "PVT Fix valid" vil markere at ny GPS plassering er tilgjengelig, så kan en annen tråd til å hente ut og behandle denne.

Tråder: Koden er produsert modulært, med korte funksjoner. Det er gjøres vurderinger på hva som bør være selvstendige tråder og workqueues. For å spare minne velger vi å ha færrest mulig workqueues, og hovedsaklig benytte system queue. Ved bruk av A-GPS krever dette en egen workqueue, og denne blir kun produsert dersom man velger å bruke assistanse før man bygger programmet.

Kommunikasjon mellom tråder Kommunikasjon mellom tråder foregår ved å sette semaphores. Dette ble valgt foran mutexes for å unngå deadlocking og logikk rundt dette. Semaphores settes når man har gyldig GPS-fiks, når man er koblet til LTE og når man har forbindelse med skyen. Dette brukes igjen for å kalle tråder som agerer på gitte hendelser.

4.3.2 Utviklermodus

Utviklermodus har som formål å endre funksjonaliteten til koden slik at den blir enklere å feilsøke på. Når funksjonen er avslått vil enheten kjøre mer strømeffektivt fordi den ikke bruker resurser på skrive UART informasjon over COM portene.

Måten dette ble løst på var at det var en variabel fra cloud_data struct kalt from_cloud.developer som sjekkes i funksjonen printk2. Denne funksjonaliteten kunne vært utvidet til å omfatte flere parametere og kan styres fjernt nRF Cloud.

```
/*like normal printk, but only runs when developer mode is on.*/
void printk2(const char *fmt,...) {
  if (from_cloud.developer == 1){
    printk(fmt);
  }
}
```

4.3.3 Geofence

For å unngå overflødig pakkeutveksling i tilfellet der enheten befinner seg innenfor det ønskede området ble det implementert et Geofence. Produktets løsning er slik at det sjekkes om enheten er innenfor GPS koordinatene som spesifiseres, og i tilfellet der den er utenfor rammene sendes det data til nRF Cloud. Grensekoordinatene kan settes på forhånd i programkoden eller bli tilordnet fra nRF Cloud over LTE.

```
bool in_geofence() {
  if (last_pvt.latitude  < from_cloud.lat_max &&
      last_pvt.latitude  > from_cloud.lat_min &&
      last_pvt.longitude < from_cloud.lon_max &&
      last_pvt.longitude > from_cloud.lon_min)
  {
    return true;
  }
  else {
    return false;
  }
}
```

4.3.4 JSON

Terminalen på nrF Cloud krever at det brukes JSON i dataflyten mellom den og nRF9160. Det ble anvendt et bibliotek kalt cJSON som er innebygd i Zephyr for å lage strenger på JSON format av datasett som sendes til og fra nRF Cloud.

I kommunikasjonen fra nRF9160 til nRF Cloud sendes en forenklet pakke med essensiell GPS informasjon i tillegg til status på utviklermodus. Det ble valgt å utelate overflødig data for å redusere pakkestørrelsene slik at forsendingene bruker mindre strøm.

```

/* Creating JSON Object example */
char *out;
cJSON *root;

// Lag JSON object
root = cJSON_CreateObject();

// Lag JSON item
cJSON_AddItemToObject(root, "Latitude", cJSON_CreateString(lat_val));

out = cJSON_Print(root); //Lagrer JSON objektet som char array

transmit_to_cloud(out); // sender JSON til cloud

//Frigjr minnet
free(out);
cJSON_free(out);

// Frigjr minnet til objektet.
cJSON_Delete(root);
```

```

/* Parsing JSON object from cloud example */
const cJSON *var = NULL;

cJSON *json = cJSON_Parse(json_from_cloud); //Lager og analyserer objekt fra Cloud

var = cJSON_GetObjectItemCaseSensitive(json, "Developer"); //Henter gjenstand
if (cJSON_IsString(var)) { //Sjekker om dataen er riktig formatert
    from_cloud.developer = atoi(var->valuestring); Skriver verdien til variabel.
}

cJSON_Delete(json);
```

Videre ble det tilrettelagt for å kunne sende data fra nRF Cloud til nRF9160 slik at parametere kunne endres fjernt. Utviklermodus kan endres slik at det blir mulig å feilsøke, man kan sette ønsket tidsintervall for hvor ofte GPS skal prøve å få fix og hvor lenge den skal vente om den ikke er i stand for å få en fix. Det siste objektet inneholder informasjon om de ytre grensene til Geofence.

```
{ /* Pakke fra nRF9160 til nRF Cloud */
  "Latitude": "63.418514",
  "Longitude":"10.403880",
  "Altitude": "45,445549",
  "Accuracy": "37,154873",
  "Developer":"1"
}
```

```
{
/* Pakke fra nRF Cloud til nRF9160 */
  "Developer":   "1",
  "fix_retry":   "1400",
  "fix_interval":"120"
  "gps_fence":{
    "lat_max": "63.4",
    "lat_min": "63.5",
    "lon_max": "10.5"
    "lon_min": "10.4"}
}
```

4.3.5 Oppdatering av programvare med MCUMGR

Enheter som støtter operativsystemet Zephyr og oppstartslasteren MCUBoot kan konfigureres til å kunne oppdatere programvare via seriell kommunikasjon og over luften via UDP eller Bluetooth. Dette krever et eksternt verktøy som heter MCUMGR som ikke kommer inkludert i nRF toolchain. For å installere dette verktøyet nå man først installere programmeringspråket GO.

```
go get github.com/apache/mynewt-mcumgr-cli/mcumgr
```

For at oppstartslasteren skal inkluderes under opplasting i nRF SDK må man legge til en konfigurasjon i prosjektets prj.conf.

```
CONFIG_BOOTLOADER_MCUBOOT=y
```

SDK opplaster først oppstartlaster og så applikasjonen om man bygger under sikker brett konfigurasjon. Om man skal gjøre dette utenom SDK må dette gjøres separat.
For å slippe å måtte oppgi COM port og ønsket baudrate kan man legge til profiler i MCUMGR.

```
mcumgr conn add acm0 type="serial" connstring="dev=/dev/ttyACM0,baud=115200,mtu=512"
```

Når man bygger et sikkert bilde med MCUBoot konfigurert i applikasjonens prj.conf blir det automatisk generert et image under bygging som er å finnes i build mappen:

```
C:\Users\#\applikasjon_navn\build\zephyr\app_uppdate.bin
```

For at MCUBoot skal godta programvare opplasting må det genererte image valideres. Som vil si at MCUBoot sjekker bildets signerte nøkkel opp mot sin egen. Om bruker ikke har definert noen nøkkel i prosjektets konfigurasjon får bildet og MCUBoot en standard nøkkel som utgjør en sikkerhetsrisiko om funksjonaliteten brukes utenfor testing. For at enheten skal gjøres mottagelig for kommando gjennom MCUMGR verktøyet må det settes i DFU modus (Device Firmware Update). Dette kan gjøres enten i software eller fysisk. For fysisk setting av DFU kan man definere en pinne som må settes lav eller høy i MCUBoot brett konfigurasjon som er lokalisert her:

```
C:\Users\#\ncs\v1.9.1\bootloader\mcuboot\boot\zephyr\boards
```

I brettets konfigurasjon kan man definere ønsket GPIO pinne. I prototypen er dette pinne 13 og er knyttet til en knapp som trigger DFU modus om lav.

```
# MCUboot serial recovery
CONFIG_MCUBOOT_SERIAL=y
CONFIG_BOOT_SERIAL_DETECT_PORT="GPIO_0"
CONFIG_BOOT_SERIAL_DETECT_PIN=13
CONFIG_BOOT_SERIAL_DETECT_PIN_VAL=0
```

For at DFU modus skal slås på i en slik konfigurasjon må DFU knappen holdes nede mens enheten slås på. I DFU modus kan man da laste opp et signert bilde ved hjelp av MCUMGR:

```
mcumgr -c acm0 image upload -e build/zephyr/app_update.bin
```

5 RESULTATER

5.1 Hardware

5.1.1 Prototype

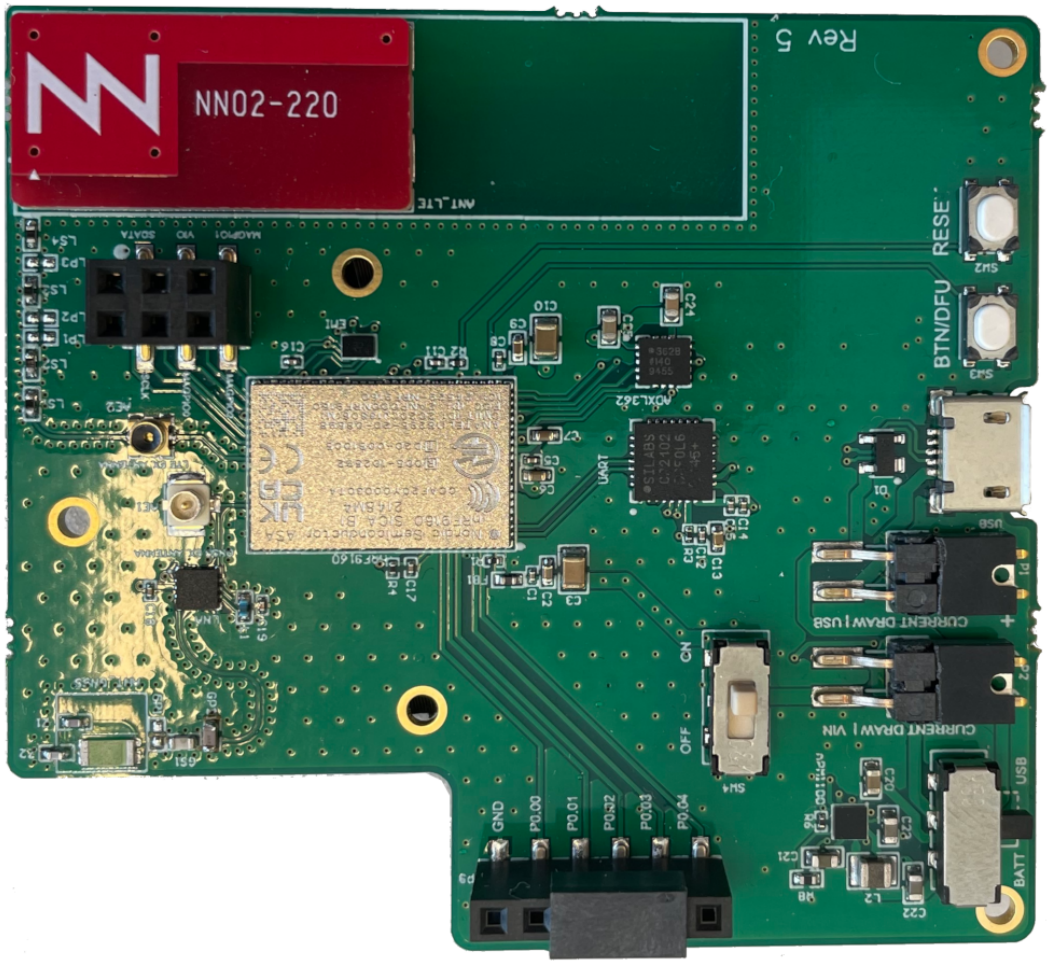


Figure 31: Bilde av prototype utlegg

PCB ble bestilt og montert av Nordic Semiconductor. Etter testing fungerte det meste som forventet, men kom også med noen mangler. Prototypens GPS system fungerte ikke fordi den ikke var tilrettelagt aktiv antenne i tillegg til utfordringer med å få impedans-matchet antenne nettverket.

