Høgskulen på Vestlandet

BACHELOROPPGAVE

Forskning på bruk av LoRaWAN i tidskritiske applikasjoner

Research on the use of LoRaWAN in time-critical applications

Jonas Bakken Engen
Endre Svenningsen Nordtvedt
Jonas Vestbø

Dataingeniør
Fakultet for teknologi, miljø- og samfunnsvitenskap /
Institutt for datateknologi, elektroteknologi og realfag /
Dataingeniør
Volker Stolz

Innleveringsdato: 13.05.2024

Høgskulen på Vestlandet

Fakultet for teknologi, miljø- og samfunnsvitenskap
Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: **Forskning på bruk av LoRaWAN i tidskritiske applikasjoner**	*Dato:* ***13.05.2024***
Forfatter(e): **Jonas B. Engen, Endre S. Nordtvedt, Jonas Vestbø**	*Antall sider u/vedlegg:* ***93*** *Antall sider m/vedlegg:* ***168***
Studieretning: **Dataingeniør**	*Antall disketter/CD-er:* **0**
Kontaktperson ved studieretning: **Volker Stolz**	*Gradering:* **Ingen**
Merknader: **Ingen**	

Oppdragsgiver: **Eidsiva Bredbånd AS**	*Oppdragsgivers referanse:* **Ingen**
Oppdragsgivers kontaktperson: **Kjartan Mikkelsen / Steinar Tofte**	*Telefon:* **489 90 013**

Sammendrag: **Prosjektet går ut på å undersøke ved hjelp av litteratursøk og eksperimentering om LoRaWAN-protokollen er pålitelig nok til å benyttes i tidskritiske applikasjoner.**

Stikkord:

LoRaWAN	**Tidskritisk**	**Eksperimentering**

Høgskulen på Vestlandet, Fakultet for teknologi, miljø- og samfunnsvitenskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

Forord

Prosjektet er gjennomført i studieprogrammet Dataingeniør ved Høgskulen på Vestlandet Campus Bergen. Prosjektet ble gjennomført våren 2024 med prosjektstart 15.01 og rapport innlevering 13.05. Alt arbeid i forbindelse med dette prosjektet er utført av Jonas B. Engen, Endre S. Nordtvedt og Jonas Vestbø.

I forbindelse med et av gruppens medlemmer sin sommerjobb for oppdragsgiver, fikk gruppen tilbud om å skrive bacheloroppgave om teknologien LoRaWAN som oppdragsgiver bruker i sin Internet of Things (IoT) virksomhet. Dette forslaget virket interessant for alle gruppens medlemmer. Med frihet til å velge retning innenfor tema og spennende tematikk med mange interessante fokusområder, ble gruppen enig i å starte på dette prosjektet.

Vi ønsker å takke oppdragsgiver Eidsiva Bredbånd og våre kontaktpersoner der, Kjartan Mikkelsen og Steinar Tofte for god kommunikasjon og nyttige tilbakemeldinger gjennom hele prosjektet. Vi ønsker også å takke vår interne veileder ved HVL Volker Stolz for god hjelp og veiledning. Til slutt vil vi også takke Carsten Gunnar Helgesen for god hjelp med undervisning og tips til utførelse av et bachelorprosjekt.

Referanseliste

Ordliste

Ord	**Forklaring**
API	Et grensesnitt i en programvare som gjør at spesifikke deler av denne kan aktiveres fra en annen programvare.(Wikipedia,2024)
Branch	Ny avgrenset del i repository til å utvikle på
Bitrate	Bitrate er et begrep innen telekommunikasjon og digitale medier, det angir hvor mange bits som overføres eller behandles per tidsenhet.(Wikipedia, 2022)
CAD Computer Aided Design)	3D-modellering eller teknisk tegning
CD (Continuous development)	Kontinuerlig leveranse av produkt
CI (Continuous integration)	Kontinuerlig implementasjon av nye funksjoner
Downlink	Er at det sendes data ut til en sensor via gateway
Go / Golang	Programmeringsspråk
Hoste / Hosting	Å hoste er det samme som å lagre en webapplikasjon på en maskin og gjøre den tilgjengelig for brukere over internett.
HTTP (Hypertext Transfer Protocol)	Protokoll som benyttes for å overføre informasjon over internett
IDE (Integrated Development Environment)	Programvare som brukes for å skrive og lage annen programvare

