BACHELOROPPGAVE:

BO22EB-09 Implementering av lav-energi Mesh Sensor Nettverk

André Sunde Jordalen
Petter Rødal

30. mai. 2022

Dokumentkontroll

Rapportens tittel: BO22EB-09 NSE Mesh Nettverk	*Dato/Versjon* 25.mai.22/0.12
	Rapportnummer: BO22EB-09
Forfatter(e): André Sunde Jordalen Petter Rødal	*Studieretning:* AUTB19
	Antall sider m/vedlegg 47
Høgskolens veileder: Tom Kjøde	*Gradering:* Åpen
Eventuelle Merknader: Vi tillater at oppgaven kan publiseres.	

Oppdragsgiver: North Sea Electronics	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson(er) (inklusiv kontaktinformasjon): Richard Fyhn: 924 80 665, rfy@nse.no Geir Lasse Kaldestad: 971 87 818	

Revisjon	**Dato**	**Status**	**Utført av**
0.11	15.05.22	Første utkast	André Jordalen og Petter Rødal
0.12	25.05.22	Andre utkast	André Jordalen og Petter Rødal
0.13	30.05.22	Siste utkast	André Jordalen og Petter Rødal

Forord

Denne rapporten er skrevet i sammenheng med vår avsluttende bacheloroppgave ved Høgskulen på Vestlandet avdeling Bergen, våren 2022. Oppgaven er gjort i samarbeid med North Sea Electronics, og omhandler mesh-nettverk, som er en del av prosjektet Fjellvåk som de holder på med.

Vi vil takke Richard Fyhn (CTO) og Geir Lasse Kaldestad (CEO) hos North Sea Electronics for muligheten til å ta del i deres prosjekt. De har vist stor tillit til oss under hele prosjektperioden, noe vi er veldig takknemlige for. Vi vil også takke Simula UiB som ga oss muligheten til å bruke laben deres sammen med utstyret de hadde tilgjengelig.

Til slutt vil vi takke vår veileder Tom Kjøde som har hjulpet oss med det vi har trengt hjelp til, om det har vert spørsmål om diverse, eller om det har vert utstyr han har stilt opp med. Det var også han som fikk oss i kontakt med Simula UiB, noe som har vert til stor hjelp i denne prosjektperioden.

Sammendrag

Dette bachelorprosjektet har gått ut på å finne den best egnede mesh-teknologien for å sende data fra fjellvegg/tunell til en basestasjon. Mesh nettverket må bruke minst mulig strøm siden de fleste nodene i nettverket skal være batteridrevet. Det må og kunne ha en rekkevidde på 50-100 meter mellom de aktuelle nodene.

Hensikten med oppgaven er at mesh nettverk er helt nytt for NSE, og de vil derfor gi oss muligheten til å undersøke og teste dette. Deretter vil vi gi videre informasjonen, koden og erfaringene vi har tatt med oss, slik at det blir lettere for dem å implementere det inn i prosjektet sitt.

Resultatet av arbeidet er et mesh-nettverk av typen Zigbee. Nodene i nettverket har evnen til å sove for å bruke minst mulig strøm, og koble seg automatisk til nettverket igjen når de våkner for å så gi data til koordinatoren i nettverket. Mer informasjon om ytelsesparameterne til nettverket finnes i kapittel 4.2 Testing. Alt i alt lever løsningen vår ganske bra opp til kravspesifikasjonen. Noen punkter som skjemategning og PCB-layout ble senere i prosjektet bestemt at NSE skulle ta seg av, så disse gikk ut av kravspesifikasjonen.

Innholdsfortegnelse

1 Innledning

1.1 Oppdragsgiver

North Sea Electronics spesialiserer seg i utvikling av høy-temperatur elektronikk for røffe omgivelser, som for eksempel elektronikk som skal operere i borehull i olje og gassnæringen. De tilbyr ende-til-ende løsninger som inkluderer kretsdesign, layout, montering og testing.
NSE tilbyr også utvikling av sann-tids datalogging og kontrollsystem. Med lang erfaring relatert til subsea-kjøretøy, og detaljert kunnskap innenfor elektronikk og mikrokontrollere, kan NSE håndtere høyteknologi prosjekt både på land, offshore, subsea og nedihull.
North Sea Electronics AS ble stiftet i 2005 og har i dag 12 ansatte. Bedriften holder til på Gravdal, og hadde i 2020 en omsetning på NOK 28 223 000. [18][19]

Figur 1 North Sea Electronics

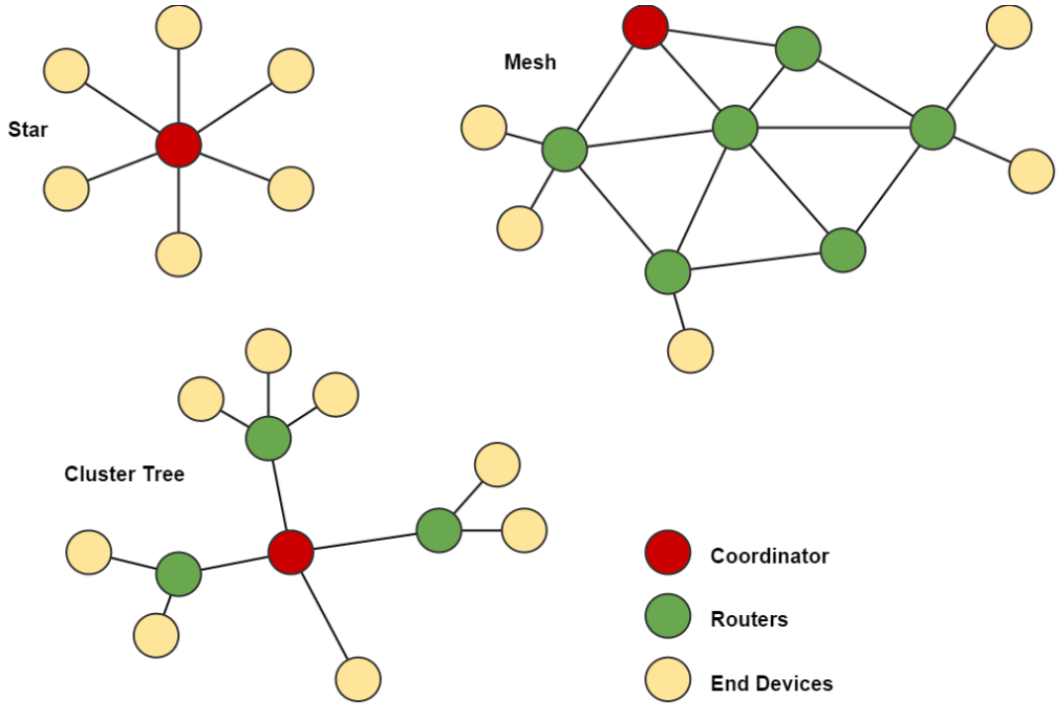
1.2 Problemstilling

Problemstillingen fra North Sea Electronics er at de skal utvikle et system for kontinuerlig overvåking av fjellskjæringer og tunneltak. Bakgrunnen bak problemstillingen er å kunne hindre tap av liv og verdier, hindre uforutsette stengninger av veier og kunne gi tidlig deteksjon av endringer for å kunne utføre preventivt vedlikehold. Dette vil de gjøre ved å installere noder med diverse sensorer på fjellsikringsbolter, kalt Fjellvåk. Fjellvåk skal inneholde sensorer for måling av kompresjon/moment, temperatur og akselerasjon (vibrasjon), og skal kunne sende sensordata ved faste intervaller til et kontrollrom. Fjellvåk må være brukervennlig for en "fjellsikrer" å installere, og ha batteritid på om lag 10 år. Vår oppgave er å identifisere den beste mesh nettverksteknologien for dette prosjektet, og deretter teste ytelsesparameter ved valgt løsning.

Figur 2 Fjellvåk

1.3 Hovedidé for løsningsforslag

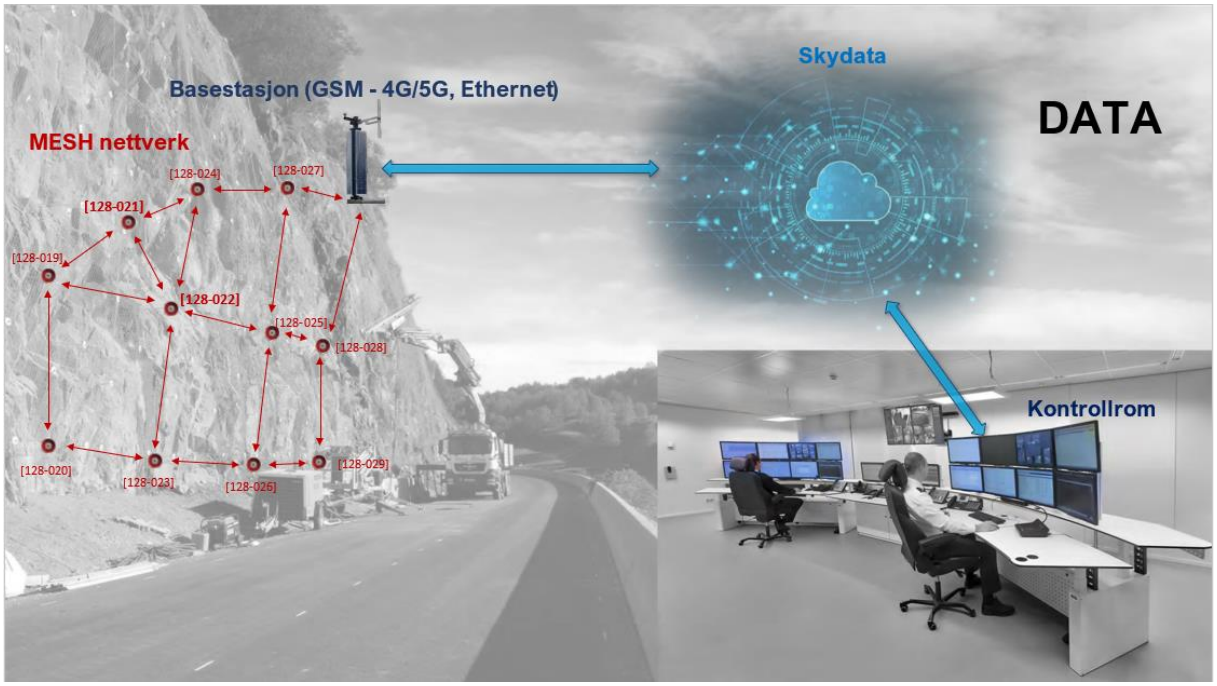
Oppdragsgivers hovedidé for nettverkstopologi er Mesh-nettverk. Et Mesh-nettverk vi gi Fjellvåk-nodene muligheten til å sende data via hverandre, og gi nodene muligheten til å «sove» for å spare strøm. Mesh-nettverk gir og muligheten for batteridrevne noder.



Figur 3 Mesh nettverk

2 Kravspesifikasjon

Sensorpakker skal festes ytterst på bolter i fjellet for å måle moment, vibrasjoner, temperatur, osv. Disse sensorpakkene skal sende data periodisk til en basestasjon som deretter sender dataen videre til et kontrollrom. Nodene må også være billigst mulig, og enkle å installere for en montør. Med andre ord skal sensorpakkene være mest mulig «plug and play». Oppdragsgiver har bestemt at nodene skal operere i et mesh-nettverk, og det er dermed utvikling av mesh—nettverket som er vår oppgave i dette prosjektet. Kravspesifikasjonen for mesh-nettverket kommer i et senere kapittel.



Figur 4 kravspesifikasjon

2 Hoveddel

2.1 Oversikt

I hoveddelen skal alle de forskjellige teknologier og funksjonalitetene som er brukt i prosjektet forklares. Med dette får vi en oversikt over helheten av oppgavene før en går dypere ned i de forskjellige temaene. Det første vi måtte gjøre i prosjektet var å velge hvilken mesh-teknologi vi tenkte ville passe best for formålet. Dette er beskrevet kapittel i 2.2. I kapittel 2.3 har vi gått dypere inn på hvordan den valgte teknologien fungerer og implementeres. Noe som gjør kapittelet ganske teknisk forklart. I kapittel 2.4 er det beskrevet hvilke utviklingskitt vi har valgt å bruke for å teste og sette opp nettverket. Til slutt i kapittel 2.5 er det beskrevet hvilket verktøy vi har brukt til å utvikle og bygge våre egne applikasjoner for utviklingskittet.