Oppkoblingen til Cloud ble ikke realisert fordi det ikke var mulig å skrive AT commands til prototype utlegget slik at IMEI kunne oppdrives for å provisjonere nRF9160 SiP.

Programmeringen av prototypen fungerte som tiltenkt ved å flashe via SWD fra nRF9160-DK.

UART kommunikasjonen fungerte som tiltenkt. Vi var i stand til å lese av UART over USB til PC og laste opp programvare.

Perifert grensesnitt som brytere og GPIO fungerer som tiltenkt.

Målepinnene for testing av strømtrekk fungerer også.

LTE-oppkoblingen fungerte godt og vi fikk koblet oss til operatør. Nettverksanalysen fra nordic i figur 32 av LTE-antenne nettverket tilsier at egenskapene er innenfor nordics egne anbefalinger og dette er gjenspeilet i resultatet.

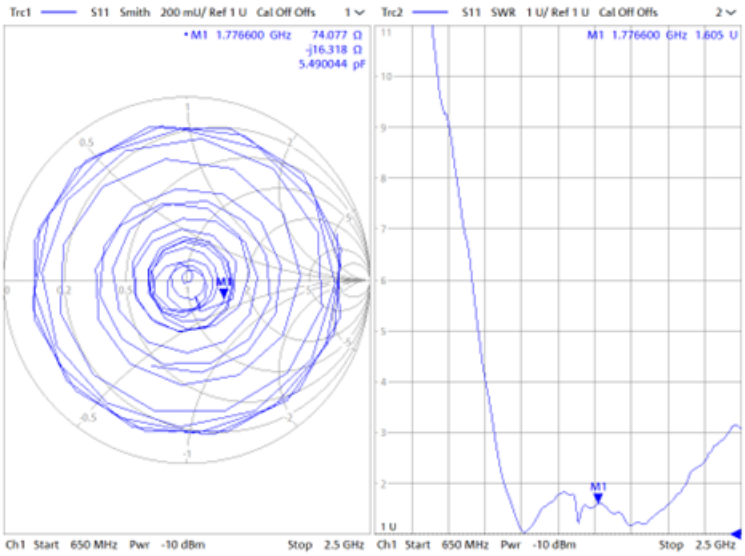
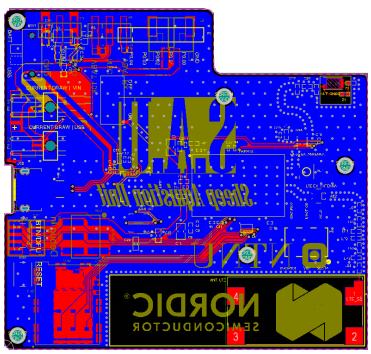


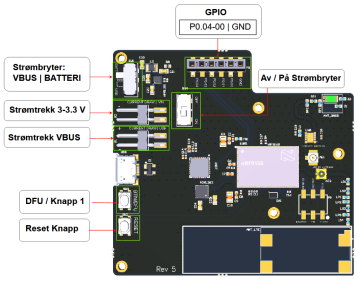
Figure 32: Nettverksanalyse av LTE-antenne nettverk

5.1.2 PCB Design

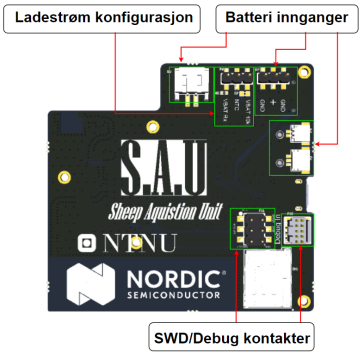
PCB design



(a) PCB design



(b) Forside av prototype utlegg



(c) Bakside av prototype utlegg

5.2 Strømtrekkanalyse

Innhenting av data er gjort med nRF9160 DK flashet med modem_shell sample-kode. Dette er gjort hovedsakelig av to grunner: Først og fremst for å bli kjent med de forskjellige funksjonene til nRF9160 SiP og modemet i utviklingsfasen. Den andre grunnen er for ha lett ha kontroll over funksjonene og modeminnstillinger under innhenting av data. Dataene er innhentet for å sammenligne med Nordics egen data om strømtrekk for å verifisere at vi opererer DK i riktig modus og samtidig danne et eget grunnlag for design av eget produkt. Modem_shell gir mulighet for kommunikasjon med modemet gjennom UART. Kommandoene forkortes til MoSh-kommandoer videre.

5.2.1 celletårn plassering

Under følger en analyse av single- og multicell posisjonering. Vi analyserer strømtrekk og treffsikkerhet med disse metodene. Disse analysene er gjort på utviklingsbrettet.

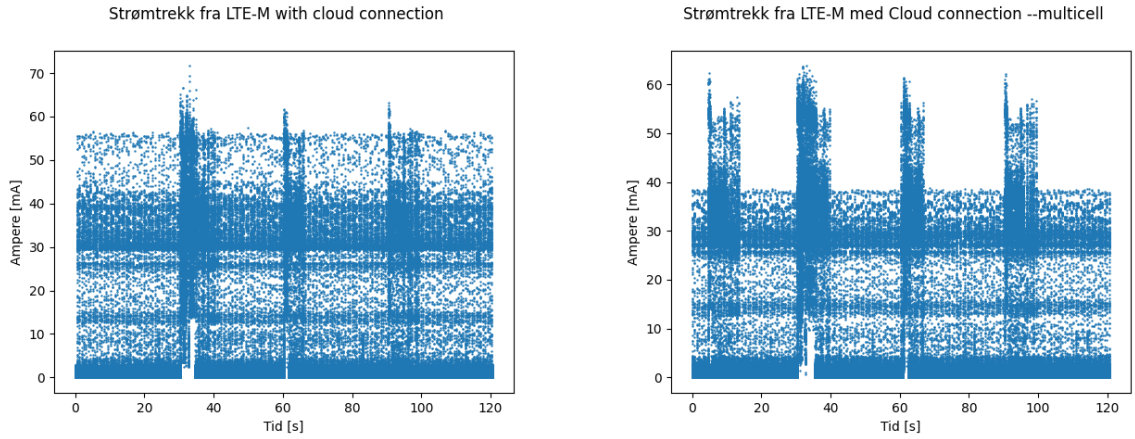


Figure 34: Strømtrekk fra single cell vs multi cell

Figure 35: Presisjon, single vs multicell

MoSh kommando	Tid
cloud connect	30 sekunder
location get −−method cellular	60 sekunder
cloud disconnect	90 sekunder

Table 6: Tidsstempel for kommandoer, single cell

Fix fra modem	63.132496N , 9.07408E
Faktisk lokasjon	63.12314N , 9.710238E
Avstand	31.99km
Treffsikkerhet	973.0m

Table 7: Resultater, single cell

Fra figur 34 ser vi strømtrekket korresponderer med tidspunktene som kommandoene ble sendt ut, som er forklart i tabellene under

Som vi kan se fra resultatene, er multicell en svært strømeffektiv metode for å skaffe plassering, men plasseringen er alt for grov for vår anvendelse. På bakgrunn av dette har vi valgt bort celletårn plassering i vårt prosjekt.
Vi ser også at single cell ga oss en høyere treffsikkerhet enn multicell, selv om multicell-målingene ble gjort i et tettbygd byområde hvor antenne tetthet antas være større enn det mer landlige området hvor single cell ble testet. Multicell var svært presis[46] på tross av sin dårlige treffsikkerhet, noe vi kan anta er en heldig tilfeldighet. Vi velger å ikke utforske metodene rundt celletårn plassering videre.

5.2.2 GNSS

Følgende er en analyse av ren GPS-plassering, med strømtrekk og treffsikkerhet. Disse analysene er gjort på utviklingsbrettet.

NMEA og Fix-output, med PSM aktivert og ekstern antenne. Start periodiske fix med 2 minutters intervall og 90 sekunders tidsavbrudd.

Basert på dette, anser vi ren GPS som en nokså effektiv og presis metode for posisjonering. Dette forutsetter naturligvis gode dekningsforhold for satellittmottak.

MoSh kommando	Tid
cloud connect	30 sekunder
location get –methode cellular –interval 0	60 sekunder
cloud disconnect	90 sekunder

Table 8: Tidsstempel for kommandoer, multicell

Fix fra modem	63.418N , 10.4055E
Faktisk lokasjon	63.41812N , 10.40533E
Avstand	15.8m
Treffsikkerhet	1432.3m

Table 9: Resultater, multicell

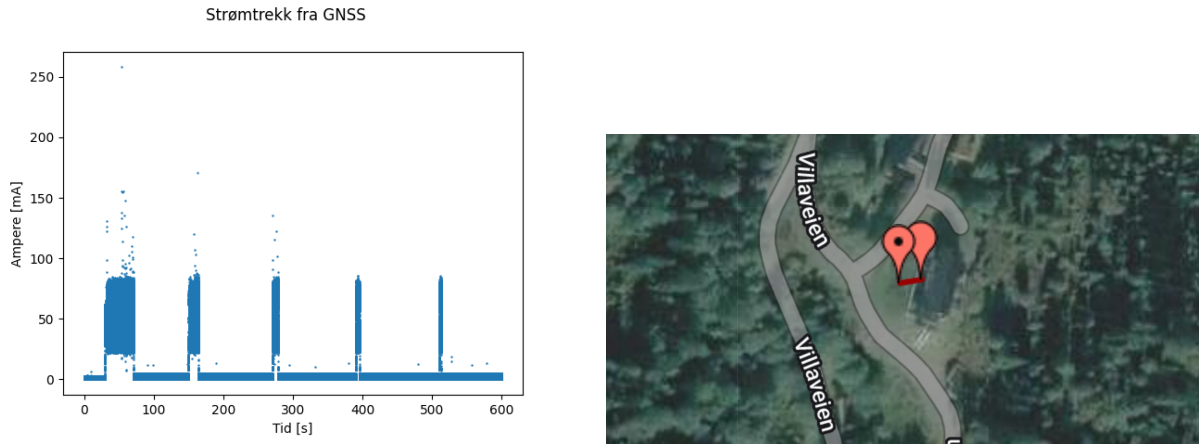


Figure 36: Strømtrekk og presisjon med ren GPS

MoSh kommando	**Tid**
link psm -e	--
gnss output 1 1 0	--
gnss mode periodic 120 90	--
gnss start	60 sekunder

Table 10: Tidsstempel for kommandoer, GNSS

Fix fra modem	63.123131N , 9.710111E
Faktisk lokasjon	63.12314N , 9.710238E
Avstand	6.5m
Treffsikkerhet	64.5m

Table 11: Resultater, GNSS

5.2.3 Nedlasting av P-GPS-data

Vi ønsket her å sammenligne strømtrekk ved nedlasting av P-GPS og A-GPS-data, men grunnet tidvis manglende A-GPS støtte fra nRF Cloud måtte A-GPS-biten utgå. Derfor presenterer vi her en strømtrekkanalyse ved nedlasting av P-GPS-data. Disse analysene er gjort på utviklingsbrettet.