IIoT (Industrial-IoT)	IoT i industrielle prosesser og bruksområder
IoT (Internet of Things)	Gjenstander utstyrt med elektronikk, programvare, sensorer, aktuatorer og nettverk som gjør gjenstandene i stand til å koble seg til hverandre og utveksle data. (Wikipedia, 2024)
IR (Infrared)	Elektromagnetisk stråling som ligger mellom synlig lys og mikrobølger. Kalles også varmestråling
JSON (JavaScript Object Notation)	JSON er en enkel tekstbasert standard for å formatere dokumenter (meldinger) som brukes for datautveksling. (Wikipedia, 2024)
KI	Kunstig intelligens
Kontinuerlig datastrøm	En kanal som hele tiden sender data
Merge	Teknikk for å kombinere arbeid gjort i forskjellige brancher
Microservice	En micro service en liten web service som er spesialisert til å gjøre én ting(UIO, 2017)
MQTT	Nettverksprotokoll for maskin-til-maskin kommunikasjon
Python	Programmeringsspråk
REST	REST – REpresentational State Transfer – er en programvarearkitektur som sikrer løse koblinger mellom to tjenester i en integrasjon(NTNU, n.d)

POCSAG-paging	Kommunikasjonsprotokoll som brukes for dataoverføring til personsøkere
Repository	Et sted på Github du kan lagre kode, filer og ha revisjonshistorie for hver fil
RSSI (Received Signal Strength Indicator)	Styrken på mottatt signal
Scrum	Scrum er et smidig prosessrammeverk utviklet for å støtte kompleks produktutvikling(Ramsøy, 2022)
Sprint	En kort tidsperiode hvor gruppen ønsker å få til en spesifikk del av et prosjekt
SNR (Signal-to-noise ratio)	Forholdet mellom signalnivå og støynivå
SF (Spreading factor)	Spreading factor (Spredningsfaktor SF) er et mål på hvordan spredningen av et signal er i et gitt område (angis som et heltall fra 7 til 12). Høy SF betyr større dekningsområde, lavere signalstyrke og redusert båndbredde. Tilsvarende betyr lav SF mindre dekningsområde, høyere signalstyrke og økt båndbredde.
Smart cities	Smartby (Smarte løsninger i urbane lokasjoner, for eksempel: gatelys system og lignende)
Uplink	Er at en sensor sender data opp til en gateway
WAF (Web Application Firewall)	Nettapplikasjonsbrannmur for å filtrere, overvåke og blokkere HTTP-trafikk.

Høgskulen på Vestlandet

Fakultet for teknologi, miljø- og samfunnsvitenskap
Institutt for datateknologi, elektroteknologi og realfag

1 Innledning

1.1 Kontekst

Oppdragsgiver Eidsiva er et energi - og telekom konsern som leverer samfunnskritisk infrastruktur og tjenester til to millioner mennesker i dag. Eidsiva har siden 1894 elektrifisert og utviklet samfunnet, og i dag er det 1300 mennesker som jobber med utviklingen. Konsernet Eidsiva består av Elvia, Eidsiva Vekst, Eidsiva Bioenergi, Eidsiva Bredbånd og morselskapet Eidsiva Energi (Eidsiva-a, n.d).

Oppdragsgiver i dette prosjektet er Eidsiva Bredbånd som til daglig jobber med å bygge, selge og drifte høyhastighets bredbånd og fiber i hovedsak til privatkunder i Innlandet og til bedrifter på Østlandet (Eidsiva-b, n.d).

LoRaWAN er en teknologi som benytter seg av lav båndbredde og bruker lite energi uten at dette går utover rekkevidden til signalet. Denne teknologien er nyttig steder med vanskelige dekningsforhold som for eksempel om sensorene ligger under bakken eller i murbygninger. Sensorer som brukes med LoRaWan- teknologien er utstyrt med radiosendere som benytter lav bitrate og har med det veldig lavt energiforbruk. Dette lave energiforbruket gir batteriene i sensorene mange års levetid (Altibox, n.d).

1.2 Motivasjon

Eidsiva benytter LoRaWAN protokollen i dag i noen få prosjekter. Uten å nevne navn på kunder, er protokollen brukt med sensorer som måler temperatur i vann, høyde på vann i demning og andre lignende bruksområder. Oppdragsgiver ønsker å se mer på protokollen og vil undersøke om protokollen er pålitelig nok til å bli benyttet i mer tidskritiske applikasjoner. Dette vil kunne øke oppdragsgiverens bruk og salg av teknologien som vil være med på å generere flere arbeidsoppgaver og kunder.