2.2 Mesh-teknologier

2.2.1 Kravspesifikasjoner for mesh-teknologien

Første del av prosjektet var å identifisere hvilke mesh-teknologi som vil egne seg best. NSE hadde følgende hovedkrav til mesh nettverket:

- relativ lav kostnad.
- 50-100m rekkevidde mellom nodene.
- lang batterilevetid.

Vi studerte ulike mesh-teknologier og fant at de følgende ville være mest aktuelle, Bluetooth mesh, ZigBee og Thread. Fellestrekk for disse teknologiene er at de er lav strøm og kan brukes i sensor nettverk.

2.2.2 Bluetooth mesh

Figur 5 Bluetooth logo

Et Bluetooth mesh nettverk inneholder en samling av spesielt forberedte Bluetooth enheter som refereres til som noder. Det kan være flere tusen av noder i et mesh nettverk og de kommuniserer med hverandre med å sende meldinger. Bluetooth mesh bruker frekvenser mellom 2.4 - 2.48 GHz.

Meldinger kan hoppe fra node til node gjennom en prosess som er kjent som videresending (relaying) til de når sin endelige destinasjon. Med dette kan kommunikasjon ta sted vesentlig utenfor rekkevidden til den underliggende Bluetooth-radioen.

Bluetooth mesh nettverk er designet til å være veldig stabilt og pålitelig, der meldinger reiser til deres destinasjon flere veier gjennom nettverket. Dette er kjent som mange-vei (multi-path) meldingsoverføring. Bluetooth mesh kan da håndtere ulike utfordringer til radiokommunikasjon som kan eksistere i et miljø, for eksempel fysiske hindringer som vegger.

Bluetooth mesh er en protokoll stakk, altså den består av en gruppe protokoller som alle jobber sammen for å tillate programvaren å utføre en funksjon. Bluetooth mesh er dermed ikke en radioteknologi. Bluetooth mesh stakk sitter på toppen av en Bluetooth Low Energy (BLE). BLE er en trådløs nettverksteknologi, som i forhold til den "klassiske" Bluetooth, skal gi betraktelig redusert strømforbruk og samtidig opprettholde lignende kommunikasjonsrekkevidde.[25] BLE gir den underliggende radio kommunikasjonsevnen brukt av Bluetooth mesh. [6]

Bluetooth mesh er en lavstrømprotokoll, og en studie gjort for Universitat Politècnica de Catalunya har funnet ut at en sensorenhet som kjører på et enkelt 235 mAh batteri (cr2032 button cell battery), og sender data meldinger hvert 10. sekund, kan oppnå en levetid på 15.6 måneder. [15]

Fordeler med Bluetooth mesh:

Fordi Bluetooth mesh er basert på BLE, bærer den over mange av protokollens fordeler, inkludert lavt strømforbruk, god sikkerhet, beaconing støtte. Bluetooth mesh er selv-formerende og selv-helbredende, med "sove" støtte for end-device. Bluetooth mesh har også god rekkevidde (100-1000m), og veldig god data rate på opp til 1Mbit\s.

Begrensninger med Bluetooth mesh:

Bluetooth mesh er fortsatt er ny protokoll, og undergår fortsatt forbedringer og revisjoner. Den er fortsatt ikke fullt støttet, som betyr at OEM utstyr, gateways og håndholdte enheter vil sannsynligvis ikke være fullstendig kompatible. Dette vil mest sannsynlig forbedres etter hvert som protokollen får gjennomslag.

"Managed flooding" (alle enheter innen rekkevidde mottar meldinger) protokollen gjør nettverk designet mer simpelt, men det er en avveiing i effektivitet og strømforbruk. Bluetooth mesh bruker heller ikke IP adressering, derfor må interaksjoner med internett og sky servere, skje gjennom en fikset gateway.[5]

2.2.3 Zigbee

Figur 6 Zigbee logo

ZigBee er en lav-kostnad, lav-energi, trådløs mesh-nettverksstandard basert på IEEE 802-15-4 standarden, brukt i trådløse kontroll- og overvåkingsapplikasjoner. Målet med ZigBee teknologien er å være enklere og billigere enn andre trådløse nettverk som for eksempel Bluetooth. Bruksområde inkluderer trådløse lysbrytere, energiovervåkning i hjemmet og andre applikasjoner som krever lav distanse dataoverføring med lav rate. [1] Zigbee opererer på 2.4 Ghz, 900 MHz og 868 MHz frekvenser.

ZigBee fungerer ved at en koordinator, som ligger ved roten av ZigBee nettverket, sender en melding til en ruter som deretter sender meldingen videre til en ende-enhet med instruksjoner for å utføre en spesifikk oppgave.

På grunn av sitt lave strømforbruk er rekkevidden til ZigBee begrenset til ca. 100 meter ved fri sikt, avhengig av miljøet. Enheter kan fortsatt sende data over større distanser ved å sende data via andre noder i mesh-nettverket. ZigBee er typisk brukt i applikasjoner som krever en lav datarate, lang batteritid og kryptering. ZigBee har en definert rate på 250kbit/s som gjør at den er best egnet til å sende data fra en input-kontroller eller sensor. For sikkerhet bruker ZigBee 128-bit symmetrisk krypteringsnøkler, som anses som veldig sterk og robust. [2]

ZigBee oppgir med at protokollen kan tilby batterilevetid på opptil flere år. Med for eksempel et CR2032 batteri kan ZigBee enheter få opptil 2.4 års levetid, og med et CR2 batteri vil levetiden være rundt 5.5 år. [14]

Fordeler med ZigBee mesh:

ZigBee er både enklere og billigere å bruke enn andre tilsvarende wireless personal area networks (WPAN) og har i tillegg veldig lang batteritid. Som en interoperabel standard kan enheter fra mange forskjellige produsenter kommunisere problemfritt, noe som bidrar til å skape Zigbees brede aksept innen både hjemme automasjon og industriell IoT, med mange OEM-utstyrsalternativ på markedet.

Andre fordeler er at ZigBee leverer også “lav latens” kommunikasjon, det vil si den kan prosessere et høyt volum av data med lite forsinkelse. Mesh-nettverket konfigurer seg selv automatisk (self-

forming) og vil rekonfigurere dynamisk for å reparere seg selv hvis noder er deaktiverte eller fjernet (self-healing). [5]

Begrensninger med ZigBee mesh:

Begrensninger inkluderer lav overføringshastighet av data, relativt lav rekkevidde (i forhold til Bluetooth mesh) og høye kostnader for vedlikehold. Zigbee bruker heller ikke IP-adressering, derfor må gateways bli installert for å kommunisere med internett og skytjenester.

2.2.4 Thread

Figur 7 Thread logo

Thread er en trådløs lav-strøm mesh nettverksprotokoll (2.4 GHz) basert på internettprotokoll (IP), og bygget med åpne og velprøvde standarder. Thread nettverk har ingen enkelt feilpunkt og inkluderer evnen til selvhelbredelse. Nettverket rekonfigurerer automatisk når en enhet er lagt til eller fjernet. Thread sikrer ende-til-ende kommunikasjon – enhet-til-enhet, enhet-til-mobil, og enhet-til-sky.

Thread enheter integreres problemfritt med større IP-nettverk og trenger ikke proprietære gateways eller oversettere (translators). Dette reduserer infrastruktur investeringer og kompleksitet, fjerner potensielle feilpunkter og reduserer vedlikeholds byrder. Thread kobler også enheter sikkert til skyen, noe som gjør det enklere å kontrollere IoT-produkter og systemer fra personlige eller administrative enheter som mobiltelefoner og nettbrett. [12] I likhet med ZigBee er ikke dataraten den best, på 250 Kbit/s, men godt nok for å sende sensordata. Batterilevetiden er også god, der en enhet som 'våkner' hvert 10. Sekund med et CR2032 batteri vil vare i omtrent 2.6 år. [16]

Fordeler med Thread:

Thread tilbyr direkte adressebarhet til enheter ved bruk av IP-adresser. Thread er også skalerbart, derfor kan den automatisk skalere fra grunnleggende stjernetopologi med enkel ruter til mesh-topologi når flere rutere er til stede. Thread Group oppgir også at protokollen er vanskeligere å hacke, der den bruker avansert kryptering.

Begrensninger med Thread:

En av ulempene med Thread er at den har relativ kort rekkevidde, opp mot 30 meter, noe som gjør at Thread er hovedsakelig brukt i smart-hjem for hjemme automasjon. [10] I tillegg gjør den innebygde IPv6-adresseringen at ruteren må konverteres fra 802.15.4 til et ekstra IP-grensesnitt. Thread har også litt dårligere latens enn for eksempel ZigBee og støtter "bare" opp mot 250 enheter, noe som er langt færre enn både Bluetooth mesh og ZigBee. En annen grunn til at Thread ennå ikke har blitt så populær, er på grunn av protokollens interoperabilitet problemer. [11]

Oppsummering av teknologiene

Tabell 1 Sammenligning av nøkkelparametere

	Bluetooth Mesh	ZigBee	Thread
Rekkevidde	100-1000m	Ca. 100m	Ca. 30m
Strømforbruk	Middels	Best	God
Overføringshastighet	1 Mbit/s	250 Kbit/s	250 Kbit/s
Sikkerhet	God	God	Best
Tilgjengelighet	God	Best	Middels
Kompleksitet	Kompleks	Enkel	Enkel
Antall noder teknologien støtter	32 000*	64 000*	250

*** I teorien, i praksis er antallet begrenset. Men teknologiene kan støtte godt over 1000 noder.**

2.2.5 Valgt løsning

Med tanke på at mesh-nettverket skal brukes i overvåking av fjellsikring, der lav strøm, lang batterilevetid og relativ lang rekkevidde er essensielt, er alle teknologiene nevnt ovenfor mulig å bruke, men vi har til slutt konkludert med at ZigBee er den beste mesh teknologien for dette prosjektet.

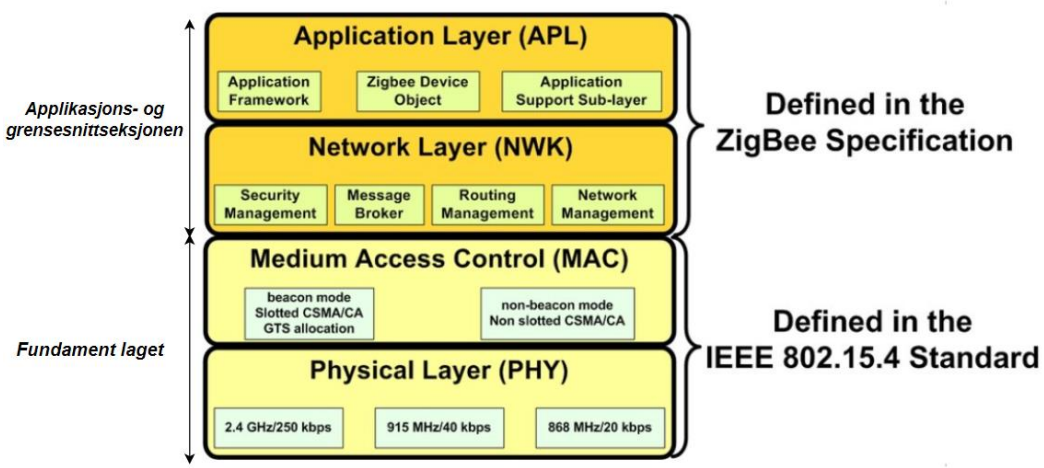
Bakgrunn for valg av Zigbee

- ZigBee utmerker seg med lavest strømforbruk av de vurderte teknologiene
- ZigBee har fordelen med at end-devices kan "sove" for å forlenge levetiden til batteriet. Det gir muligheten til å sende data fra en sensor på rimelige batterier for en rekke overvåkings- og kontroll applikasjoner. Noe som er veldig gunstig for dette prosjektet.

- Protokollen det beste utvalget av pålitelige kommersielle enheter, der dokumentasjon er allment tilgjengelig og dekker mange forskjellige bruksområder.
- Rutingtabeller, adresse oppløsning, sikkerhet, gjenforsøk og bekreftelser innebygd i protokollen, noe som sparer utviklingstid i forhold til de andre protokollene.

2.3 Zigbee

2.3.1 Stakk arkitektur



Figur 8 Zigbee stakk arkitektur

Zigbee arkitekturen som også er kalt Zigbee stakken, består av to lag som er grunnlaget for Zigbee. De to lagene er fundament laget og applikasjons- og grensesnittseksjonen.