Vi bemerker oss spesielt at nedlasting av P-GPS-data tok opp mot 10 minutter, og brukte en betydelig mengde data. Sammenlignet med å hente en ren GPS-fix, hvor

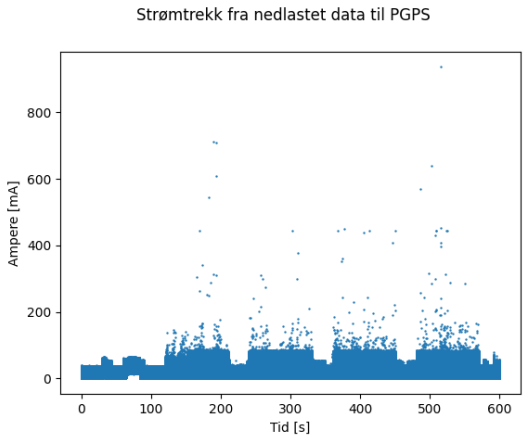


Figure 37: Strømtrekk ved nedlasting av P-GPS data

MoSh kommando	**Tid**
gnss output 1 1 1	--
gnss pgps enable	30 sekunder

Table 12: Tidsstempel for kommandoer, nedlasting av P-GPS data

nødvendig data kommer fra satellittene på ca samme tiden, ser vi det ikke som en stor fordel å benytte P-GPS dersom man har tilnærmer klar sikt der man befinner seg. I bykjerner og områder med dårlig sikt til satellittene kan det være en fordel å ha lastet ned dataen, men vi velger å ikke basere vår løsning på P-GPS.

5.2.4 Vårt eget program på DK og prototype

I denne analysen ønsket vi å sammenligne strømtrekket med vårt eget program på DK og på vår egen prototype. Vår prototype fikk ikke fix. Vi fikk ingen fix på prototypen fordi GPS antennen ikke var riktig matchet. Et annet problem som ble synlig under strømålingene av prototypen var at den ikke klarte å koble seg til cloud. Dette skyldes at nRF9160 SiP ikke var provisjonert for cloudtilkobling på tiden av målingene.
Det følgende er dermed en analyse av vårt eget program kun på DK.

Programmet ble satt til å koble seg til cloud, hente periodisk GPS-fix hvert femte minutt for så å sende fix til cloud. Målingen er gjort med UART tilknyting for å kunne sammenligne strømtrekk med hva modemet gjorde. Det endelige strømtrekket vil være litt lavere dersom vi ikke printet informasjon. Fra figur 38 ser vi at koden på DK bruker 69 sekunder på TFF. Sender fix til cloud og venter intervallet til neste fix. Koden gjør en ekstra fix rundt 5 minutter etter at modemet slås på men fortsetter ikke å gjøre dette etterpå.
Koden på prototypen tracker satellitter å prøver og koble seg til cloud hvert 30. sekund. Strømmålingen fremhevet en svakhet i koden der modulen ender opp med å bruke mer

strøm fordi den ikke får koblet seg sammen med cloud.

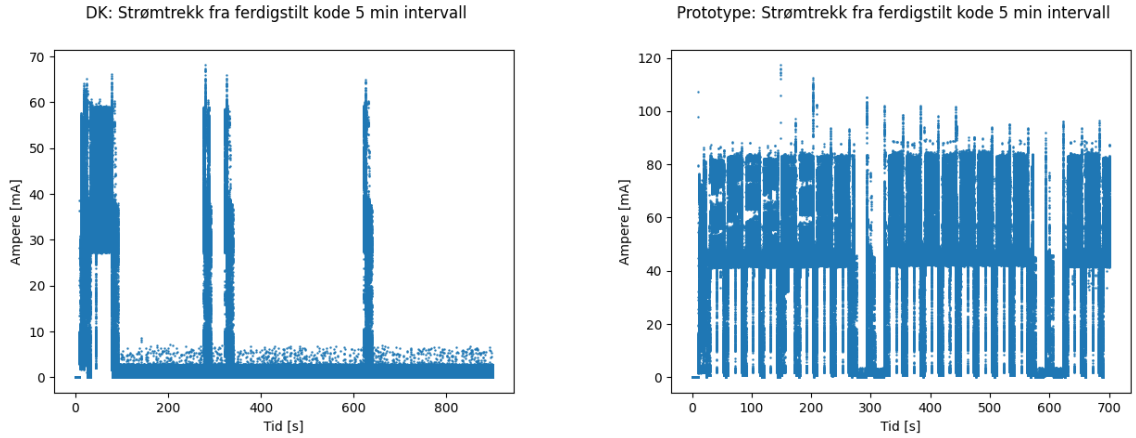


Figure 38: Strømtrekk fra DK og Prototype

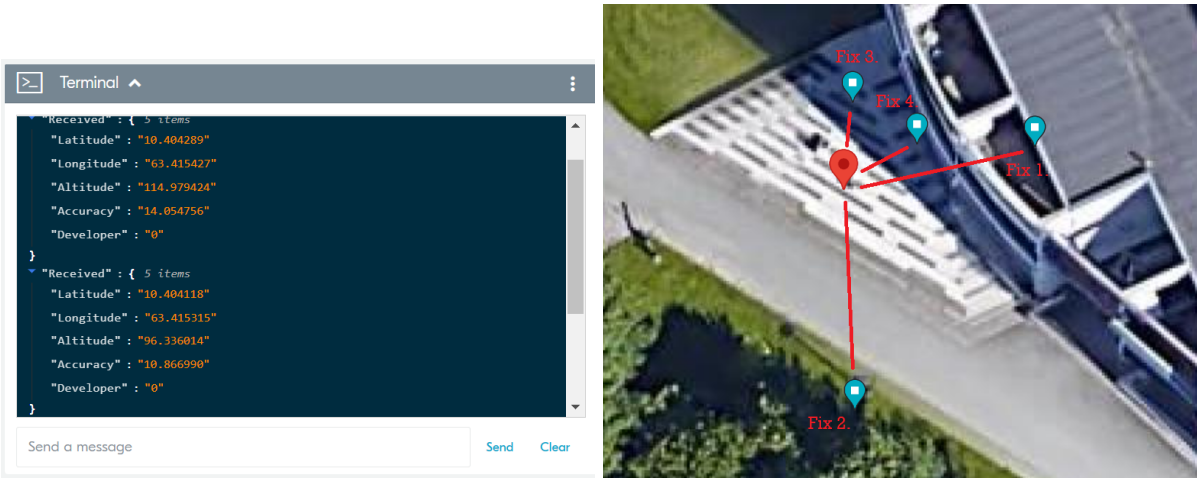


Figure 39: Data mottatt i Cloud og GPS fix fra DK

Det er flere ting å legge merke til i analysen av strømdata. Selv om prototypen har en betydelig lavere oppstartstrøm så er peak startup stor i begge tilfeller. Dette begrenser muligheten for batterier med lav mAh. Grunnet til dette skyles som nevnt i teoridelen at batterier med lav kapasitet kan permanent skades av store peaks. En annen viktig data fra analysen er at Cloud sending i gjennomsnitt bruker mindre strøm enn GPS posisjonering. Med dette i mente er det lett å forstå at A-GPS kunne hjulpet med å optimalisere trackeren. Både med å få hurtigere fixer og redusere strømtrekket.
Som vi ser av analysen, fungerer systemet godt med svært strømsparende hviletid mellom innhenting av fix. Når fix innhentes gikk dette i løpet av få sekunder med intervall på fem minutter. Det legges spesielt merke til avvik fra de ønskede intervallene på fem minutter i starten av programmet. Denne prekære oppførselen fra modemet meldes tilbake til Nordics utviklere.

*Peak startup er filtrert vekk for at grafen lettere skal vise data. Ufiltrert data ligger som

Hendelse DK	Gjennomsnittlig strømtrekk	Enhet
Peak startup*	741.979	mA
TFF**	27.747	mA
PSM	0.488	mA
GPS-fix	37.291	mA
Cloud-sending	5.241	mA
Total	3.777	mA
Hendelse PT		
Peak startup*	245.227	mA
TFF	–	–
Hvilestrøm	766.426	µA
GPS-fix	–	–
Cloud-sending	–	–
Total	36.088	mA

Table 13: Strømtrekkanalyse, eget system på DK

vedlegg. **TFF inkluderer oppstartsstrøm (som har en stor peak), oppkobling til LTE og nRF Cloud, og sending av første fix til cloud.

	Lokasjon	Avstand	Treffsikkerhet
Fix 1	63.415427N, 10.404289E	9.4m	14.1m
Fix 2	63.415315N, 10.404118E	10.7m	10.9m
Fix 3	63.415446N, 10.404116E	3.9m	13.1m
Fix 4	63.415428N, 10.404177E	3.6m	15.8m
Faktisk posisjon	63.415411N, 10.404104E	0	∞

Table 14: DK: GPS posisjonering fra ferdig kode

5.2.5 Beregning av batterilevetid

For å beregne levetid til batteriet gjør vi som nevnt i teoridelen en omforming fra mA til mAh for så å beregne hvor mye energi de forskjellige modusene trekker i et gitt tidsrom. Strømtrekket beregnet når modulen er i PSM er 0.488mA over 200 sekunder. For disse 200 sekundene blir:

$$0.488mA \cdot 200s/3600s/h = 0.0271mAh.$$

Denne omformingen gjøres på PSM, GPS fix og send til cloud moduser i koden. Strømtrekket i PSM modus er stabilt og kan dermed forventes at trackeren å holde samme strømtrekk ved forskjellige tidsintervaller av fixer.