1.3 Prosjekteier

Prosjekteier er Eidsiva Bredbånd som til daglig jobber med kontinuerlig utvikling og innovasjon, og i tillegg tilbyr lynrask fiber. Ola Børke i Eidsiva sier at det fremdeles er for mange som ikke er tilknyttet fiberbånd, og dette kan være med på å skape et digitalt klasseskille. Børke forklarer at alle i Norge i praksis bør være direkte tilknyttet det digitale fibernettet i Norge innen få år. Dette jobber Eidsiva med i dag (Eidsiva-c, n.d).

1.4 Problembeskrivelse og mål

LoRaWAN er et lavt-forbruk, wide area nettverksprotokoll bygd på topp av LoRa radio moduleringen. Bruk av LoRaWAN vokser innenfor bruk i "smartbyer", grunnen til dette er at LoRaWAN er billig, har toveiskommunikasjon og har veldig lavt strømforbruk (TekTelic, 2021). I land som Sverige og England er bruk av LoRaWAN mer utbredt. Eidsiva benytter teknologien i Norge innenfor noen områder, men ønsker å finne ut om bruk av LoRaWAN protokoll er pålitelig nok til å brukes i tidskritiske applikasjoner. Basert på dette ble følgende problemstilling valgt:

Er LoRaWAN egnet til bruk innenfor tidskritiske applikasjoner?

Prosjektets overordnede mål er å forske på LoRaWAN og finne begrensninger og fordeler ved bruk av LoRaWAN innenfor tidskritiske applikasjoner og dermed konkludere om det er gunstig å bruke LoRaWAN innenfor denne typen applikasjoner.

For å nå det overordnete målet er det satt tre delmål for prosjektet:

1. Finne data på LoRaWAN protokollen sin ytelse med tanke på spredningsfaktor, signalstyrke og signal-to-noise ratio.
2. Finne ut hva som gir gode signal og hva som er med å forstyrre signal (Felteksperiment)
3. Utforske om datapakker går tapt på veien og eventuelt finne årsaker til hvorfor datapakker går tapt og om det finnes sammenheng mellom tapte datapakker og signalstyrke.

For å finne ut om LoRaWAN er egnet til bruk innenfor tidskritiske applikasjoner har gruppen utviklet to forskningsspørsmål som er relevante for prosjektet:

1. Hva er fordelene og ulempene ved bruk av LoRaWAN innenfor tidskritiske applikasjoner?
2. Er LoRaWAN protokollen pålitelig?

LoRaWAN er en protokoll som har lang rekkevidde med lav båndbredde. Dersom LoRaWAN oppfyller krav og har god nok ytelse til å kunne bli brukt innenfor tidskritiske applikasjoner, åpner dette opp for mange muligheter. En av de største grunnene til at det er ønskelig å bruke LoRaWAN sensorer dersom det er mulig er på grunn av deres batterilevetid.

> "*LoRaWAN lar deg sende data informasjon over lange avstander og bruker veldig lite strøm. Dette lar LoRaWAN enheter operere på batteri i opp til 10 år eller lengr*e" (The Things Industries, 2022, gruppemedlem sin oversettelse).

En annen fordel med LoRaWAN er at de har dekning selv om de blir plassert innendørs og at de er lette å oppdatere over luften. På bakgrunn av dette, dersom LoRaWAN møter krav for å kunne bli anvendt til bruk i tidskritiske applikasjoner, vil det gjøre LoRaWAN til et veldig attraktivt alternativ til bruk innenfor IoT. Det betyr at LoRaWAN kan benyttes bredere innenfor helsesektoren, logistikk eller gårdsbruk.

Samfunnsproblemer som IoT løser er effektivisering av ulike prosesser som for eksempel utrykninger, overvåkninger og alarmer som ville krevd mer manuelt arbeid. Hvorfor er LoRaWAN valgt til disse problemene? LoRaWAN er valgt fordi det kan brukes på steder der det er vanskelig å komme til, sensorene bruker lite batteri som gjør at levetiden på sensorene er mange år, og LoRaWAN er en protokoll som kan sende datapakker over veldig lange avstander.