Fundament laget:

Dette laget er definert av IEEE 802.15.4 standarden. Både det fysiske laget og Media Access Control (MAC) lagene fungere som fundament lagene for Zigbee arkitekturen.

Det fysiske laget:

De fysiske og elektriske kjennetegnene er definert av det fysiske laget. Dette laget er ansvarlig for dataoverføring, mottakelse og for å kartlegge informasjonsbits og tillate dem å reise ved modulasjons- og spredningsteknikker.[20]

Det fysiske laget er ansvarlig for følgende funksjoner:

- Aktivering og deaktivering av overføring og mottakelse.
- Kanal valg.
- Sende og mottakelse av pakker.
- Effektdeteksjon i kanalen.

Medium Access Control (MAC) laget:

Dette laget gir grensesnitt mellom det fysiske laget og nettverkslaget. Den definerer hvordan 802.15.4-radioer som opererer i samme område vil dele frekvensbåndet. Datahåndtering og dataledelse er hovedfunksjonene til MAC laget.

Datahåndtering inkluderer funksjoner som «Data forespørsel» og «Data bekreftelse». MAC laget legger til destinasjons adresse og overførings alternativer for de utgående data-rammene. Når Zigbee nettverket kaller «data forespørsel» funksjonen, blir dataen formattert inn i relevant MAC header og rammelengden er lagt til. Data-rammen er da klar til å bli overført.[20]

Formålet med «Data bekreftelse» funksjonen er å kommunisere statusen av den overførte dataen. Den sender en feilmelding når overførings-rammene overskrider eller når det er ingen respons til den overførte dataen.

MAC laget er ansvarlig for følgende:

- Beacon generering og styring.
- CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance) implementering.
- Garantert Time Clot-styring (GTS).
- Dataramme validering og bekreftelse.
- Dataoverføring for de øvre lagene.

Applikasjons- og grensesnitt seksjon:

Denne seksjonen er definert av Zigbee-spesifikasjonene og inneholder nettverkslaget og applikasjonslaget.

Nettverkslaget:

Nettverkslaget gir et grensesnitt mellom MAC laget og applikasjonslaget. Den er ansvarlig for ruting og å etablere forskjellige Zigbee nettverkstopologier, hovedsakelig stjerne, mesh og tre-topologien.

Når en koordinator prøver å etablere et Zigbee nettverk, settes en effektskanning i gang for å finne den beste radiokanalen for det nye nettverket. Når en kanal har blitt valgt, tilordner koordinatoren en

PAN-ID som vil bli brukt på alle enhetene som kobles til nettverket. PAN-ID er et 16-bit nummer som er brukt som en nettverks identifikator. [20]

Funksjonene til nettverkslaget er:

- Oppstart av nettverk.
- Tilordning av node adresser.
- Konfigurering av nye enheter.
- Gi sikker overføring.

Applikasjonslaget:

Applikasjonslaget i Zigbee arkitekturen består av de tre underlagene, Application Support Sub Layer (APS), applikasjonsrammeverket og Zigbee Device Object (ZDO).

Application Support Sub Layer (APS):

Dette laget er ansvarlig for filtrering av pakker for ende-enhetene og sjekke om pakker er blitt duplisert. APS er også involvert i vedlikehold av binding tabeller. Binding er forbindelsen mellom endepunktene på noden, til ett eller flere endepunkter på andre noder.

APS sin funksjon er:

- Vedlikehold av binding tabeller.
- Adresse definisjon, kartlegging og vedlikehold.
- Sikre at enheten kan kommunisere med hverandre.

Applikasjonsrammeverket:

Applikasjonsrammeverket avhenger av leverandøren som har valgt at spesifikke applikasjoner skal samhandle med Zigbee-protokollen. Applikasjonsrammeverket representer hvordan endepunkter implementeres, og hvordan dataforespørsler og databekreftelse utføres. [20]

Zigbee Device Object (ZDO):

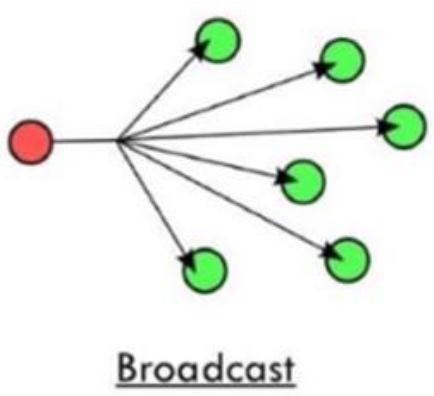
ZDO holder styr på tilstanden til Zigbee enheter på nettverket, og gir et grensesnitt til ZDP. ZDP er en spesialisert applikasjon som oppdager, konfigurerer og vedlikeholder Zigbee enheter og tjenester på nettverket. [22]

2.3.2 Kommunikasjon

Zigbee datapakker kan bli sendt enten som unicast eller kringkastings overføringer. Unicast overføringer ruter data fra en kilde-enhet til en destinasjonsenhet. Kringkastingsoverføringer er sendt til flere eller alle enheter i nettverket.

Kringkastingsoverføringer (Broadcast):

Kringkastingsoverføringer innenfor Zigbee protokollen er tiltenkt å bli forplantet gjennom hele nettverker sånn at alle noder mottar overføringen. For å oppnå dette, re-sender koordinatoren og all ruterne som mottar en kringkastingsoverføring, pakken på nytt tre ganger. Dette gjelder ikke når en ruter eller koordinator leverer en kringkastingsoverføring til en ende-enhet, denne overføringen er bare sendt én gang. [23]

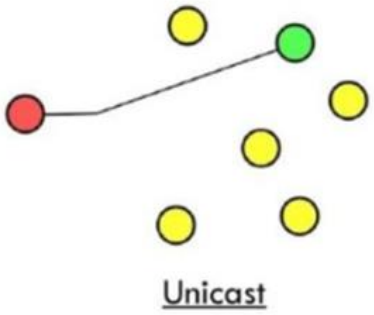


Figur 9 Broadcast

Hver node som sender kringkastingen, lager også en oppføring i en lokal kringkastings overførings tabell. Denne oppføringen holder styr på hver mottatt krinkastingspakke for å sikre pakkene ikke overføres i det uendelige.

For hver kringkastingsoverføring, reserverer Zigbee stakken bufferplass for en kopi av datapakken. Store krinkastingspakker krever mer bufferplass.

Unicast-overføringer:



Figur 10 Unicast

Unicast-overføringer sendes fra én kilde-enhet til en annen destinasjonsenhet. Destinasjonsenheten kan være en umiddelbar nabo til kilden, eller den kan være flere «hopp» unna.

2.3.3 Sikkerhet

Sikkerhetsarkitekturen i ZigBee utfyller sikkerhetstjenesten til IEEE 802.15.4-lagene (fundamentlaget). Dette er en modell basert på visse forutsetninger som er beskrevet nedenfor:

- Kommunikasjonen mellom forskjellige stakk lag på den samme enheten, er ikke kryptert.
- Kommunikasjonen mellom to enheter er kryptert og sikret.
- Maskinvaren er manipulasjonssikker.[24]

Tillitssenter

Tillitssenteret er en applikasjon som kjører på en enhet innenfor et ZigBee nettverk. Dens oppgave er å distribuere nøkler. Bare én tillitssenter kan eksistere per nettverk, og er ofte en koordinator. Alle medlemsnoder i nettverket gjenkjenner denne enheten som et tillitssenter. Tillitssenteret er ansvarlig for:

- Konfigurere og opprettholde nettverkssikkerhets policyer.
- Generering av nøkler ved å bruke en nøkkel etableringsprotokoll.[24]

Sikkerhetsmoduser

For Zigbee nettverk finnes det to ulike sikkerhetsmoduser, sentralisert og distribuert.

Sentralisert sikkerhetsmodus

Den sentraliserte sikkerhetsmodusen er startet av en koordinator, som har tillitssenter funksjonaliteten. I denne modusen er det bare tillitssenteret som kan gi nettverksnøkkelen til enheter som kobles til nettverket.

Distribuert sikkerhetsmodus

Distribuert sikkerhetsmodus er etablert av en ruter og har ingen sentralisert node som styrer sikkerheten i nettverket. I denne modusen er alle rutere like og der er ingen “tillit” mellom dem. Hver ruter kan gi nettverksnøkkelen til enheter som kobles til. Denne modusen er ofte mindre pålitelig enn den sentraliserte modusen. [21]

ZigBee sikkerhetsnøkler

ZigBee standard definerer to typer symmetriske nøkler, hver av dem, en 128-bit nøkkel brukt for kryptert kommunikasjon.

- **Nettverksnøkkel**

128-bit Nettverksnøkkel er brukt i kringkastingskommunikasjon. Hver node trenger en nettverksnøkkel for å kommunisere sikkert med andre enheter på nettverket. En enhet anskaffer en nettverksnøkkel første gang den blir med et nettverk. [21]

- **Koblingsnøkler:**

Koblingsnøkkel er en 128-bit unik nøkkel delt av to enheter, brukt i unicast kommunikasjon mellom enheter. En enhet kan få koblingsnøkkel enten gjennom nøkkeltransporttjeneste, eller den kan være forhånds installert.

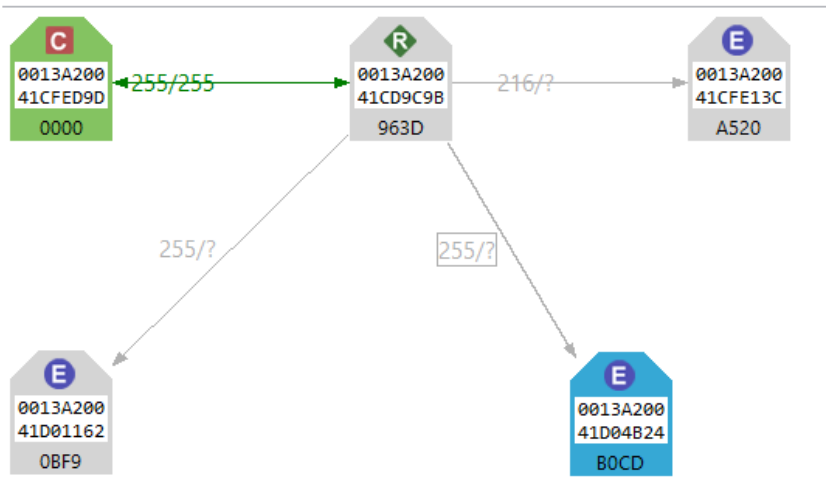
2.4 Utviklingskit

2.4.1 Digi Xbee 3 Zigbee Mesh Kit

For å sette opp et ZigBee mesh nettverk, teste ytelsesparametere og utvikle egne applikasjoner, trengte vi noen utviklingskit. Vi undersøkte hvilke kit som virket greie å bruke og som vi kunne utvikle til vårt formål. Til slutt var det to kit vi ville teste, Digi Xbee 3 Zigbee Mesh Kit og Nordic nRF52840/nRF5340. Det første kitet vi fikk hendene på var Digi Xbee 3 Zigbee Mesh Kit.

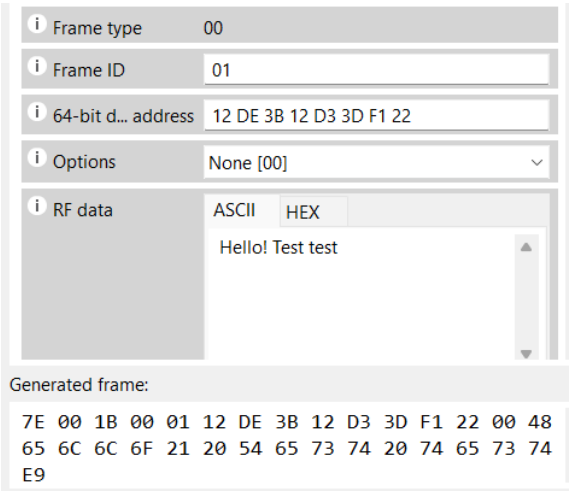
Figur 11 Digi Xbee 3 kit

Dette kitet var veldig brukervennlig, med et tydelig brukergrensesnitt. Her kunne vi enkelt konfigurere enhetene bare med å justere noen parametere, for eksempel bestemme hvor lenge de skulle sove. For å få enhetene på samme nettverk måtte vi bare forsikre oss om at de hadde lik PAN ID. Vi fikk ganske raskt satt opp et mesh nettverk, med en koordinator, en ruter og tre endeenheter.