PSM	**Enhet**	**Varighet**	**Enhet**
0.0271	mAh	200	sekunder
0.0408	mAh	300	sekunder
0.0816	mAh	10	minutter
0.4896	mAh	60	minutter
3.9168	mAh	8	timer
GPS Fix			
0.0425	mAh	4.1	sekunder
Send til Cloud			
0.0243	mAh	16.7	sekunder

For et batteri av type CR2032 med kapasitet på 235mAh [47] får vi følgende analyser:

5 min intervall mellom posisjonering:

$$\frac{235mAh}{(0.0408+0.0425+0.0243)mAh} \cdot (300 + 16.7 + 4.1)s = 700631s$$

$$700631/60/60/24 = 8.1091dager$$

60 min intervall mellom posisjonering:

$$\frac{235mAh}{(0.4896+0.0425+0.0243)mAh} \cdot (3600 + 16.7 + 4.1)s = 1529273.9s$$

$$1529273.9/60/60/24 = 17.69dager$$

8 timers intervall mellom posisjonering:

$$\frac{235mAh}{(3.9168+0.0425+0.0243)mAh} \cdot (28800 + 16.7 + 4.1)s = 1529273.9s$$

$$1700192.7/60/60/24 = 19.68dager$$

8 timers intervall mellom posisjonering med et 10000mAh batteri:

$$\frac{10000mAh}{(3.9168+0.0425+0.0243)mAh} \cdot (28800 + 16.7 + 4.1)s = 1529273.9s$$

$$72348629.38/60/60/24 = 837.36dager$$

5.3 Software

5.3.1 Beskrivelse av program

Programmet vårt består først av rutiner for å sette opp kommunikasjon med LTE-nettverket, iverksette GPS-modul med ønsket anvendelse og evt etterspørre assistanse data dersom man ønsker det.
Vi benytter periodisk etterspørsel av plasseringsdata, med variabel intervall som kan endres fra skyen. Når man får en gyldig fix, vil denne sendes opp til skyen så fremst man har kontakt med den.
Vi velger å være tilkoblet LTE kontinuerlig i PSM-modus. Da vi alltid skal sende data innen 68 timer, vil dette være strømsparende fremfor å reforhandle med nettverket.
Vi benytter MQTT til kommunikasjon mellom enheten vår og skyen. Dette gir oss full duplex i sanntid, og muliggjør utvidelser som å etterspørre posisjon eller endre intervaller uten venting, slik det ville vært om vi brukte REST.
Vi benytter GNSS-handler og Cloud-handler for å sette flagg via semaphores når gyldig PVT-data er tilgjengelig, og når vi er tilkoblet skyen.

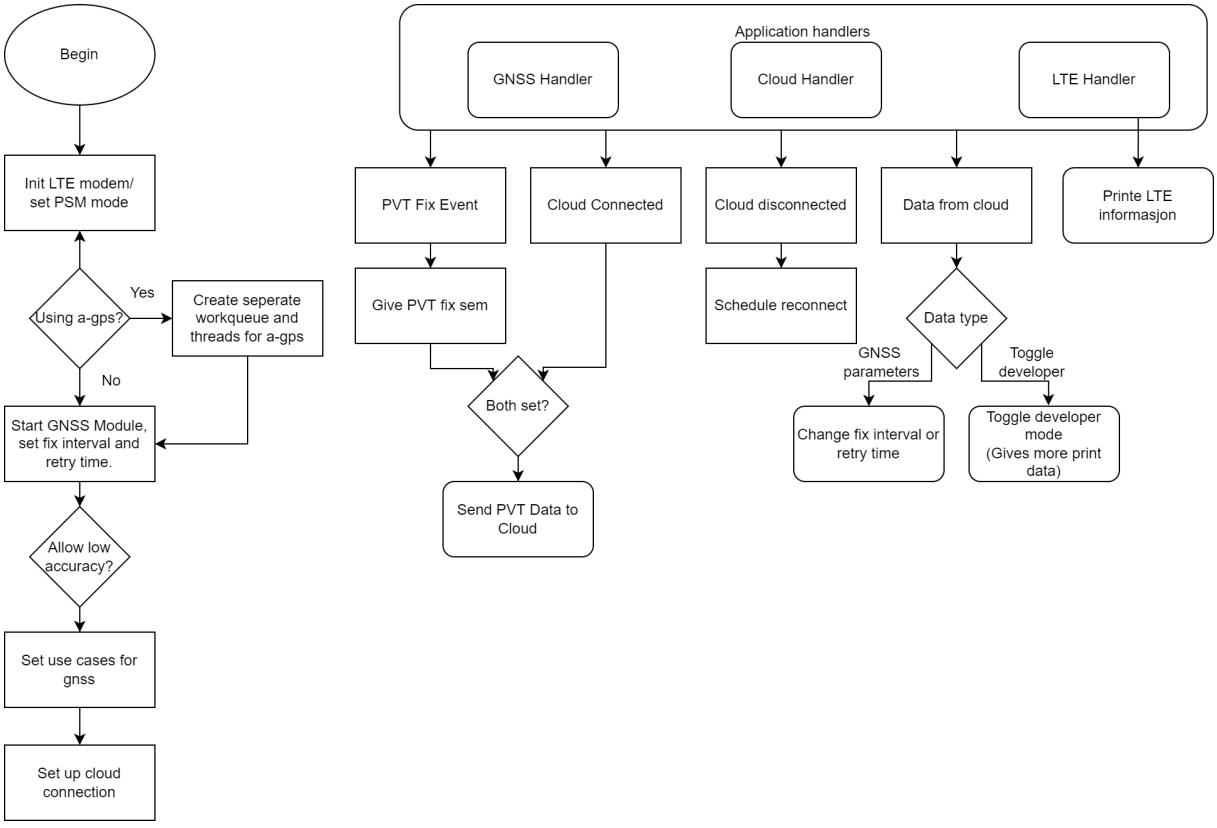


Figure 40: Flytskjema av program

5.3.2 Konfigurering av programvare

Avhengig av applikasjonen har vi muliggjort for ulike måter å innhente GPS, samt ulike innstillinger for LTE-kontakt. Disse endringene gjøres i programmets kconfig, og påvirker dermed hva som kompileres når man bygger programmet.
Man kan velge hvilket antennesett man vil ha, da produktet vårt har både innebygde antenner og konnektor for å koble til eksterne. Hva man velger her avgjør hvilke konfigureringer man sender til LNA og LTE-enhet via MAGPIO og COEX0.
Man kan også velge om man vil benytte seg av A-GPS, eller bruke ren GPS. Dersom man velger A-GPS blir det satt opp workqueues og tråder tilknyttet dette som agerer når modemet ønsker å etterspørre A-GPS data.
Man har også muligheten til å slå LTE helt av når den ikke er i bruk, fremfor å alltid være tilkoblet med PSM.
Disse valgene kan vil påvirke strømforbruket, og hva som er beste valget vil være applikasjons avhengig. For vårt tilfeller velger vi å bruke kun GPS og være tilkoblet PSM hele tiden. Dette forsvares ved at vi antar gode GPS-forhold ute på beitemarkene og lite bevegelse, og dermed spare datatrafikken for A-GPS, og at vi skal sende GPS-data så ofte at PSM er å foretrekke.

5.3.3 Assistert GPS

nRF Cloud leverer A-GPS data når det etterspørres. Dette har vært svært ustabilt, og mer manglende enn tilgjengelig. Vi har ikke fått A-GPS data når vi har etterspurt det, og utviklingen- og testing av denne funksjonaliteten ble rammet som en konsekvens av dette. Det har fungert som det er implementert i koden nå, men om det kan forbedres/endres har vi ikke hatt mulighet til å teste.

5.3.4 Kommunikasjon

Flytskjema i figur 41 beskriver all kommunikasjon i systemet. Merk at nRF9160 Modem er innebygd i samme SiP som mikrokontrolleren, men er logisk sett adskilt. Vi vil også bemerke at GPS-satellittene ikke sender ut NMEA-strenger, men de blir satt sammen av data fra satellittene.
Kommunikasjon fra nRF9160 til skyen går via MQTT, hvor JSON-strenger med plasseringsdata sendes. Andre veien kan det forespørres endring i tidsintervall for innhenting av data, og man kan slå utviklermodus av-og-på, som bestemmer om info printes til konsoll eller ikke.
nRF Cloud har et grafisk brukergrensesnitt via nettsiden nRFcloud.com. Dette kan tilpasses i svært liten grad, og består av et konsollvindu som viser mottatt data og som man kan skrive inn data som sendes til enheten.
Under utvikling kommuniserer man med produktet vårt fra en PC via UART. Dette er

kun ment for feilsøking og utvikling.

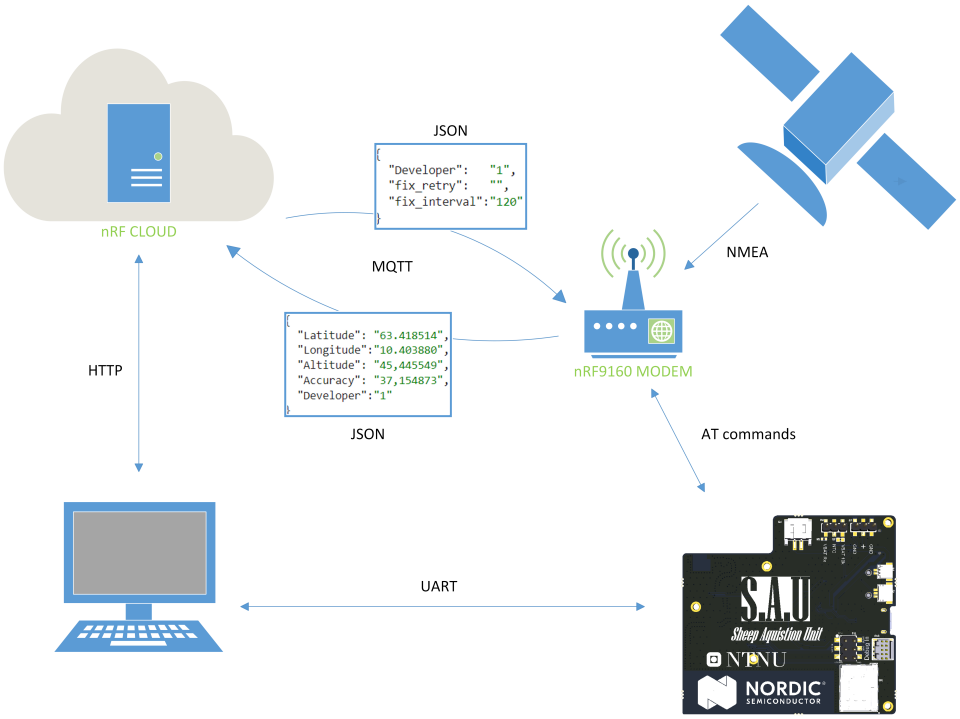


Figure 41: Flytskjema av kommunikasjon i systemet

6 DRØFTING

6.1 Hardware

6.1.1 nPM1100 og nRF9160 strømtopper

nRF9160 har store strømtopper under normal operasjon som gjør at den ikke kan drives av nPM1100s buck converter som er begrenset til 150 mA. Disse strømtoppene skjer under korte tidsrom og kan være vanskelig å måle. Nordic Semiconductor har Power Profiler [48] på nett for strømanalyse av LTE og GPS konfigurasjon. Disse tallene kan være villedende siden de er gjennomsnittstrømmer og viser derfor ikke topper. Under målinger med Power Profiler under oppstart oppsto det en strømtopp som kunne være så høy som 1.2 A. Det finnes dokumentasjon om strømtopper ved bruk av modemet [49], men det ble ikke funnet dokumentasjon om strømtopper under oppstart og annen drift. For å bruke nPM1100 med nRF9160 må man da gå utenom buck converteren og forsyne kretsen direkte via batteriet. Dette gjøres ved å knytte nRF9160s VDD pinner til VSYS utgangen på nPM1100. Man må ellers finne alternativer hvis man ønsker å forsyne nRF9160 gjennom en buck converter.