1.5 Oppbygging av rapport

Denne rapporten er delt inn i 7 hovedkapittel med flere underkapittel. Kapittel 2 inneholder praktisk bakgrunn, initielle krav, initiell idè, ulike avgrensninger for prosjektet og ressurser benyttet for løsningen på oppgaven. I kapittel 3 vil forskjellige alternative løsninger diskuteres og drøftes. Prosjektmetodikk er også tema i dette kapittelet, her blir utviklingsmetode og metode for litteratursøk diskutert noe grundigere. Videre i kapittel 4 vil det bli presentert en detaljert beskrivelse av design, utvikling, eksperimentering og løsning. Kapittel 5 går inn på resultater og prosjektgjennomføring samt evalueringsmetode og evalueringsresultat. Kapittel 6 tar for seg diskusjon om hvordan resultater har løst problemstillingen, delmål og krav fra oppdragsgiver. I tillegg til dette blir også prosjektets gjennomføring også diskutert. Til slutt konkluderer kapittel 7, innhold, resultater samt beskrivelse om mulig videre arbeid. Avsluttende vil kapittel 8 inneholde litteraturliste og kapittel 9 inneholder vedlegg.

2 Prosjektbeskrivelse

Dette kapittelet gir en dypere beskrivelse av problemet, med problemstilling og hovedmål for prosjektet, inkludert delmål og forskningsspørsmål. Videre i kapittelet kommer en innføring i praktisk bakgrunn for prosjektet. Dette inneholder tidligere arbeid, initielle krav og en initiell løsnings-idé. Deretter forklares også avgrensninger i prosjektet, og nødvendige ressurser. Avslutningsvis gis en oversikt over litteratur som gjelder for problemstillingen.

2.1 Praktisk bakgrunn

2.1.1 Tidligere/pågående arbeid

Dette bachelorprosjektet er tilknyttet et pågående arbeid i Eidsiva Bredbånd som går ut på å se potensialet i LoRaWAN og utvide LoRaWAN nettverket i Norge. Dette prosjektet skiller seg fra arbeidet til Eidsiva ved at det blir utforsket dypere i egenskapene til LoRaWAN-teknologien enn det Eidsiva har gjort. I dag har Eidsiva kunder som bruker LoRaWAN fra dem i ulike områder, som blant annet temperaturmåling i vann og luft, inn- og utpasseringer på et punkt, bevegelse, tilstedeværelse via IR og avstandsmåling til vannflate.

2.1.2 Initielle krav

Oppdragsgiveren har følgende forventninger:

- “Undersøke om LoRaWAN kan brukes i tidskritiske applikasjoner med tanke på pakketap, signalstyrke og pålitelighet”
- “Se på endringer / forbedringer for å redusere eventuelle problemer. Det mest åpenbare er å sikre god nok radiodekning der tjenesten skal leveres.”

Basert på disse forventningene fra oppdragsgiver og forskningsspørsmålene gitt i punkt 1.4 er dette mål prosjektet ønsker å oppfylle.

2.1.3 Initiell løsnings-idé

Løsningen gruppen velger for å løse oppgaven og kunne svare på spørsmålet: Er LoRaWAN pålitelig nok til å brukes i tidskritiske applikasjoner? er delt opp i tre deler. Første del som omhandler litteratursøk er å se om det er gjort tidligere arbeid som underbygger eller motbeviser forskningsspørsmålet. Her ønsker gruppen å finne artikler som ikke bare kan brukes til inspirasjon, men som også kan bygges videre på.

Andre del er å eksperimentere med signalene gruppen har tilgang til via oppdragsgiverens kontinuerlig datastrøm. Her skal gruppen i hovedsak lage to python script, hvor hvert av scriptene har hver sin oppgave. Første script skal hente ut nødvendig informasjon om signalstyrken og sammenligne disse verdiene med LoRaWAN sine grenseverdier. Her vil gruppen se hvor stor del av alle signalene gruppen har tilgang til som kan defineres som et bra- eller brukenes signal. Gruppen ønsker også å visualisere resultatet i en graf slik en kan se tydelig hvilke signal som ikke er gode nok til å tolkes av mottaker.
Det planlegges også å lage et script som overvåker om datapakker går tapt. Dette ønsker gruppen å gjøre ved å lage et script som sjekker intervallet på hvor ofte signal kommer inn, og deretter sjekker om det er "hull" i intervallet, altså at en datapakke har gått tapt på veien. Scriptene har som oppgave å underbygge eller motbevise om LoRaWAN er pålitelig og at signal kommer frem til mottaker som det skal og med riktig signalstyrke.