Figur 12 Eksempel nettverksoppsett

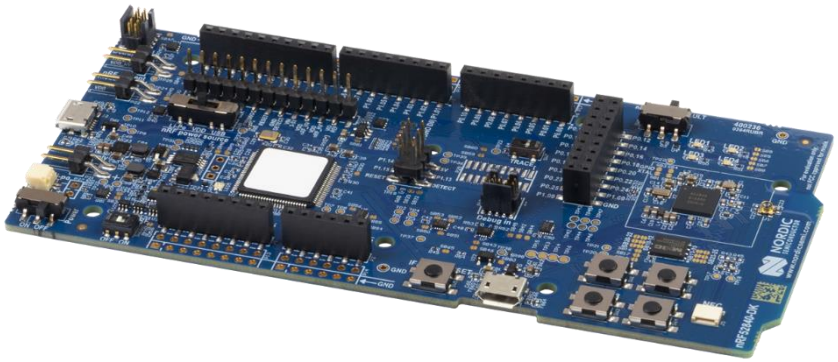
Etter å ha fått opp nettverket, kunne vi lage og sende pakker fra enten endeenhetene eller ruteren til koordinatoren som kunne da åpne og lese pakken. Dette ble gjort ved å bruke et verktøy kalt XBee API Frames Generator. Her måtte vi legge til 64-bit (MAC adressen) til mottaker enheten. Deretter skrive dataen som skulle sendes, enten i ASCII tegnsett eller i heksadesimal.



Figur 13 Frames generator

2.4.2 Nordic nRF52840/nRF5340

Det andre utviklings kittet vi brukte i prosjektet var Nordic nRF52840/nRF5340. Dette kittet var litt mer avanserte å bruke, der vi måtte også installere litt mer verktøy for å kunne konfigurere dem. Fordelen med kittet i forhold til Digi Xbee 3, er at det har flere muligheter til å modifisere og utvikle til vårt formål, der mye eksempelkode og dokumentasjon er tilgjengelig. I tillegg støtter kittet også Bluetooth mesh og Thread, noe som Digi Xbee 3 ikke gjør. I kapittel 3.5 beskrives det hvordan Nordic kitene brukes og utvikles.



Figur 14 Nordic nRF52840

2.4.3 Simula UIB

Kitene fra Nordic hadde vi egentlig tenkt å kjøpe og bestille, men her var vi så heldig å få låne Nordic kit som Simula UIB hadde tilgjengelig.

Figur 15 Simula UiB

Simula UIB driver med forsking og utdanning innen kryptografi og informasjonsteori. Selskapet eies av Simula Research Laboratory og Universitet i Bergen (UIB). Simula UIB har sitt utspring fra Selmersenteret ved Institutt for informatikk, som har satt søkelys på kodingsteori og kryptografi i flere tiår. Senteret har oppnådd høyest mulig karakter i Norges Forskningsråd i de to siste evalueringene av alle nasjonale informatikkgrupper. [32]

Vi hadde et lite møte med Simula UIB der vi beskrev hva prosjektet vårt gikk ut på og hva vi skulle bruke Nordic kittene til. De synes prosjektet virket spennende og var bare glad noen ville bruke kittene og laben deres. Med mye takknemmelighet, var det da på deres lab vi stort sett jobbet med prosjektet.

2.5 nRF Connect

2.5.1 Hva er nRF Connect?

nRF Connect for Desktop er en tverrplattform programvare for utviklingsapplikasjoner. Med nRF Connect SDK kan man begynne å utvikle lav-strøm trådløse applikasjoner med Nordic Semiconductor nRF52 og nRF53 Serie enheter. [33]

Figur 16 nRF Connect

SDK inneholder optimalisert Bluetooth Low Energy, Thread og Zigbee stakker. En rekke applikasjoner, flere sampler, referanse implementeringer, i tillegg til en full pakke med drivere for Nordic enheter.

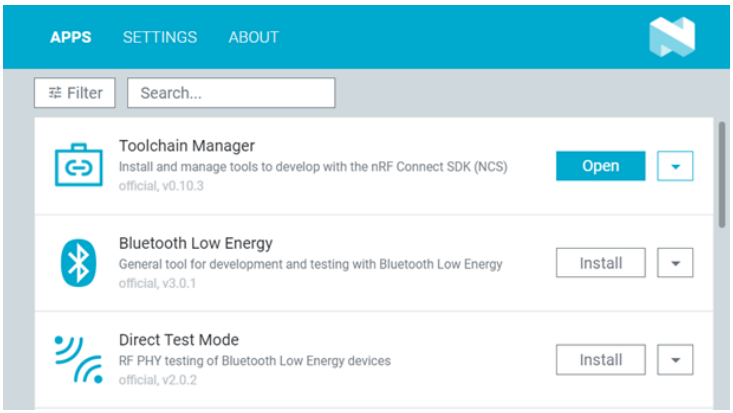
For vårt prosjekt har vi brukt nRF Connect for VS Code. nRF for VS Code lar oss utvikle, bygge og feil søke applikasjoner på nRF Connect SDK ved å bruke Visual Studio Code Integrated Development Environment (VS Code IDE). VS Code er en rask, tverrplattform og populær IDE som inneholder mange nyttig funksjoner. Blant annet et grensesnitt til kompilator, en rask feil søker (debugger), et grensesnitt til nRF Connect SDK og en seriell terminal. [34]

Programmeringsspråk

Programmeringsspråket brukt i VS Code er C. Vår erfaring med programmeringsspråket C var begrenset, men vi hadde brukt C litt i faget ELE102 Programmering og mikrokontrollere, der vi programmert Arduino. Når det gjaldt syntaks, datatyper og oppsett, var en del nytt for oss, mens også noe hadde gått i glemmeboken.

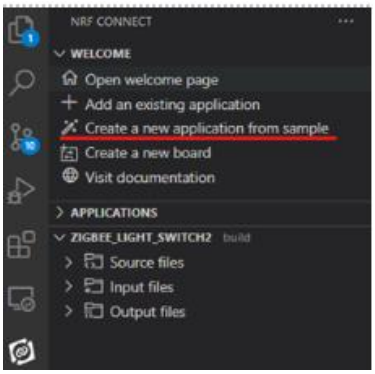
2.5.2 Hvordan fungerer nRf Connect?

For å bruke nRF Connect for VS Code er det noen verktøy som må installeres først. Det første verktøyet så må installeres er nRF Command Line Tools som man finner på Nordic Semiconductors nettside. Her finner man også nRF Connect for Desktop som også trengs å installere. Når disse to verktøyene er på plass, kan vi åpne nRF Connect for Desktop. Neste steg er så å trykke på «install» ved Toolchain Manager.



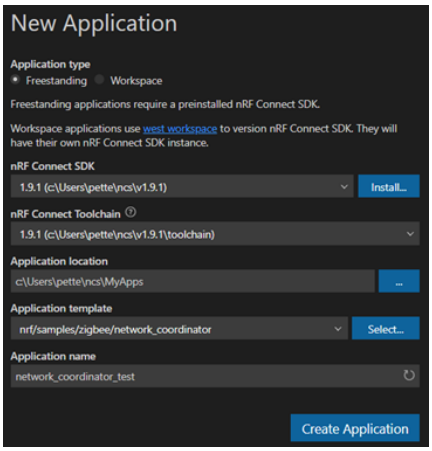
Figur 17 nRF command line tools

Deretter er det bare å trykke «open» og velge den nyeste versjonen av nRF Connect SDK. Hvis ikke man har VS Code fra før av må dette også installeres. Videre må VS Code åpnes gjennom nRF Connect, verktøyet vil da detektere at VS Code er installert og automatisk installere noen utviklingspakker for VS Code som man trenger. Når alt dette er på plass kan vi begynne å programmere og utvikle egne applikasjoner.



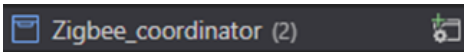
Figur 18 Hvordan lage ny applikasjon

For å lage en applikasjons i VS Code må man trykke på nRF Connect ikonet, og så trykk «Create a new application from sample».



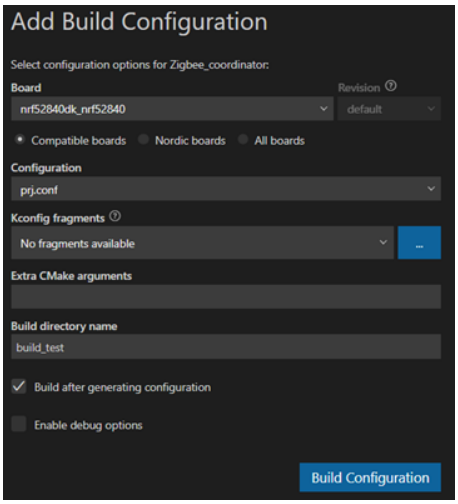
Figur 19 Ny applikasjon konfigurasjon

Da vil man få opp et vindu der vi kan velge hvor applikasjonen skal være lokalisert og hvilke templat applikasjonen skal bruke. For vårt prosjekt har vi for eksempel brukt Zigbee Network Coordinator, Zigbee Router og Zigbee Light Switch (fungerer som en endeenhet) og bygget videre på disse templatene.



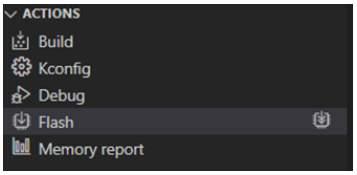
Figur 20 Add build configuration

Når man har så lagd applikasjonen er det viktig å trykke på «Add Build Configuration» under Application vinduet. Da vil enda et nytt vindu poppe opp, der man legger til hvilke kort man skal "bygge" til og om man vil aktivere feilsøkingsalternativer.



Figur 21 Build configuration, legge til kort

Når det er gjort, er alt på plass for å kunne kjøre applikasjonen/programmet. For å faktisk kjøre programmet må man under Actions vinduer trykke «Flash», som vil da «flashe» applikasjonen til chipset på kortet.



Figur 22 Flash

2.5.3 Implementering av ZigBee i nRF Connect

NRF Connect ZigBee protokoll bruker ZBOSS biblioteket, en tredjeparts forhåndskompilert ZigBee stakk som inkluderer alle obligatoriske funksjoner i ZigBee 3.0 spesifikasjonen. Stakket kommer med følgende funksjoner:

- Fullstendig implementasjon av ZigBee kjerne spesifikasjoner.
- Støtte for alle ZigBee enhet roller. (Koordinator, ruter og endeenhet).

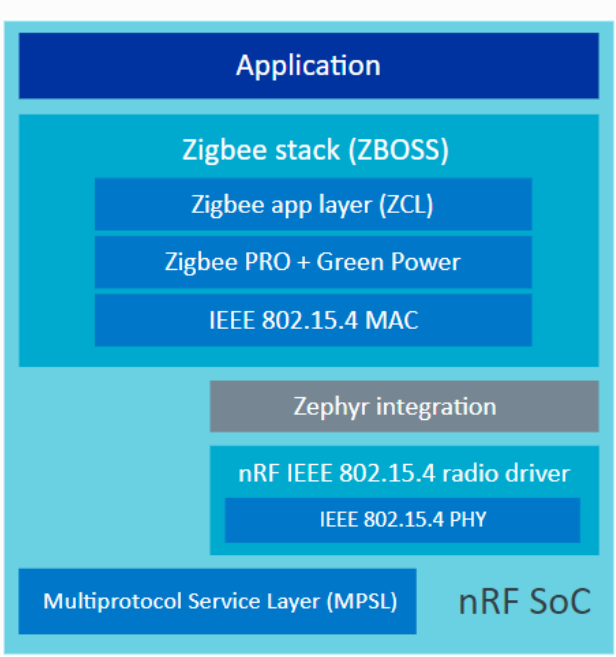
- ZigBee Cluster bibliotek (Et bibliotek som gir beskrivelser av enheter, for eksempel lysdimmer eller temperatursensor, og hvordan disse implementeres).

ZigBee plattformdesign

Det er flere plattformdesign som er mulig med ZigBee stakken på Nordic Semiconductor enheter. Designene går fra enkle applikasjoner som består av en enkel brikke som kjører én eller flere protokoller, til scenarioer der nRF fungerer som nettverksmedprosessor når applikasjonen kjører på mye kraftigere vertsprosessor.