6.1.2 Akselerometer

Prototypen har et akselerometer som ikke ble anvendt etter vi mottok utlegget. Implementasjon av denne i koden ble ikke prioritert på grunn av mangel på tid og relevans til prosjektet. Akselerometeret var tiltenkt å brukes til å vekke nRF9160 eller spare strømforbruk ved lav fysisk aktivitet og derav ingen posisjonsendring eller varsle om mistanke om sykdom ved uvanlig lite aktivitet i lengre tidsrom.

6.1.3 Antenneforbedringer på prototypen

Det viste seg å være vanskelig å måle impedansen på transmisjonslinjene på prototypen. En ettertanke er å lage et spesifikt utlegg for å optimalisere antennenettverkene eller ha et testpunkt nært modemet som bryter tilkoblingen til modemet slik i 42. Vi burde hatt et tilsvarende design for LTE antennenetverket bare uten LNA men prinsipielt samme som i figur 42. I tillegg er det viktig å ha mulighet til å holde LNA aktiv under testing, dette krever at EN (enable) inngangen kan overstyres høy.

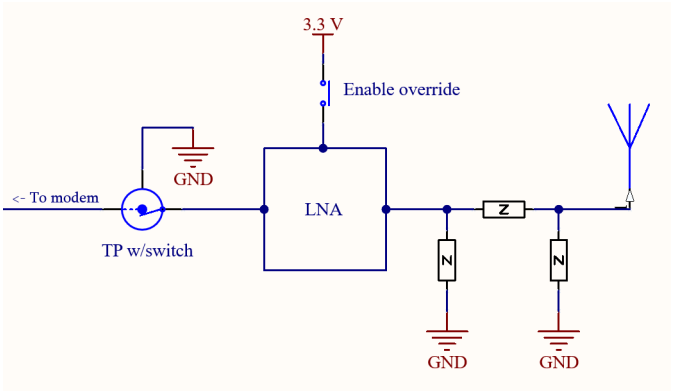


Figure 42: Prinsippskjema for forbedret antenne nettverk

6.1.4 Aktiv ekstern antenne uten tilførsel

Den eksterne antenne utgangen på prototypen har mangler tilførsel til den eksterne antennen. I utviklingskort til nRF9160 har de satt DC-filter før og etter utgangen til ekstern antenne og matet den med 3.3V slik antennen får forsyning sammen med signalet slik som vist i figur 43.

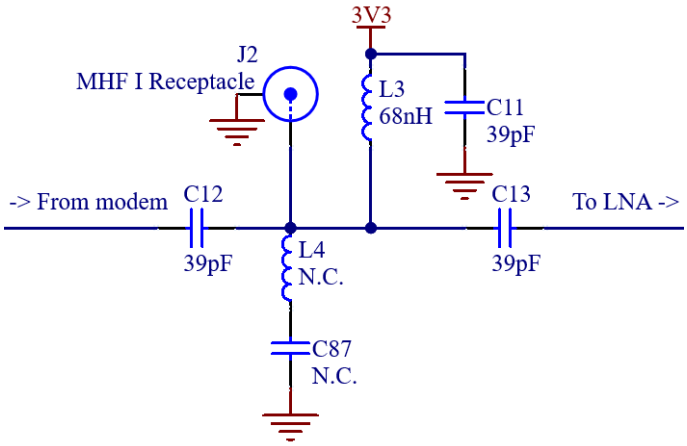


Figure 43: Krets som forsyner aktiv antenne

6.1.5 Kritikkverdig utvikling av antennenettverk

Da vi ønsket å gjøre tester med full funksjonalitet på det prototype utlegget som ble produsert men ikke hadde mulighet til å bruke en ekstern antenne for GPS. Det ble forsøkt av Nordic å gjøre en impedansmatch, men resultatet var ikke godt nokk til å kunne til å få kontakt med satellittene. Vi har heller ikke mulighet til å bruke en ekstern GPS-antenne som henvist til i forrige avsnitt. Vi valgte derfor å anta at den første målingen de utførte på Nordic var en god måling og bruke et verktøy på nettet for å finne verdier på gode estimat på impendanskomponenter.

I figur 44 ser vi at GPS-antennenettverket er godt justert for 1.688GHz, men ønsket bruksfrekvens for GPS (L1-båndet) ligger på 1575.42MHz. I L1-båndet har antennenettverket uten impedanskorrigering en impedans på 11.310Ω + j16.208Ω framstilt Kartesisk. Dette gir en VSWR på 12.385 som gir betydelig refleksjon og effekttap.

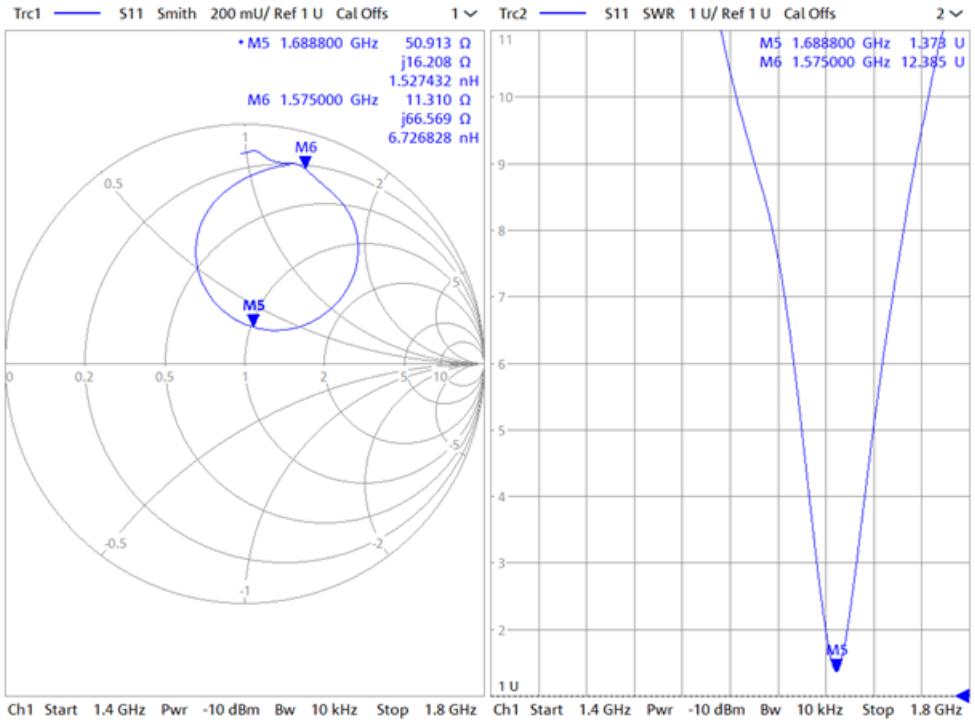


Figure 44: Spektralmåling med kun 0Ω motstand i serie

I figur 45 ender de på en verdi dårligere verdi med impedans på 1.640Ω + j8.375Ω og en VSWR på 31.652 etter å ha forsøkt å bruke verdiene fra evalueringsbrettet som er oppgitt i databladet til GPS-antennen. Antennenettverket her er mye mindre induktivt men er langt unna ønsket reell ohmsk verdi.

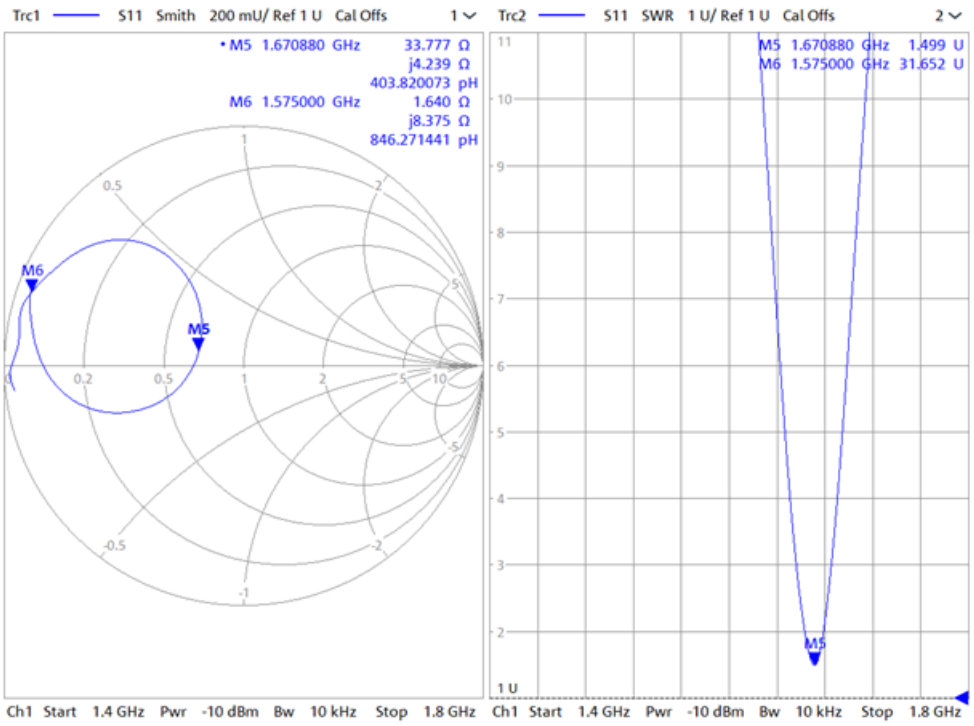


Figure 45: Spektralmåling med 2.7pF i serie og 0.8pF parallelt

6.1.6 Feil ved klareringsområdet til GNSS antennen

GPS antennen har kobber i klareringsområdet under jordpinnen. Dette er en designfeil som kan ha hatt del i å skape utfordringer med impedansmatchingen. Se figur 45.