Tredje delen gruppen ønsker å gjennomføre for å enda grundigere sjekke påliteligheten til LoRaWAN er å faktisk måle signalstyrken med en Field tester i Bergen sentrum. Da gruppen befinner seg i Bergen blir dette eksperimentet avgrenset til Bergen og byens LoRaWAN gateways. Eksperimentet går ut på å sjekke hvordan avstand fra sender, bygninger og fjell (garasjeanlegg og underetasjer) påvirker signalstyrken.

Alle delene blir grundigere forklart og beskrevet i kapittel 4 Detaljert løsning.

2.2 Avgrensninger og begrensninger

I dette bachelorprosjektet har det vært noen avgrensninger og begrensninger som kan ha påvirket prosjektforløpet. Begrensningene og avgrensningene har hjulpet prosjektet med å gjøre det lettere å forholde seg til problemstillingen.

En begrensning som prosjektet har er at det bare er tilgang på Eidsiva sine sensorer som vil si at det bare er et utvalg sensortyper som testes. Disse sensorene er av høy kvalitet som vil si at det blir forsket mer på selve protokollen enn på sensor-hardware. Hadde prosjektet brukt billigere sensorer i tillegg så hadde man tydeligere sett om det var hardware på sensorene som utgjorde forskjellen eller om det er protokollen.

Gruppen har liten erfaring med LoRaWAN teknologien når prosjektet starter som gjør at gruppen bruker litt tid på å sette seg inn i teknologien og andre teknologier som gjør det tilsvarende. Tilgjengelig litteratur har vært begrenset til dette spesifikke området med tidskritiske applikasjoner, men det er mye annen god litteratur som snakker om egenskapene til LoRaWAN.

Avgrensninger som er gjort i prosjektet er at det bare er gått i dybden på om det kan brukes innenfor tidskritiske applikasjoner. Det nevnes andre bruksområder for LoRaWAN, men går ikke i dybden for om det faktisk er gunstig å bruke LoRaWAN overfor andre lignende protokoller.

Det som gruppen definerer som en tidskritisk applikasjon er: En applikasjon som utfører tidskritiske operasjoner. En tidskritisk operasjon er en operasjon som manipulerer data som har tidsbegrensninger assosiert med dens behandling. En behandling av tidskritiske data er følgende operasjoner på data: mottakelse, prosessering, transportering, lagring og presentering. Et eksempel på en tidskritisk applikasjon som bruker data fra en LoRaWAN sensor kan være en LoRaWAN sensor som er plassert på en fryserom i en dagligvarebutikk. Denne sensoren skal gi beskjed dersom temperaturen i rommet stiger over -4 grader. Dersom applikasjonen mottar denne beskjeden, skal det gå en alarm. Her er det viktig at alarmen går med en gang temperaturen faller utenfor og dermed er det tidskritisk. Et annet eksempel kan

være i en helseapplikasjon der en bruker har en panikknapp som den kan trykke på dersom den trenger akutthjelp, når applikasjonen registrerer at brukeren har trykket på knappen skal den sende en ambulanse til lokasjonen til brukeren. Det vi ønsker å finne ut er om LoRaWAN er et relevant alternativ til bruk i tidskritiske applikasjoner og hvilke tidskrav LoRaWAN sensorer oppfyller. Et tidskrav er definert som hvor ofte data går tapt og hvor forsinket data kan være. For eksempel i de to tidligere eksemplene vil tidskravet for helseapplikasjonen være at data aldri kan gå tapt og dataen kan ikke være forsinket i det hele tatt. For applikasjonen i dagligvarebutikken har applikasjonen litt bedre tid så tidskravet for de to applikasjonene vil ikke være det samme. Prosjektets fokus er lagt til å forske på om LoRaWAN er pålitelig nok til å kunne brukes i tidskritiske applikasjoner som for eksempel innenfor helse.