System-on-Chip design

Enkel-chip løsning har kombinert IEEE 802.15.4 og prosessor, der ZigBee stakken og applikasjons laget kjører på den lokale prosessoren. Dette enkle designet kommer med en del nyttige fordeler. Den har den billigste kostanden, laveste strømforbruk og minste kompleksitet. Noen ulemper med dette enkle designet, er at applikasjon- og nettverksstakken deler flash- og RAM-plass. Dette gir mindre ressurser til applikasjonen. I tillegg er en ekstern flash nødvendig for å oppdatere firmware. Endeenheter og rutere bruker som oftest denne enkle-chip løsningen. I vårt prosjekt har vi jobbet med dette designet. [27]



Figur 23 Sytsem-on-chip design

Igangkjøring av enheter (Commissioning)

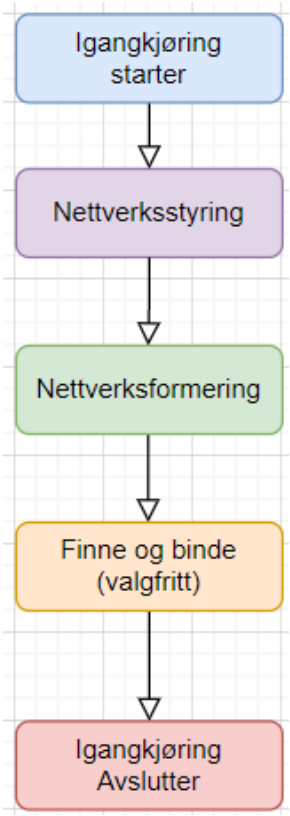
Igangkjøring er en prosedyre som tillater en ny ZigBee enhet til å bli med et nettverk. Enheten er konfigurert inn i nettverket, så den kan starte å kommunisere med andre noder.

Igangkjøringsprosedyren sikrer også at et nytt nettverk blir lagd, hvis det er ikke noe nettverk for enheten å bli med i.

Ifølge ZigBee spesifikasjonene er det flere måter igangkjøringsprosedyren kan starte:

- Automatisk etter initialisering av enheten.
- Med en brukerhandling, for eksempel ved å trykke på en knapp eller en kommando.

I ZBOSS stakken, kan igangkjøringen starte ved å bruke zboss_start() funksjonen. Zboss_start() initialiserer stakken, og starter igangkjøring prosedyren med standardmodusene (Nettverksstyring, nettverksdannelse, finne og binde). [28]

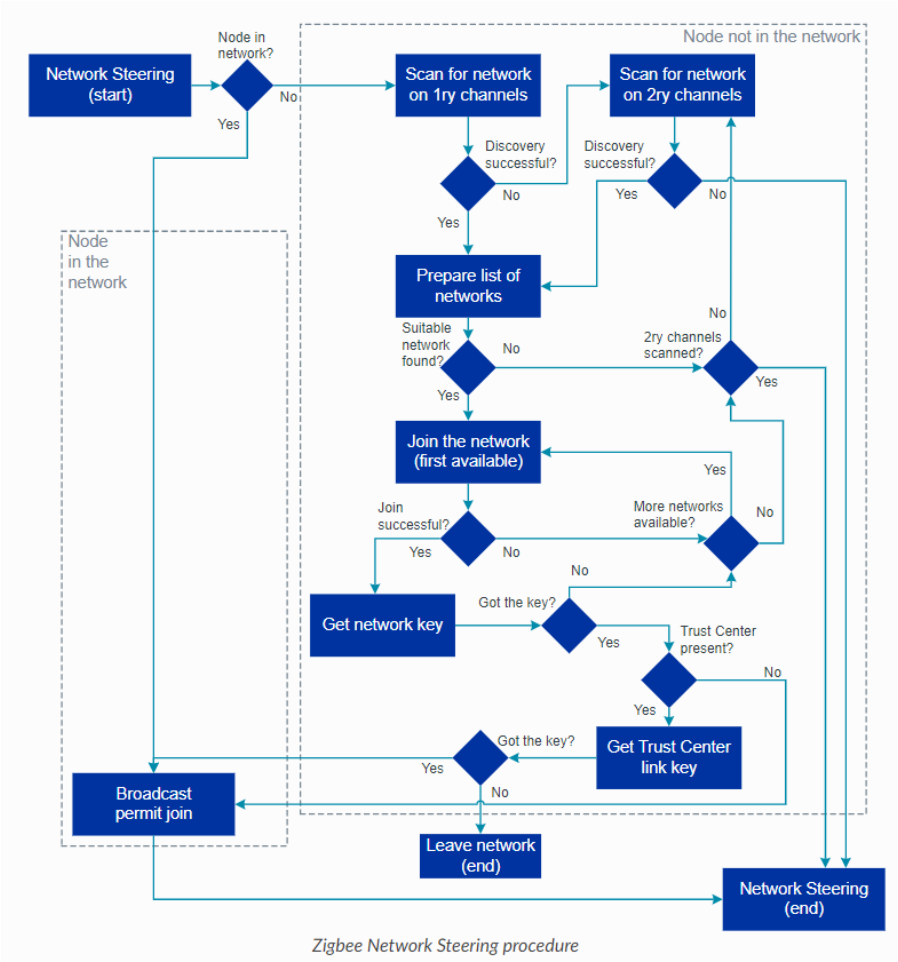


Figur 24 Igangkjøringsprosedyren

Denne prosedyren definerer den tiltenkte rekkefølgen for utførelsen av ZigBee igangkjørings modusene, som vist i figur 19. Igangkjøringsprosedyren er felles for alle enheter, men noen moduser er ikke utført for en gitt rolle. Nettverk formasjon er for eksempel ikke utført for endeenheter. Finne og binde er også valgfritt å ha med.

Nettverksstyring

Nettverksstyringsmodusen er brukt for å finne et åpent nettverk som en enhet kan bli med.



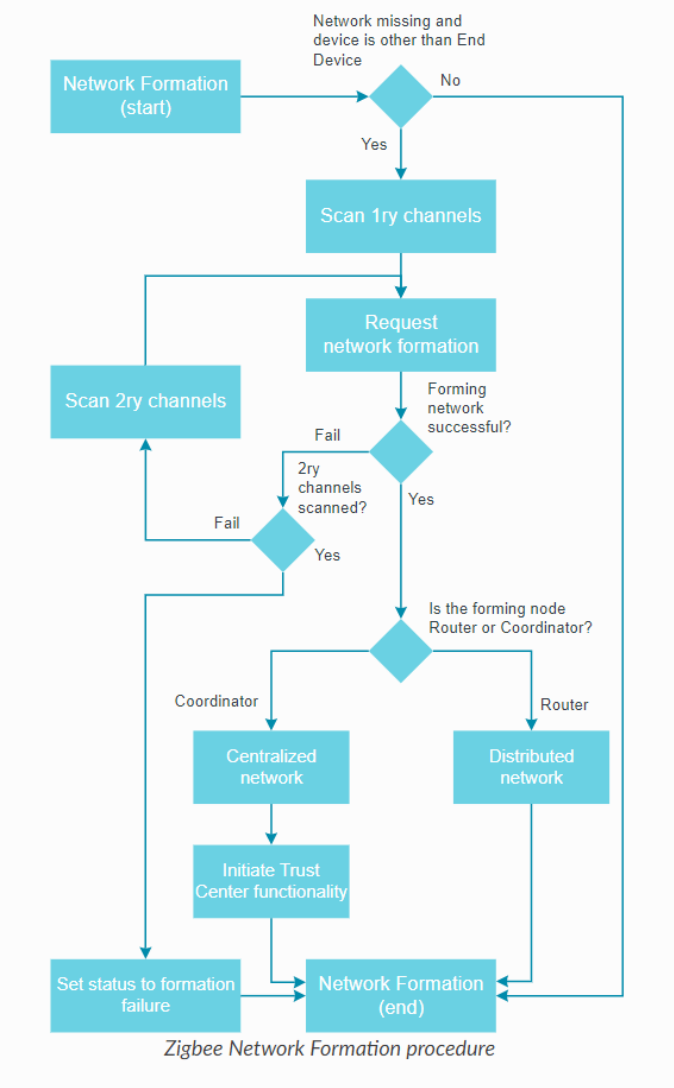
Figur 25 Nettverksstyrings prosedyren

Hvis noden allerede er et medlem av nettverket, vil den åpne nettverket og lukket den igjen etter en gitt tid (180 sekunder som standard). Hvis noden ikke er en del av et Zigbee nettverket, vil den skanne kanaler for å finne et åpent nettverk. Når den finner et passende nettverk, prøver den så å bli med i nettverket. [28]

Derimot hvis noden ikke finner et nettverk og den har enten ruter eller koordinator rollen, vil den prøve å etablere ett nytt nettverk. For ende-enheter vil igangkjøringsprosedyren stoppe her.

Nettverk formasjon

Nettverk formasjon modusen er bare tilgjengelig for rutere eller koordinatorer, fordi endeenheter ikke er i stand til å danne et nettverk.



Figur 26 Nettverksformasjon prosedyren

I nettverks formasjon modusen, har enheten ikke funnet noe nettverk å bli med i, så den prøver å forme et nytt nettverk.

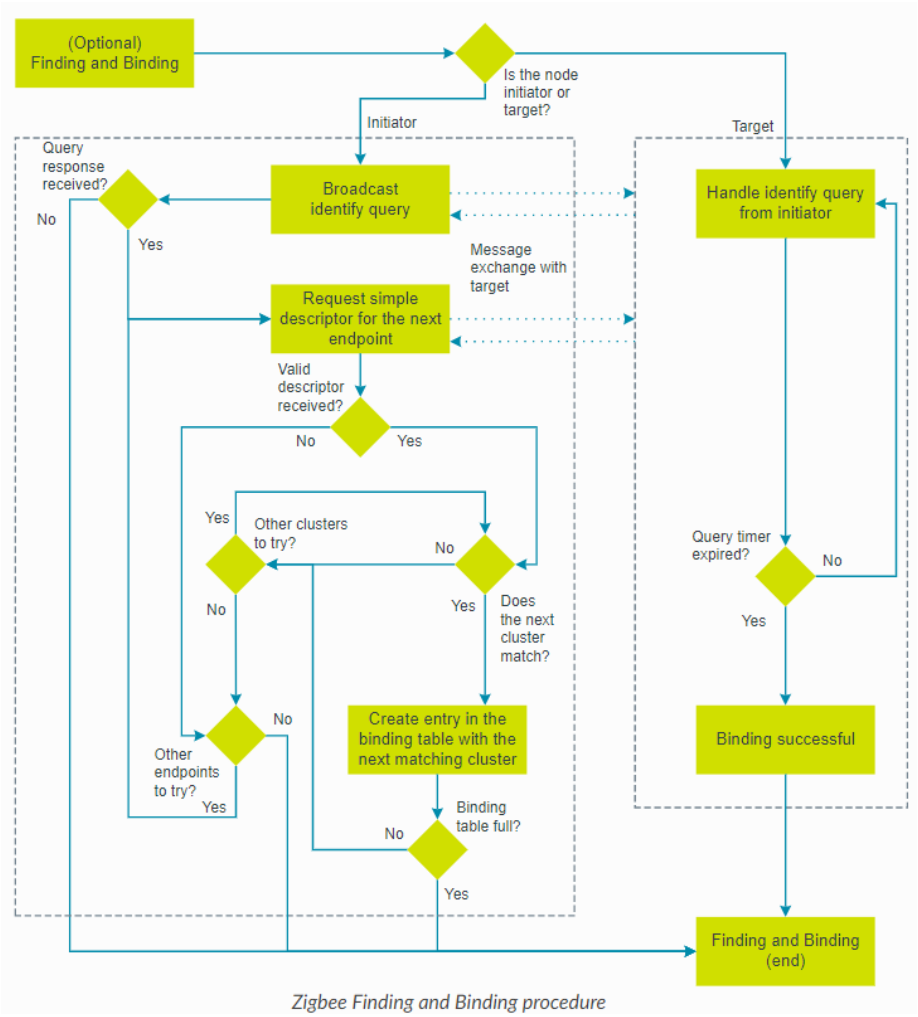
- En koordinator danner et sentralisert sikkerhetsnettverk.
- Ruter, hvis den er konfigurert for å danne et nettverk. Vil etablere et distribuert sikkerhetsnettverk. (Sentralisert- og distribuertnettverk er forklart i kapittel 3.3.3).

Hvis nettverksdannelsen ikke er vellykket, stopper igangkjøringsprosedyren her. [28]

Finne og binde

Etter at en node blir med nettverket, kan den gjennomgå en "Finne og binde" prosedyre. Med denne prosedyren, blir det etablert en forbindelse mellom tilsvarende endepunkter på to forskjellige noder automatisk. Som et resultat av dette, opprettes det nye oppføringer i bindingstabellen. Da kan nodene

kommunisere uten å bruke adressen til enhetene, men kan heller stole på forbindelser mellom endepunktene. [28] Denne prosedyren er valgfritt å ha med.