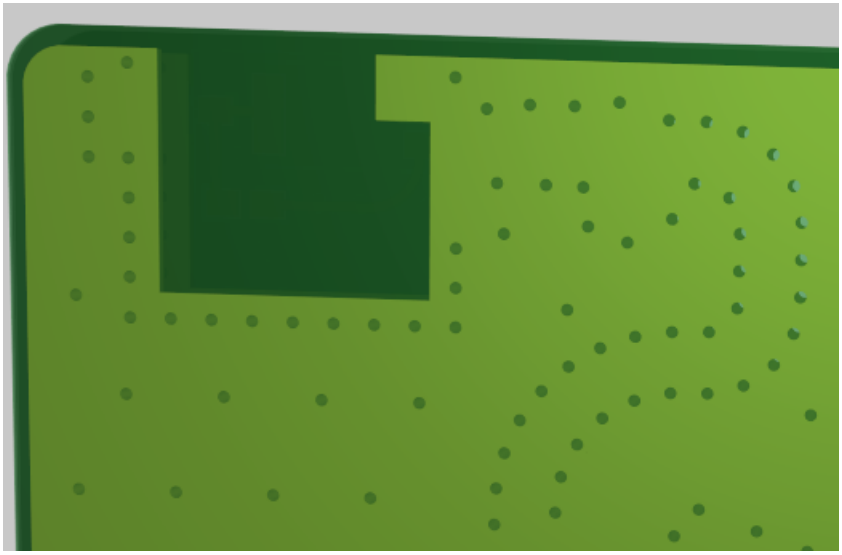


Figure 46: GNSS antennens klareringsområdet har ekstra kobber under jord pinnen

6.1.7 GNSS transmisjonslinjen

GNSS antennens transmisjonslinje har en unødvendig bue. Den ble designet slik for å følge evalueringsbrettet, men legger til ekstra lengde på linja som fører til større I^2R tap.

6.1.8 Prototypeutlegg ankom sent

Prosjektet ble preget av at prototypeutlegget kom en uke før innleveringsdato. Dette gjorde at GPS og Cloud funksjonaliteten ikke rakk å bli testet. Det var ikke nok tid til å få testet og registrert kortet slik at det kunne etableres en tilkobling til Cloud. Det ville trolig og vært tid til å få impedansmatchet antenne nettverket til GPS og fått den til å fungere. Dette ville krevd omfattende modifikasjoner som det ikke var tid til.

6.2 Strømanalyse

6.2.1 Generelle kommentarer

På bakgrunn av strømanalyser med PPK2 har vi gjort designvalg for vårt system. Disse analysene har vært gjort på ulike stedet til ulike tider, noe som kan ha påvirket sluttresultatene. Vi synes fortsatt datagrunnlaget er godt nok til å forsvare valgene vi gjorde. Vi synes det ergerlig på at manglende A-GPS-støtte fra nRF Cloud har begrenset vår mulighet til å ta dette i bruk.
Vi ønsket også å utforske de ulike strømsparingsmodusene i nettverket, med PSM kontra eDRX kontra å koble av nettet. Tidsbegrensninger hindret oss i å gjennomføre disse analysene, men dokumentasjonen vi har funnet hos Nordic og råd fra veiledere tilsier at PSM er det beste alternativet her.
Vi står fast ved at å tilsidesette bruk av P-GPS er fornuftig for vår anvendelse, men det ville vært interessant å utforske dette mer. Gitt mer tid ville vi gjort flere målinger med P-GPS under ulike forhold. Vi er også noe usikre på validiteten til målingen vi gjorde rundt P-GPS da det er mistenksomt at den bruker 10 minutter på å laste ned dataen. Vi kunne vært mer kritisk til dette da vi gjorde målingene. Men, det er den dataen vi fikk da vi målte så vi må anta at den er valid.

Det største savnet vårt er mangelfull testing av egen prototype. Vi har en antagelse om at vår krets kunne vært mer strømeffektiv enn utviklingskortet, da det er en enklere krets med færre komponenter. Gitt mer tid ville vi gjort de nødvendige endringer på prototypen for å få sammenlignet dette.

6.2.2 Manglende målinger av større intervaller

Av praktiske hensyn har vi gjort våre målinger med korte intervaller. Vi antar at dataen ikke er skalerbart for større intervaller, da den ferske GPS-dataen som ligger i modemet gjør det lettere å få neste fix. Dersom intervallene var lengre ville det vært mer krevende å få fix, og dermed ville strømtrekket reelt vært mer enn vi har beregnet. Strømmålinger ved en gitt intervall er trolig bare representativt for den gitte intervall.

6.3 Software

6.3.1 GPS Interval Issues

Ved bruk av fastsatte intervaller har vi erfart at de første tre posisjoneringene ikke følger det gitte intervallet. Vi kan ikke se noen grunn til at modemet skal oppføre seg slikt. Dette gir oss fix uønsket ofte i starten, og strømbruket blir naturligvis preget av dette.

6.3.2 Generelt om sanntidsprogrammering

Sanntidsprogrammering var et helt nytt tema for oss alle da vi startet oppgaven. Begreper som tråder, semaphores, workqueues etc var alle nye. Dette gjorde at det ble en veldig høy terskel for å forstå samplene vi baserte oss på, og mye tid i starten gikk bare til tolking av kode og dokumentasjon. Gjennom dette, og med god hjelp fra veileder Richard, har vi nå fått en langt større forståelse og grunnlag til å produsere kode.
Vi synes dessverre at for mye tid gikk til dette i starten, og et ”lynkurs” i starten kunne ført til et langt bedre sluttprodukt. Vi føler ikke at vi har utnyttet alle verktøyene i sanntidsprogrammering til det fulle, men er på tross av dette fornøyde med å ha produsert en velfungerende løsning i et helt nytt spillefelt.

Flere av samplene vi har basert oss på har benyttet logger-modulen i zephyr. Dette ga oss tidlig merkelige utfordringer, så vi endret at fra logging med et gitt nivå til å bare printe ting som vanlig. Derfra ”oppfant” vi funksjonen printk2, som bare printer dersom utviklingsmodus er skrudd på. Ved bruk av logging og definere hvilke lognivå man ønsker å printe ville dette blitt implementert på en mer fornuftig måte. Sett etterpåklokskapen burde vi fulgt veien med logging fremfor å finne opp hjulet på nytt med printk2.

Programkoden ble realisert på bakgrunn av samples fra Nordic. Med den kunnskapen vi har opparbeidet igjennom prosjektutførelsen kunne vi skapt et mer ryddig og effektivt program hvis vi startet på nytt med blanke ark. Dette er vel hele hensikten med en bacheloroppgave.

6.3.3 Feilhåndtering

Vår system, i likhet med alle samples, har lite eller ingen feilhåndtering. Dersom en rutine skulle feile på et vis, må systemet restartes for å forsøke å kjøre på nytt. Dette kunne vært håndtert bedre, med smartere bruk av tråder og scheduling av disse gitt feiling. Vi har feilhåndtering med gjenforsøk dersom vi mister tilgang til skyen; men ikke hvis GNSS- eller LTE-modul plutselig ikke svarer som forventet. Gitt mer tid ville vi lagt mer i å lage et mer driftsikkert system.

6.3.4 nRF Cloud

Gjennom hele utførelsen av prosjektet har vi hatt utfordringer med at nRF Cloud har blitt utilgjengelig for oss. Vi har hatt fem utviklerbrett som alle tidvis har mistet kontakten med skyen. Veileder har hjulpet oss med å få enkelte av oss opp her, og tilbudt flere brett og SIM-kort i en feilsøkingsprosess rundt dette. Disse utfordringene har gjort at enkelte funksjonaliteter har vært utilgjengelig for gruppens medlemmer, og har vært en hemsko i utviklingen. En periode hadde vi bare ett utviklingskort som fungerte med skyen, og dette gjorde selvsagt at utviklingen av skyfunksjonalitet gikk tregere enn ønsket. På tross av dette er vi svært fornøyde med den toveis-kommunikasjonen vi har fått til.
Et ønske for å fått ennå større læringsutbytte ville vært bruke en annen/egen skyløsning og back end, for å få et mer håndfast forhold til MQTT, REST, TCP etc. Nå er det veldig mye ”black magic” som håndteres (fabelaktig godt) av cloud-bibliotekene som vi gjerne skulle lært mer om. Vi ser at dette selvsagt ville gjort oppgaven betydelig større, og gitt oss strammere tidsbegrensninger.

6.3.5 NB-IoT

Vi har hatt et ønske om å sammenligne NB-IoT og LTE-M, og teste grensene for hvilke datamengder og hastigheter NB-IoT støtter, og hvordan dette kunne påvirket vårt produkt. Dette er også noe oppdragsgiver Nordic Semi viste interesse i at vi produserte data om.
Etter testing oppdaget vi at mangler NB-IoT-støtte i Norge fra iBasis, som er tjenesteleverandør for SIM-kort for Nordic Semi. Vi finner det ergelig å ha brukt tid på dette, når det var dømt til å feile.

7 KONKLUSJON

Strømmålinger
Det har blitt tydelig for oss at målinger og tester er viktige faktorer for å skape et ferdig produkt som yter bra. Stømtrekkanalysen med utgangspunkt i et veldig lite batteri høstet resultater som ikke var tilfredsstillende. 19 dager er for realistisk sett for liten tid til å ha et batteri som må byttes. Et større batteri er nødvendig. Samtidig viser målingene at strømtrekket er så lavt at solceller eller andre former for energihøsting er mulig. Dette er også noe som allerede er på markede fra firmaer som den norske Nofence. Våre beregninger med batterier som har 10000mAh kan man oppnå flere år med levetid. Med en slik levetid kan man begynne å tenke på videre problemstillinger som erosjon, innkapsling og vekt.

Hardware
Prototype utlegget endte opp som et godt og fungerende første utkast med noen mangler som kunne ha blitt fikset om det hadde vært rom til iterasjon. Dersom vi hadde hatt én iterasjon mer på prototypen er vi sikre på at produktet ville fungert med GPS-posisjonering og skykommunikasjon.
Det har vært krevende å utvikle maskinvare rundt nRF9160, noe grunnet manglende tilgang på god dokumentasjon på utfordringer man støter på. På tross av dette er vi svært fornøyde med eget resultat på maskinvaresiden.

Konseptdesignet ble ikke fysisk realisert men gir en god pekepinn på størrelsesorden og format som er mulig. Et faktisk sluttprodukt måtte naturligvis vært større for å gi plass til et batteri som gir tilstrekkelig batterilevetid.

Software
Vi har skapt et godt fungerende system, som oppfører seg som ønsket med effektiv posisjonering, toveis-kommunikasjon og lang batterilevetid.
Vi har lært veldig mye som sanntidsprogrammering, som ikke har vært et tema i studiet ellers. Vi er derfor svært fornøyde med oppgaven vi fikk tildelt.

Det vi angrer mest er at vi ikke fikk testet vår programvare på vårt eget produkt. Da ville oppgaven vært fullkommen i våre øyne, men når første utkast av prototypen kom først en uke før innleveringsfristen ble det som det ble.

Anerkjennelse
Vi vil avslutningsvis gi en spesielt stor takk til våre veiledere i Nordic Semiconductors, Richard McRae og Eirik Hovde Skanke. Takk for en interessant og lærerik utfordring, og for all hjelp og kunnskap underveis.

8 VEDLEGG

1. Poster
2. All kode ligger vedlikeholdt på GitHub
3. Ufiltrert data
4. Komplett PA-mappe ligger på teams som veileder har tilgang til
5. Skjematikk, utlegg og BOM fra Altium tilgjengelig på Teams

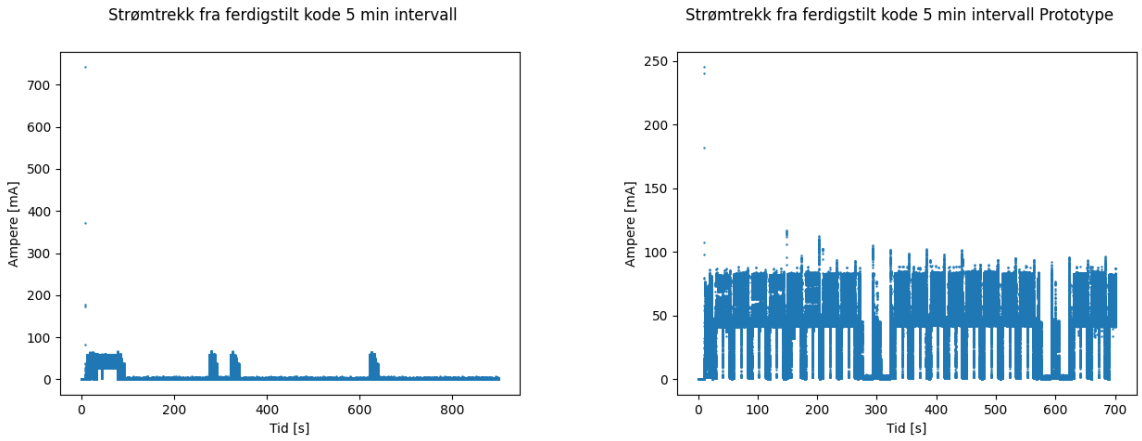


Figure 47: Vedlegg 3: Ufiltrert data fra oppstartstrømmer DK og Protorype

Referanser

[1] Map Toaster. How gps works. https://www.maptoaster.com/maptoaster-topo-nz/articles/how-gps-works/how-gps-works.html, 2014.

[2] Nordic Semiconductor. nrf9160: A-gps. https://developer.nordicsemi.com/nRF_Connect_SDK/doc/1.7.1/nrf/samples/nrf9160/agps/README.html, 2021.

[3] Nordic Semiconductor. nrf9160 p-gps. https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/libraries/networking/nrf_cloud_pgps.html, 2021.

[4] Nordic Semiconductor. Gnss interface. https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrfxlib/nrf_modem/doc/gnss_interface.html, 2021.

[5] Telenor IoT. Lte-m vs nb-iot – a guide exploring the differences between lte-m and nb-iot. https://iot.telenor.com/iot-insights/lte-m-vs-nb-iot-guide-differences/, Hentet 2022.

[6] RF WIRELESS. Psm vs edrx-difference between psm and edrx gsm,lte-m,nb-iot. https://www.rfwireless-world.com/Terminology/Difference-between-PSM-and-eDRX-in-GSM-LTE-M-NB-IoT.html, Hentet 2022.

[7] Nordic Semiconductor. Low power cellular iot. https://www.nordicsemi.com/Products/Low-power-cellular-IoT/What-is-cellular-IoT, Hentet 2022.

[8] Constantine A. Balanis. Modern antenna handbook - fundamental parameters and definitions for antennas. Hentet fra NTNU Universitetsbibliotek, 2008.

[9] Cuong Phu Le. Transmisjonslinje. Power Point - IELET2109 - Elektronikkonstruksjon, 2021.

[10] Antenna Test Lab. Return loss and vswr, 2022.

[11] ZM Peterson. Coplanar waveguide design for your rf pcb. https://www.nwengineeringllc.com/article/coplanar-waveguide-design-for-your-rf-pcb.php, Oktober 22 2020.

[12] Atar Mittal. Differential pairs in pcb transmission lines. https://www.protoexpress.com/blog/differential-pairs-in-pcb-transmission-lines/, Februar 3 2020.

[13] Torbjørn Øvrebekk. The importance of average power consumption to battery life. https://blog.nordicsemi.com/getconnected/the-importance-of-average-power-consumption-to-battery-life, September 30 2020.

[14] Pål Kastenes. Power consumption explained. https://blog.nordicsemi.com/getconnected/power-consumption-explained, July 29 2020.

[15] Luís Cruz. All you need to know about energy metrics in software engineering. https://luiscruz.github.io/2021/09/01/energy-units.html, September 01 2021.

[16] Nordic Semiconductors. Power profiler kit ii. https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2, 2022.

[17] Steven Keeping. The advantages of pulse frequency modulation for dc/dc switching voltage converters. https://www.digikey.no/en/articles/the-advantages-of-pulse-frequency-modulation-for-dc-dc-switching-voltage-converte Mars 25 2014.

[18] Nordic Semiconductor. nrf9160 antenna and rf interface guidelines. https://infocenter.nordicsemi.com/pdf/nwp_033.pdf, Hentet 2022.

[19] Skyworks. Sky65943-11. https://www.skyworksinc.com/Products/Front-end-Modules/SKY65943-11, Hentet 2022.

[20] Nordic Semiconductor. Cloud services. https://www.nordicsemi.com/Products/Cloud-services, Hentet 2022.

[21] Nordic Semiconductor. nrf connect for desktop. https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-desktop, Hentet 2022.

[22] Nordic Semiconductor. nrf connect sdk. https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/index.html, Hentet 2022.

[23] The Linux Foundation. About the zephyr project. https://www.zephyrproject.org/learn-about/, Hentet 2022.

[24] Software Freedom Conservancy. About - git. https://git-scm.com/about, Hentet 2022.

[25] The Linux Foundation. West (zephyr's meta-tool). https://docs.zephyrproject.org/latest/develop/west/index.html, Hentet 2022.

[26] Tom Preston-Werner. Semantic versioning 2.0.0. https://semver.org/, Hentet 2022.

[27] The Linux Foundation. Zephyr cmake package. https://docs.zephyrproject.org/latest/build/zephyr_cmake_package.html, Hentet 2022.

[28] The Linux Foundation. Devicetree. https://docs.zephyrproject.org/latest/build/dts/index.html, Hentet 2022.

[29] The Linux Foundation. Kconfig. https://docs.zephyrproject.org/2.6.0/guides/kconfig/index.html, Hentet 2022.

[30] Nordic Semiconductor. 3rd party toolchain. https://docs.zephyrproject.org/3.0.0/getting_started/toolchain_3rd_party_x_compilers.html, Hentet 2022.

[31] IncludeHelp. The 'super loop' architecture for embedded c programming. https://www.includehelp.com/c/the-super-loop-architecture-for-embedded-c-programming.aspx, Hentet 2022.

[32] The Linux Foundation. Threads. https://docs.zephyrproject.org/3.0.0/reference/kernel/threads/index.html, Hentet 2022.

[33] The Linux Foundation. Mutexes. https://docs.zephyrproject.org/2.6.0/reference/kernel/synchronization/mutexes.html, Hentet 2022.

[34] The Linux Foundation. Semaphores. https://docs.zephyrproject.org/3.0.0/reference/kernel/synchronization/semaphores.html, Hentet 2022.

[35] Dr. Johan Kraft Embedded Computing Design. Rtos debugging, part 5: Deadlock ? when everybody wants the fork. https://embeddedcomputing.com/technology/software-and-os/rtos-debugging-part-5-deadlock-when-everybody-wants-the-fork, Hentet 2022.

[36] Javapoint. The dining philosophers problem. https://www.javatpoint.com/os-dining-philosophers-problem, Hentet 2022.

[37] Nordic Semiconductors. nwp037 - nrf9160 hardware design guidelines. https://infocenter.nordicsemi.com/index.jsp?topic=%2Fnwp_037%2FWP%2Fnwp_037%2Fnwp_037_intro.html, Hentet 2022.

[38] Silicon Labs. Data sheet cp2102. https://www.silabs.com/documents/public/data-sheets/CP2102-9.pdf, Januar 20 2017.

[39] Nordic Semiconductor ASA. nrf9160 antenna and rf interface guidelines. https://infocenter.nordicsemi.com/pdf/nwp$_0$33.pdf, 2020.

[40] TAOGLAS. Dsgp.1575.18.4.a.02. https://no.mouser.com/datasheet/2/398/DSGP$_1$575$_1$8$_{4A0}$2−1282464.pdf, $Hentet2022$.

[41] Ignion. All mxtendtm (nn02-220). https://no.mouser.com/datasheet/2/1029/DS$_N$N02$_2$20−2657955.pdf, 2021.

[42] Ethertronics. P822601/p822602 datasheet). https://docs.rs-online.com/4550/A700000006940710.pdf, 2020.

[43] Ignion. Trio mxtendtm (fr01-s4-210)). https://no.mouser.com/datasheet/2/1029/UM$_F$R01$_S$4$_2$10$_1$−1863540.pdf, 2017.

[44] Mad PCB. Pp. https://madpcb.com/pp/, Hentet 2022.

[45] Nordic Semiconductors. ppk2_pc_nrf9160_dk_usb.svg. https://infocenter.nordicsemi.com/topic/ug_ppk2/UG/ppk/PPK_images/ppk2_pc_nrf9160_dk_usb.svg, Hentet 2022.

[46] Movable Type Scripts. Calculate distance, bearing and more between latitude/longitude points. https://data.energizer.com/pdfs/cr2032.pdf, Hentet 2022.

[47] Energizer. Energizer cr2032 datasheet. https://www.movable-type.co.uk/scripts/latlong.html, Hentet 2022.

[48] Nordic Semiconductor. Online power profiler for lte. https://devzone.nordicsemi.com/power/w/opp/3/online-power-profiler-for-lte, Hentet 2022.

[49] Nordic Semiconductor. Modem current consumption. https://infocenter.nordicsemi.com/index.jsp?topic=%2Fps_nrf9160%2F_tmp%2Falta.nRF9160%2Fautodita%2FCURRENT%2Fparameters.id_current_modem.html&cp=2_0_0_4_1_0_14, Hentet 2022.

Sheep Acquisition Unit - S.A.U.
A Low Power GPS Tracker Made with Nordic Semiconductors nRF9160

Torje Farbu Johansen, Eirik Ekrheim Bjørklund, Håvard Kalliainen, Andreas Ipsen & Jonas Saari

Institutt for Elektroniske Systemer, NTNU

Introduksjon

På oppdrag fra Nordic Semi har vi blitt bedt om å produsere en Low Power GPS Tracker. Anvendelsen var valgfri, og vi valgte å lage et produkt for å spore sau.
Prosjektet er todelt. vi skal produsere og sette sammen både det fysiske produktet samt kode den med spesielt hensyn på lav energibruk.