Figur 27 Zigbee finne og binde prosedyre

I vårt prosjekt er denne prosedyren startet gjennom en temp_bind_req() funksjon som vi har implementert.

Obligatorisk konfigurasjon for Zigbee i nRF Connect

Å definere ZigBee enhet roller for Zigbee applikasjoner er obligatorisk. Dette kan gjøres i VS Code på følgende måte:

- Ruter rolle: **CONFIG_ZIGBEE_ROLE_ROUTER = y**
- Endeenhet rolle: **CONFIG_ZIGBEE_ROLE_END_DEVICE = y**
- Koordinator rolle: **CONFIG_ZIGBEE_ROLE_COORDINATOR = y**

Ved å angi noen av disse alternativene aktiveres det respektive ZBOSS rolle biblioteket. Dette er nødvendig fordi endeenheter bruker andre biblioteker enn rutere og koordinatorer. [29]

Lage egne applikasjoner

For å lage egne applikasjoner, er dette gjort med å bruke et Zigbee templat og bygge videre på dette med ZCL clustere. Å legge til ZCL clustere til en applikasjon inkluderer å utvide Zigbee templatet med nye applikasjons clustere. Som standard, inkluderer templatet bare obligatoriske Zigbee clustere. Disse er grunnleggende og identifiserende cluster, og er nødvendig for å identifisere en enhet innenfor et Zigbee nettverk.

Cluster er en representasjon av en enkel funksjonalitet innen en Zigbee node. Hver cluster inkluderer attributter som er lagret i enhetens minne. For eksempel har temperatursensor clustere disse attributtene:

- Måle verdi
- Minimum måle verdi
- Maksimum måle verdi
- Toleranse

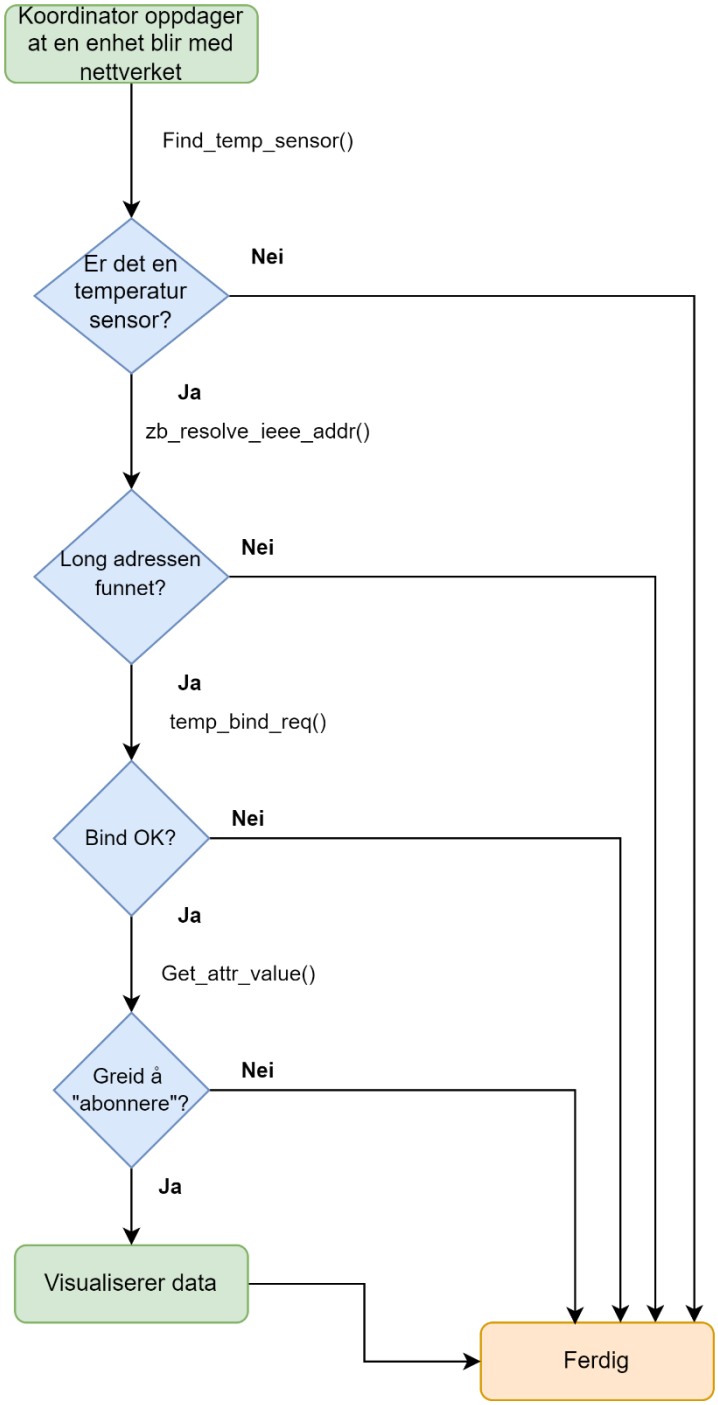
Hvert cluster inkluderer også kommandoer som kan bli brukt til å modifisere attributt verdiene eller lese statusen til enheten.[30] Clustere som passer for en enkel enhetstype, for eksempel en sensor, er organisert i en adresserbar beholder. Denne beholderen kalles et endepunkt.

3 Realisering av valgt løsning

3.1 Sluttprodukt og resultat

Sluttproduktet består av et mesh nettverk som inneholder én koordinator, én ruter og noen endeenheter. Koordinator vil hente sensordata fra endeenhetene som blir med nettverket eller som våkner fra “system off”. I “system off” slåes systemets kjernefunksjonalitet av og alle pågående oppgaver avsluttes. Dette nettverket kan også enkelt utvides med enda flere rutere og endeenheter, siden mesh nettverket er selvkonfigurerende.

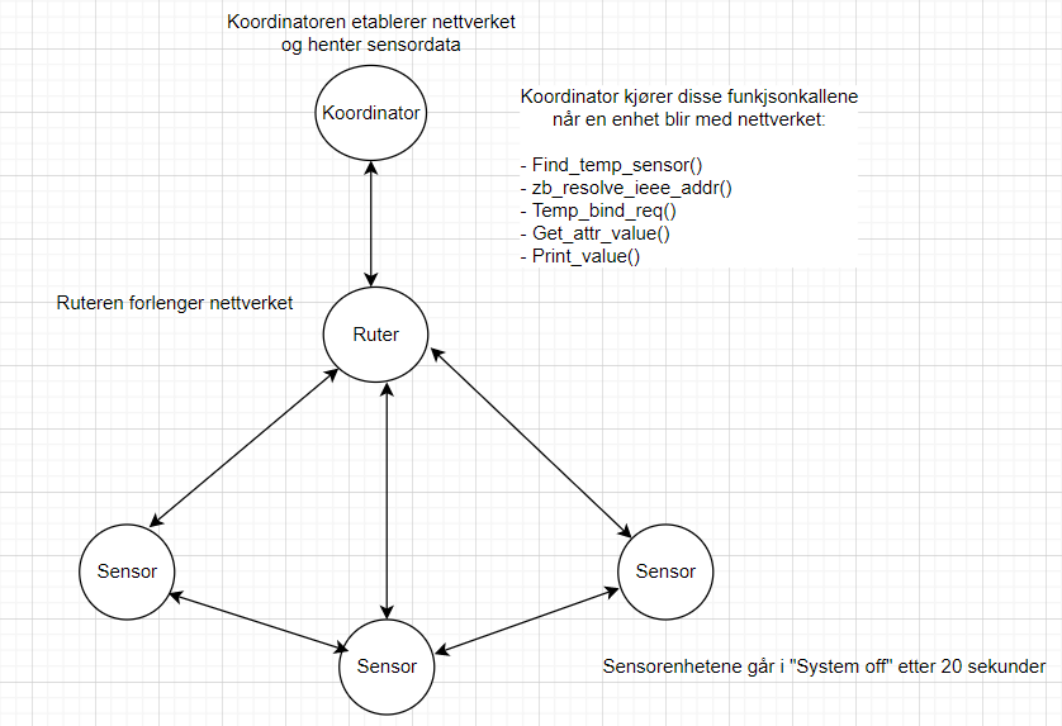
Når en enhet blir med nettverket, er koordinatorens oppgave å indentifisere om enheten er en temperatursensorenhet. Hvis en temperatursensor er funnet vil koordinatoren “binde” seg på enheten. Dette lager en bundet forbindelse mellom koordinatoren og tempretursensoren. Videre vil koordinatoren “abonnere” på enheten. Dette aktiverer rapportering på noden identifisert av koordinatoren, og koordinatoren kan da hente sensor data fra temperatursensorenheten. Koordinatoren visualiserer så dataen med et Print_value() funksjonskall.



Figur 28 Flytskjema for hvordan koordinatoren henter sensordata

Endeenhetene i nettverk oppfører seg som sensorer og henter temperaturverdier fra en temperatursensor som ligger i chipset på kortet. Endeenheten vil gå i “system off” 20 sekunder etter den blir med i nettverket, og vil “våkne” igjen ved å trykke en knapp på kortet. Når enheten er i “system

off" bruker den minimalt med strøm og vil da spare mye batterilevetid, noe som er essensielt for prosjektet. Etter at enheten blir "vekket", vil den gå i "system off" igjen etter nye 20 sekund. Ruteren i nettverket har egentlig bare én oppgave og det er å utvide nettverket.



Figur 29 Mesh-nettverket oppsett for vårt prosjekt

Sensordataen som koordinator henter, er visualisert som vist i figuren under. Dette blir vist i terminalen i Visual Studio Code.

```
Addr: 0x366b ep: 12 temp: 25.2 °C
I: Joining period extended.
I: Received EUI64 address for device 0x366b -> f4ce36f99c84f440.
Bind OK!
```

Figur 30 Sensordataen visualisert i VS Code

Her får koordinatoren informasjon om tempertursensorens "short"/16-bit adresse (0x366b), endepunkt (12) og temperaturen den måler (25.2 °C). I tillegg blir det vist at koordinator har greid å hente EUI64 (Extended Unique Identifier) adressen til temperatursensoren. Denne er nødvendig å hente for at koordinatoren kan binde seg på enheten. Koordinatoren får også beskjed at den har greid å binde seg på sensoren.

Løsning med Zigbee shell

Zigbee shell-biblioteket implementerer et sett av shell-kommandoer som gir muligheter til å kontrollere enhetens oppførsel og utføre operasjoner i Zigbee nettverket. For eksempel nettverksstyring, oppdage enheter og lese nettverksegenskapene. Med shell kommandoer forenkles testing og feilsøking (debugging) av et eksisterende nettverks. [31]

En ulempe med shell, er at alle kommandoer må skrives inn manuelt. Dette er litt tungvint i forhold til koden vi har implementert, der finne sensor, binde på sensoren og rapportere data, skal skje automatisk. For eksempel, er dette kommandoen som må skrives for at koordinatoren skal binde seg på en temperatursensor:

```
zdo bind on (h:sensorens EUI64 adresse d:sensorens endepunkt h:koordiantorens adresse
            d:koordinatorens endepunkt h:sensorens cluster id)
```

H: heksadesimal D: desimalverdi

Likevel er Zigbee shell en mulighet for koordinator å bruke, som kan da hente sensordata ved hjelp av kommandoer.