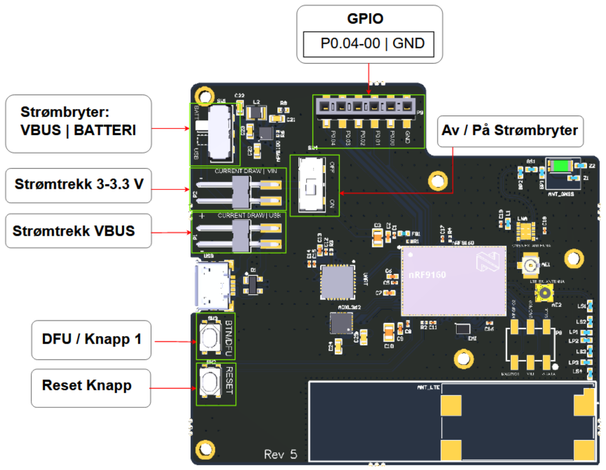
Prosjektet baserer seg på Nordic Semiconductors nRF9160 SiP, og enheten kommuniserer med Nordics skyløsning nRF Cloud.

I prosjektet benytter vi Nordics Power Profiling Kit II (PPK2) for å gjør strømtrekkanalyse. Med denne har vi analysert strømtrekk ved ulike operasjonsmoduser, LTE moduser, GPS moduser og ulike kommunikasjonsprotokoller.

Hardware

Følgende designvalg og betrakninger ble hensyntatt under produksjon:

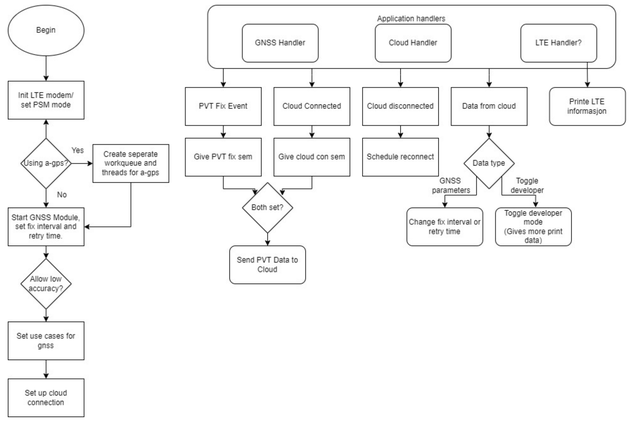
- Antenner - Adskilt for LTE og GPS, både innebygd og mulighet for ekstern
- PMIC - Ekstern strømforsyning og for mer stabil drift
- GPS LNA - Low Noise Amplifier for mottak av GPS
- Transmisjonslinje og tuning av denne
- Kablet kommunikasjonsgrensesnitt (UART etc)
- Grensesnitt for strømmåling (spesielt i utviklerfasen)



Figur 1. Forside av prototype

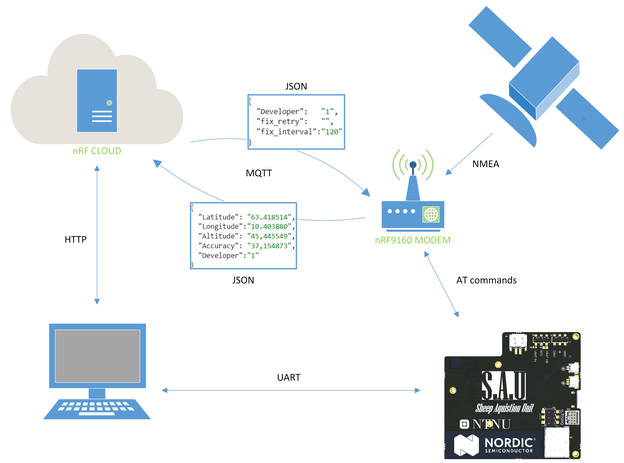
Software

Modemet på nRF9160 er en egen, ikke-programmerbar enhet, og kommuniserer med vårt program via AT-commands. Disse er wrapped i funksjoner tilgjengeliggjort via Nordics biblioteker. Hendelser fra modemet kommuniseres til programmet via handlers. Vi som utviklere legger til rette for dette ved å sette opp handlers i programmet, og når modemet har informasjon som ny GPS-plassering, endring i LTE-tilkobling, data fra skyen etc. blir disse handlers kallt. Fra utviklersiden setter vi oss selv hvordan vi agerer på de ulike hendelsene.
nRF9160 kjører Zephyr RTOS, og gir oss med dette kraftige verktøyer for sanntidsprogrammering.



Figur 2. Flytskjema av program

Funksjonalitet: Programmet vårt innhenter GPS-posisjon (fix) på forhåndsbestemte intervaller, og sender disse til skyen. Enheten er stadig koblet til LTE med Power Saving Mode (PSM)Fra skyen kan brukeren endre intervallet for innhenting av GPS-data. En JSON-streng kan sendes vil endre intervalltiden. Et lengre intervall vil være mer strømbesparende.
Brukeren kan velge å benytte seg av A-GPS, eller å slå av LTE utenfor sendinger. Dette vil også påvirke strømtrekket.



Figur 3. Dataflyt i prosjekt

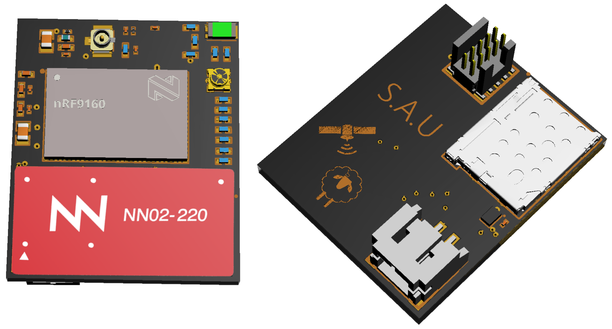
Strømtrekkanalyse

Gjennom å analysere vårt produkt i vanlig drift har vi følgende resultater for ulike intervaller av GPS-posisjonering. Estimatene er basert på et CR2032-batteri med 220 mAh:

Beregnet levetid ved **5 minutters** intervaller	8.1 dager
Estimert levetid ved **60 minutters** intervaller	17.7 dager
Estimert levetid ved **8 timers** intervaller	19.7 dager

Endelig produkt

Det endelige produktet er på 31x26mm, og har en estimert levetid på opp mot 20 dager med 8-timers sendingsintervall.
Den støtter full duplex skykommunikasjon, hvor posisjonsdata sendes etter ønsket intervall som kan styres fra skyen.



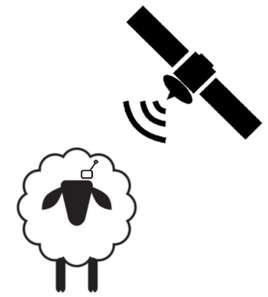
Figur 4. Konseptdesign

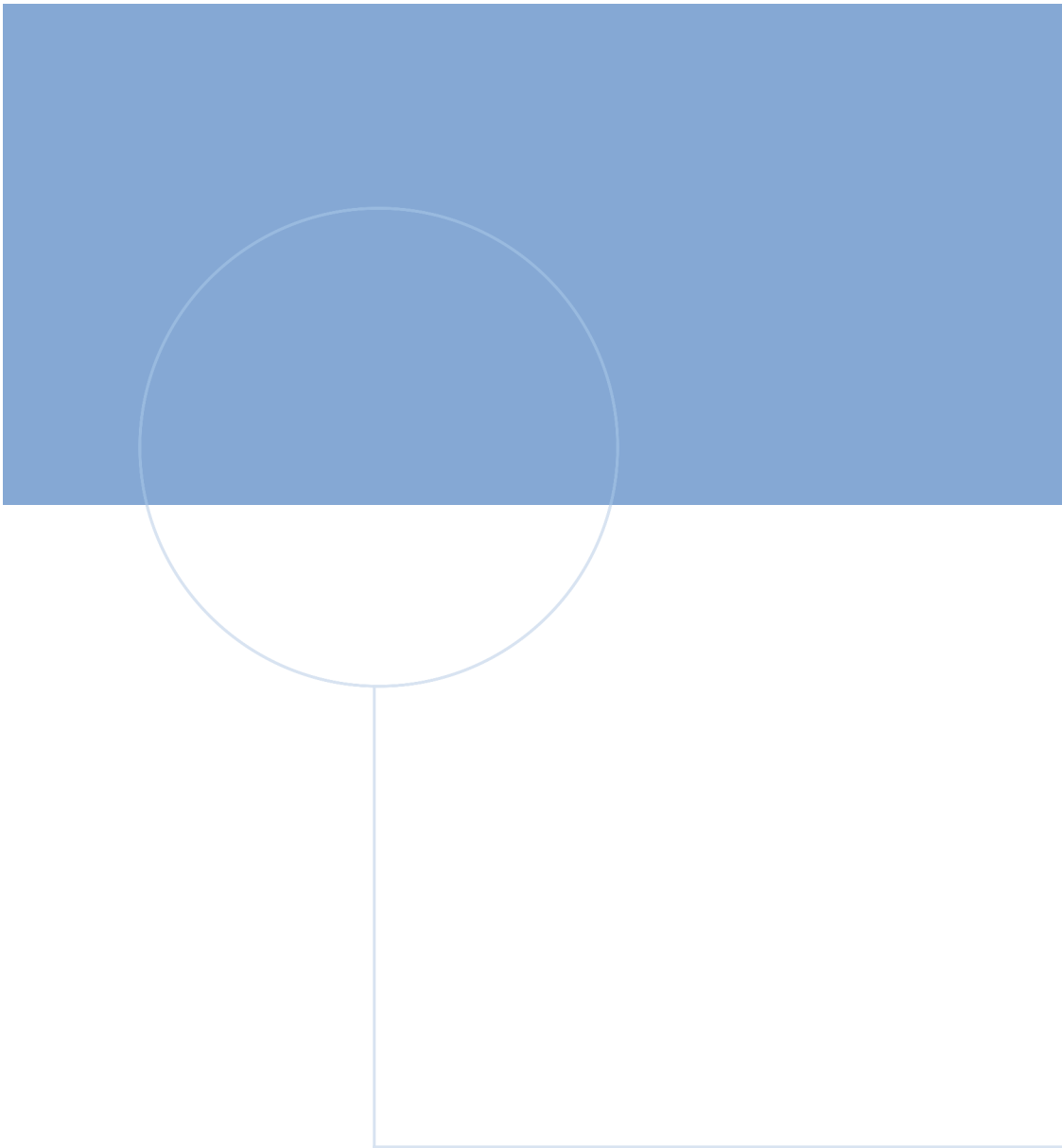
Notat om estimering av strømtrekk

Det er gjort målinger på GPS fix med 5-minutters intervaller. Dette er deretter skalert opp til større intervaller. Den eksisterende og ferske GPS-dataen som da ligger i modemet gjør at de følgende posisjoneringene er mindre strømkrevende enn en fersk måling etter f.eks 8 timer ville vært.

Anerkjennelse

Vi ønsker å gi en spesielt stor takk til Richard McCrae og Eirik Skanke ved Nordic Semiconductors for å tilby en spennende og lærerik oppgave, og for all hjelp og kunnskap de har delt med oss underveis i gjennomføringen.





NTNU
Norwegian University of
Science and Technology

NORDIC
SEMICONDUCTOR