3.2 Testing

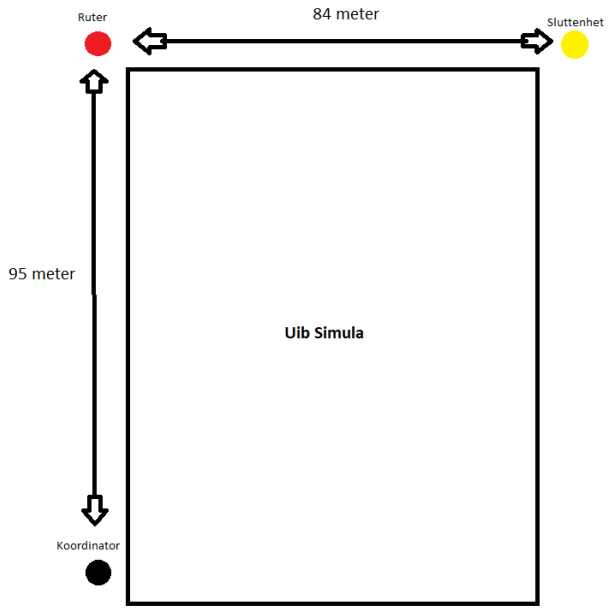
En av testene vi utførte, var å måle rekkevidden vi kunne oppnå mellom enhetene. Vi forventet at maks rekkevidde skulle være rundt 100m med fri sikt. For å måle rekkevidden brukte vi en Nikon MONARCH 3000 STABILIZED avstandsmåler. Denne kan måle avstander på opp mot 3000 meter og har et trådkors og en stabilisator som gjør det enkelt å sikte når en skal måle rekkevidden. Ved rekkevidder under 700 meter er feilmarginen +-0.5 meter. [17]

Figur 31 Avstandsmåler

Den første målingen ble gjort med bare 2 enheter for å se om vi kunne oppnå den angitte rekkevidden på 100 meter. Vi satte opp en koordinator og en ruter, og «pinget» ruteren fra koordinatoren. Dette gjorde vi regelmessig samtidig som vi bevegde ruteren lenger og lenger vekk fra koordinatoren. Når vi ikke lenger fikk respons på pinget, flyttet vi ruteren litt tilbake til vi fikk kontakt igjen. Vi målte med avstandsmåleren og fikk en distanse på 98 meter, noe som stemmer ganske bra overens med dokumentasjonen på utviklingskittet.

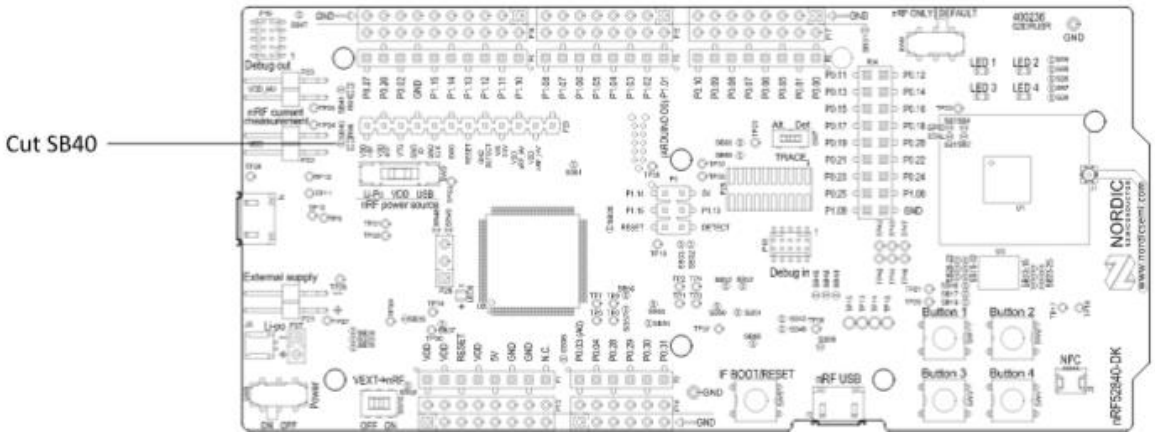
En annen ting vi testet var om mesh-nettverket var selvformende slik som det er dokumentert å være. Vi satte opp et nettverk med en koordinator, en ruter og en slutt-enhet, der slutt-enheten ved et trykk på en knapp kan slå av og på et led lys på ruteren.

For å teste om nettverket formet seg slik det skulle plasserte vi koordinatoren på langsiden av bygget der UiB Simula har lokale. Ruteren ble plassert på hjørnet av bygget, og slutt-enheten tok vi med oss videre på en annen side av bygget (se figur 32). Dette gjorde vi slik at koordinator og slutt-enhet ikke kunne ha direkte kontakt med hverandre, og vi var på denne måten sikre på at nettverket ble formet slik vi forventet. Mellom koordinator og ruter målte vi en distanse på 95 meter, og mellom ruter og slutt-enhet målte vi 84 meter.



Figur 32 Avstandsmåling visualisert

En annen test vi utførte var å måle strømtrekket for å få en indikasjon på hvor stort batteri som vil være nødvendig for å få ønsket levetid. For å kunne måle strømtrekket må man kutte over en loddebro. Dette gjør at strømleveransen til SoC (System on Chip) og resten av utviklingskittet blir splittet. En kan da koble til et amperemeter på P22 polene for å sette dette i serie med kortet og chipen. Det er anbefalt å bruke litium batteriet på kortet for å unngå eventuell støy fra USB under måling.



Figur 33 Loddebro som må kuttes (SB40)

Når amperemeteret var koblet til ville vi måle strømtrekket i forskjellige moduser. Den første modusen var ved kortet i vanlig modus, der det opererte som en ruter i et nettverk. Her varierte strømtrekket mellom 0.5mA - 2.6mA.

Den andre målingen vi utførte var ved kortet i såkalt "system on mode". Detter er en form for søvn der kortet er i en slags "idle" modus og vil våkne ved enhver hendelse. Ved kortet i "sytem on mode" målte vi et strømtrekk på 0.3mA - 0.7mA.

Den siste målingen, og kanskje den mest relevante var ved kortet i "system off mode". Dette er den dypeste formen for søvn kortet kan ha. I "system off" er system funksjonaliteten slått av og alle pågående oppgaver blir avsluttet. Eneste måten å vekke enheten fra "sytem off" er å enten binde en knapp på enheten til å vekke den, eller å ha et signal inn på GPIO'en.

I "system off" målte vi et strømtrekk på 0.4µA. Litium batteriet har en kapasitet på 500mAh. Et raskt regnestykke viser at om enheten er i "system off" hele tiden, vil batteriet ha en levetid på ca. 142 år og 8 måneder.

$$Levetid = \frac{Batteri\ kapasitet\ (mAh)}{Strømlast\ (mAh)} = \frac{500mAh}{0.0004mAh} = 1250000h \sim 142år, 8\ måneder$$

Dette regnestykket er selvfølgelig ikke reelt, da enheten må vekkes i faste intervaller for å måle og sende data. Om vi sier at enheten skal våkne en gang per time, og bruker 20 sekunder før den går i søvn modus, vil følgende regnestykke gjelde:

$$Gjennomsnittlig\ strømtrekk = \frac{(trekk1 \times tid1 + trekk2 \times tid2)}{(tid1 + tid2)}$$

Der trekk er i mA og tid er i timer.

Setter vi inn tallene fra målingene får vi følgende resultat:

$$Gjennomsnittlig\ strømtrekk = \frac{(2{,}7mA \times 0{,}00556h + 0{,}0004mA \times 0{,}9944h)}{(0{,}00556h + 0{,}9944h)} = 0{,}015398mA$$

Levetiden blir da: $\frac{500mAh}{0{,}015398mA} = 32471h \sim 3år\ og\ 8\ måneder$

Denne levetiden virker kanskje litt liten, men en ting å tenke på er at disse utrekningene er gjort med tanke på det lille klokkebatteriet som driver utviklingskortene. NSE vil nok bruke et større batteri, for å oppnå ønsket levetid.

3.3 Endringer og avvik

Under prosjektfasen ble det noen endringer og avvik i forhold til det som var planlagt i forprosjektet. Skjemategning ved utvikling av PCB var en oppgave som ble tatt vekk, dette vill nok North Sea Electronics jobbe med selv i ettertid. Oppdragsgiver ville også at sensordataen skulle bli visualisert ved bruk av en GUI (Grafisk brukergrensesnitt), denne deloppgaven ble også lagt bort, ettersom vi hadde litt dårlig tid mot slutten av prosjektfasen.

3.4 Videreutvikling etter prosjektet

Etter prosjektet vil NSE ta over koden vi har skrevet og ta inspirasjon eller forbedre denne til deres formål. De vil bruke samme chip som vi har brukt i våre utviklingskort, men på et eget kretskort. Kretskortet vil bli plassert inn i sensorpakken «fjellvåk» sammen med et batteri som kan drive enhetene i ønsket tid.

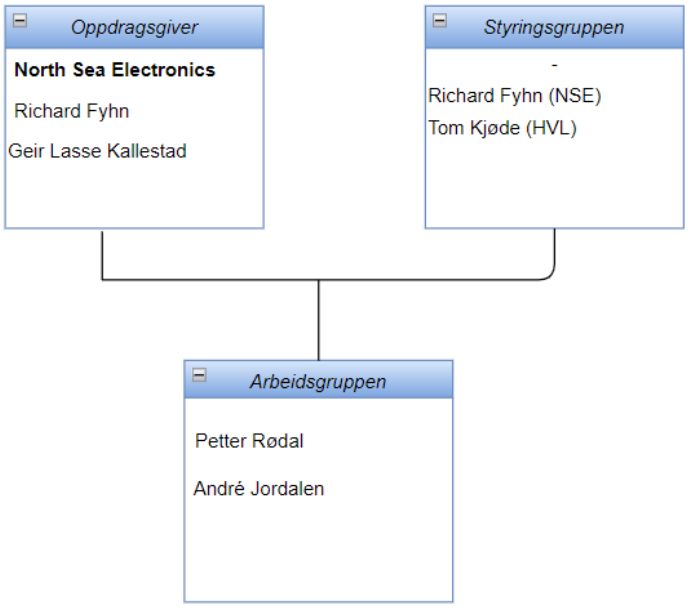
For å vekke enhetene fra «system off» mode vil NSE lage en klokke som bruker ekstremt lite strøm, og sender et signal på GPIO for å vekke enheten periodisk. Med andre ord vil dette eksterne signalet bestemme hvor ofte enhetene skal våkne og sende data.

Nettverket må også ha en basestasjon som tar imot sensordataene og sender disse videre til et kontrollrom. Kontrollrommet vil vise dataene sammen med identifikasjon på hvilken node det gjelder, for å kunne ha et bilde på hvor på fjellveggen/tunellen de aktuelle dataene gjelder.

4 Prosjektadministrasjon

4.1 Organisering

Organiseringen bestod av to nivåer der oppdragsgiver og styringsgruppe er øverst, under dem er arbeidsgruppen.



Figur 34 Organisasjonskart

Styringsgruppen

Styringsgruppen bestod av kontaktperson fra oppdragsgiver og intern veileder. De ga oss gode tips/veiledning og hadde ansvaret for å ta prosjektets viktige avgjørelser som revidere og godkjenne prosjektmålene.

Arbeidsgruppen

Arbeidsgruppen bestod av Petter Rødal og André Jordalen.

4.2 Prosjektarbeid og administrasjon

Oppstart prosjekt

Vi fikk tildelt oppgave og oppdragsgiver før jul. Vi tok initiativ til å møtes og starte arbeid tidlig i januar. Da hadde vi et oppstartsmøte med North Sea Electronics, her fikk vi mer detaljer rundt prosjektet og lagde en plan for arbeidet videre.

Styringsmøter

I prosjektperioden hadde arbeidsgruppen og styringsgruppen flere styringsmøter ved behov. Første styringsmøte skjedde etter arbeidsgruppen hadde valgt mesh-teknologi. Intervallet for styringsmøtene var litt varierende, der vi hadde flere styringsmøter i 2.halvdel av prosjektet enn i første halvdel. Tema som ofte ble diskutert i styringsmøtene var:

- Prosjektstatus

- Plan videre
- Sluttrapport
- Endring og avvik

Kommunikasjon med oppdragsgiver

Kommunikasjon med oppdragsgiver skjedde stort sett gjennom e-post. Både arbeidsgruppen og styringsgruppen tok initiativ til kommunikasjon. Misforståelser er en risikofaktor for prosjektet, så dette var viktig å unngå under kommunikasjonen.

Fordeling av oppgaver

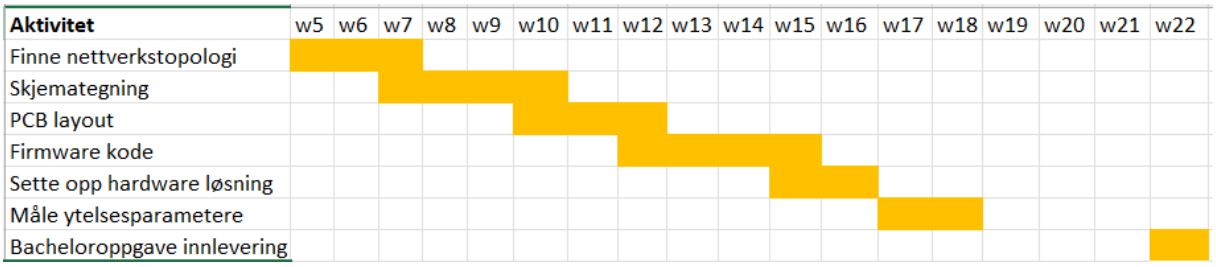
Det ble ofte lagd en oversikt over oppgaver som skulle gjøre i prosjektet. Gruppen arbeidet ofte samtidig med samme oppgave når det gjaldt praktiske oppgaver. Når det kom til rapportskriving fordelt vi litt mer.

Verktøy brukt for prosjektstyring

Microsoft Teams har blitt brukt til fildeling, i tillegg har vi hatt noen videomøter gjennom Teams. Vi har brukt Excel til å lage fremtidsplan (Gnatt diagram), regnskapsføring og tabeller. For å lage sekvensdiagrammer og flytskjema har vi brukt programmet Draw.io.

4.3 Gjennomføring i forhold til plan

Fremdriftsplanen som vist i diagrammet nedenfor ble laget under forprosjektet som ble levert i uke 4. Diagrammet viser en plan som var tenkt å følge, men det ble noen avvik. Noe av grunnet til dette, var at etter vi hadde valgt mesh-teknologi måtte vi nesten vente 2 uker får vi fikk det første utvikingskitet. Her tapte vi en del tid. Programmeringsdelen tok også opp mer tid enn forventet.



Figur 35 Gnatt-diagram, framdriftsplan fra forprosjektet

4.4 Generell prosjektevaluering

Måloppnåing

Prosjektet hadde flere hovedmål som vi stort sett fullførte, men det var fortsatt noen ting vi ikke fikk gjort. Vi skulle måle flere ytelsesparameter inkludert rekkevidde, datakapasitet og strømforbruk, her fikk vi aldri målt datakapasitet. Oppdragsgiver ville også ha skjemategninger ved utvikling av kretskort, noe som aldri ble påbegynt. Selv om de fleste hovedmålene ble oppnådd, trengs det videreutvikling før det kan eventuelt brukes i et kommersielt produkt.

Erfaring med prosjektgjennomføringen

Vi har tatt med oss flere nyttige erfaringer fra dette prosjektet. I denne typen prosjekt har vi erfart at samarbeid og kommunikasjon er sentralt. Det er viktig å være nøye med å forstå hva oppgaven og hovedmålene går ut på for å unngå misforståelser. I møtene er det også viktig å være tydelig når vi gir statusoppdateringer, sånn at oppdragsgiver forstår det vi har gjort.

For administrasjon er det viktig at gruppen blir enig om arbeidsfordeling og passer på at det blir framdrift. Vi må oppdatere hverandre om statusen på oppgavene, og avtale møte dersom en har lite å gjøre eller sitter fast. Med dette prosjektet har vi også fått litt innsikt i hvordan et prosjekt foregår i arbeidslivet, der vi har et ansvar for å få det fullført.

En annen erfaring vi har fått smake på, er at det er lett å henge seg opp i enkelte detaljer som til slutt ikke er så viktig for prosjektresultatet. Spesielt under programmering kunne vi gjøre mye arbeid på uviktige detaljer. Da kan en fort glemme hva som er viktig for helhetsresultatet og tape tid.

5 Konklusjon

I dette prosjektet var målet å indentifisere den beste mesh-teknologien som skal brukes for overvåking av fjellsikring. Deretter skulle vi sette opp et mesh-nettverket med den valgte teknologien og vise overføring av sensordata. For at mesh-teknologien skal være aktuell å bruke må den ha lavt strømforbruk, fordi de fleste nodene på nettverket skal være batteridrevet. I tillegg må den greie 50-100 meter rekkevidde mellom nodene.

Arbeidsgruppen konkluderte med at Zigbee teknologien vill være det beste for prosjektet. Mye på grunn av dens veldig lave strømforbruk, men også fordi den har godt utvalg av kommersielle enheter. Den har også noe kortere utviklingstid i forhold til mange av de andre teknologiene.

For å sette opp og teste Zigbee mesh-nettverk brukte vi utviklingskitene Nordic nRF52840/nRF5340. Vi kunne da måle ytelsesparametere som strømforbruk og rekkevidde, og dermed verifisere at denne teknologien kan brukes for overvåking av fjellsikring. Vi målte at rekkeviddene mellom nodene i mesh-nettverket var på 95 meter. Hvis vi la til en ruter fikk vi utvidet nettverket med 84 meter. Videre målte vi at nodene som er i “system-off” brukte µ0.4 amper, mens når nodene var “våken” brukte de 0.5-2.6 mA. Noe som betyr at hvis endenodene våkner én gang i timen for å gi sensordata, der noden kjører på et 500 mA/time litium batteri, vil den ha en levetid på ca. 3 år og 8 måneder. Her må det poengteres at endenodene våkner i 20 sekunder hver gang de gir sensordata, våkentiden kan nok reduseres noe for å få enda bedre levetid. Med disse målte ytelsesparameterne fikk vi verifisert at Zigbee teknologien kan fint brukes for dette prosjektet.

Arbeidsgruppen har også utviklet programkode for å få overført sensordata gjennom mesh-nettverket ved bruk av VS Code for nRF Connect. Her har vi modifisert at koordinatoren henter sensordata fra endenodene når nodene blir med nettverket, og når de våkner fra “system-off”. Endenodene er modifisert slik at de går i “system-off” 20 sekunder etter de har våknet. Nodene kan så bli vekket ved å trykke på en knapp. Vi hadde litt problemer med å få koordinatoren til å hente sensordata fra endenodene. Vi mistenker at det har noe med hvordan funksjons kallene er implementert i koden, der koordinator av og til vil prøve å hente sensordata før den for eksempel binder seg på endenoden. Dette burde ikke være avansert å fikse. Derimot hvis koordinatoren bruker Zigbee shell, greier den fint å hente data ved hjelp av noen kommandoer som må skrives inn manuelt. Til gjengjeld mistes mye av automatikken ved å bruke Zigbee shell.

Vi anbefaler av North Sea Electronics bruker Zigbee for mesh-nettverket de skal sette opp, og da tar over programkodene vi har implementert. NSE kan så fikse litt og videreutvikle koden til deres formål.

6 Referanser

[1] Wikipedia, «ZigBee», accessed 14.02.2022. Available: https://no.wikipedia.org/wiki/ZigBee

[2] Wikipedia, «ZigBee», accessed 14.02.2022. Available: https://en.wikipedia.org/wiki/Zigbee

[3] LinkedIN, «How to choose between ZigBee and BLE mesh for your IoT application», accessed 14.02.2022. Available: https://www.linkedin.com/pulse/how-choose-between-zigbee-ble-mesh-your-iot-rahul-yadav

[4] Digi, «What is ZigBee wireless mesh networking?», accessed 15.02.2022. Available: https://www.digi.com/solutions/by-technology/zigbee-wireless-standard

[5] Digi, «ZigBee vs Bluetooth choosing the right protocol», accessed 08.02.2022. Available: https://www.digi.com/blog/post/zigbee-vs-bluetooth-choosing-the-right-protocol

[6] Martin Wolley, «Deevolper Study Guid: An introduction to Bluetooth Mesh Networking», 29 June 2021, p.4-5, Acessed 05.02.2022.

[7] CSA, «ZigBee», accessed 15.02.2022. Available: https://csa-iot.org/all-solutions/zigbee/

[8] Wikipedia, «Bluetooth Mesh Networking», accessed 05.02.2022. Available: https://en.wikipedia.org/wiki/Bluetooth_mesh_networking

[9] Nordic Semiconductor, «Bluetooth Mesh», accessed 05.02.2022. Available: https://www.nordicsemi.com/Products/Bluetooth-mesh

[10] WSN & IoT, «Thread Network Technology – 01 introduction», Youtube, 21.12.2021. [Video]. Available: https://www.youtube.com/watch?v=r4ORGVOCvpQ

[11] RF Wireless World, «Advantages of Thread protocol | disadvantages of Thread protocol», accessed 16.02.2022. Available: https://www.nordicsemi.com/Products/Bluetooth-mesh

[12] Thread Group, «What is Thread», accessed 15.02.2022. Available: https://www.threadgroup.org/What-is-Thread/Thread-Benefits

[13] Digi-Key, «Comparing Low Power Wireless Technologies (Part 3)», accessed 15.02.2022. Available: https://www.digikey.no/no/articles/comparing-low-power-wireless-technologies-part-3

[14] Silicon Labs, «How long will my ZigBee enabled EM35x-based product operate on a battery?», accessed 23.02.2022. Available: https://community.silabs.com/s/article/how-long-will-my-zigbee-enabled-em35x-based-product-operate-on-a-battery-x?language=en_US

[15] Seyed Darroudi, Raul Caldera-Sànchez and Carles Gomez, Department of Network Engineering, Universitat Politècnica de Catalunya, «Bluetooth Mesh Energy Consumption: A Model», 25 January 2019, p.1, Acessed 23.02.2022.

[16] Thread Group, «Battery-Operated Devices white paper», accessed 23.02.2022. Available: https://www.threadgroup.org/Portals/0/documents/support/BatteryOperatedDevicesWhitePaper_656_2.pdf

[17] Nikon, «MONARCH 3000 stabilized», accessed 22.05.2022. Available: https://imaging.nikon.com/lineup/sportoptics/laser/monarch_3000_stabilized/spec.htm

[18] NSE, «about», accessed 20.04.2022. Available: https://nse.no/#about

[19] Proff, «North Sea Electronics AS», accessed 20.04.2022. Available: https://www.proff.no/selskap/north-sea-electronics-as/laksev%C3%A5g/tekniske-konsulenter/IGC3FAQ01OU/

[20] Laxmi Ahrit, electricalfundablog, «ZIGBEE Architecture (ZIGBEE Stack) – ALL Layers and its Functions», Accessed 10.05.2022. Available: https://electricalfundablog.com/zigbee-architecture-zigbee-stack-layers/

[21] Silicon Labs, «AN1233: Zigbee Security», Accessed 12.05.2022. Available: https://www.silabs.com/documents/public/application-notes/an1233-zigbee-security.pdf

[22] Electronics Hub, «What is Zigbee Technology? Architecture, Topologies and Applications» Accessed 17.05.20202. Available: https://www.electronicshub.org/zigbee-technology-architecture-applications/

[23] Digi, «Zigbee data transmission», Accessed 11.05.2022. Available: https://www.digi.com/resources/documentation/Digidocs/90002002/Reference/r_zb_data_transmission.htm

[24] Payatu, «IoT Security – Part 6 (ZigBee Security – 101)», Accessed 11.05.2022. Available: https://payatu.com/blog/dattatray/zigbee-security-101

[25] Wikipedia, «Bluetooth Low Energy», Accessed 20.05.2020. Available: https://en.wikipedia.org/wiki/Bluetooth_Low_Energy

[26] Bluetooth, «An Intro to Bluetooth Mes Part 2», Accessed 20.05.2022. Available: https://www.bluetooth.com/blog/an-intro-to-bluetooth-mesh-part2/

[27] Nordic Semiconductor, «Zigbee architectures», Accessed 13.05.2022. Available: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_zigbee_architectures.html

[28] Nordic Semiconductor, «Zigbee commissioning», Accessed 13.04.2022. Available: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_zigbee_commissioning.html

[29] Nordic Semiconductor, «Configuring Zigbee in nRF Connect SDK», Accessed 16.04.2022. Available: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_zigbee_configuring.html

[30] Nordic Semiconductor, «Adding ZCL clusters to application», Accessed 17.04.2022. Available: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_zigbee_adding_clusters.html

[31] Nordic Semiconductor, «Zigbee shell», Accessed 29.05.2022. Available: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/1.5.0/nrf/libraries/zigbee/lib_zigbee_shell.html#zdo-bind-on

[32] Simula, «Simula UiB», Accessed 14.05.2022. Available: https://www.simula.no/research/projects/simula-uib

[33] Nordic Semiconductor, «Welcome to the nRf Connect SDK!», Accessed 18.05.2022. Available: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/index.html

[34] Nordic Semiconductor, «nRf Connect for VS Code», Accessed 18.05.2022. Available: https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-VS-Code

Vedlegg

1. **Programkode**
   - **1.1 Zigbee_coordinator**
   - **1.2 Zigbee_light_bulb(Fungerer som ruter)**
   - **1.3 Zigbee_light_switch2(Fungerer som temperatursensor)**