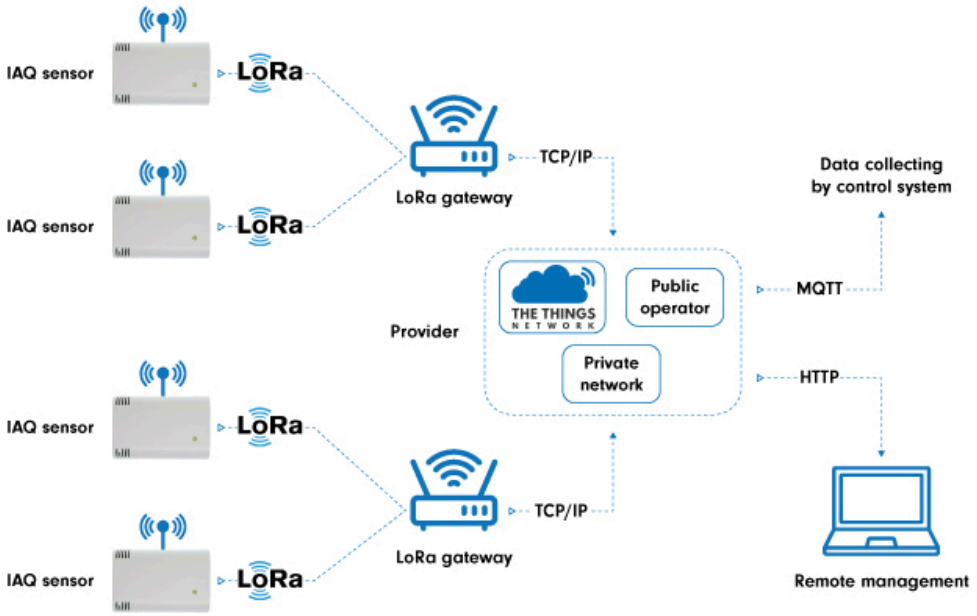
2.3 Ressurser

For å gjøre dette prosjektet var gruppen avhengig av å ha tilgang til tekniske og faglige ressurser. Denne seksjonen vil ta for seg de tekniske ressursene som var nødvendige for å kunne utføre prosjektet.

2.3.1 Tekniske ressurser

For å kunne forske på LoRaWAN og se på pakketap, signalkvalitet, og lignende, er det nødvendig å ha tilgang på sensorer og sensordata.

I arkitekturen til LoRaWAN er det gatewayer som tilbyr dekning til sensorer. Er en sensor innenfor denne dekningen, kan gatewayen motta datapakken, mer om gateways i kapittel 2.4. Når gatewayen har mottatt datapakken vil den sende den videre til en provider som så sender denne dataen ut på MQTT. MQTT er en standard melding-protokoll for IoT enheter. Den er designet for å transportere lette publisere/abonnere meldinger som er ideell for å koble til eksterne enheter med et lite kodeavtrykk og minimal nettverksbåndbredde (MQTT, n.d). Figur 2.1 viser hvordan arkitekturen mellom sensorer, gateways, provider og MQTT er bygd opp.



Figur 2.1: Arkitektur og sammenheng mellom LoRaWAN-enheter (Unipi technology, n.d)

For å få tilgang til sensor-dataene kan man koble seg opp til en datastrøm med MQTT. Med MQTT kan man bare hente ut dataene manuelt. Det er derfor nødvendig å lagre dataen et sted og etter det kunne hente den ut igjen. Gruppen har valgt å lage et program som gjør dette. For å hente ut dataene igjen er det lagt et API.

> *"Et API er et grensesnitt som gir direkte tilgang til data og funksjonalitet i et datasystem,..."* (SNL, 2024)

GO er brukt for å utvikle API som henter og lagrer data fra MQTT datastrømmen. Python er brukt for å utvikle script som analyserer og visualiserer pakketap og signalstyrke. Grunnen til at gruppen har gjort det sånn er forklart i kapittel 3.3. Det hadde vært mulig å implementere løsningen ved å bruke bare ett av programmeringsspråkene. Men siden de to språkene har forskjellige egenskaper som gjør dem best egnet til hver sin oppgave, og gruppen ønsker å prøve ny teknologi blir det brukt to programmeringsspråk for å implementere løsningen. For å utvikle API´et ble det valgt et utviklingsmiljø og programmeringsspråk som er optimalt for å sette opp API-er. Go (Golang) er et språk som er kjent for å være enkelt å sette opp enkle API-er og webservere, på grunn av sin lette syntax og importerte pakker. I Go er det ikke mange linjer med kode som trengs for å sette opp en webserver som hoster API-et, sammenlignet med for eksempel Java som har en forholdsvis lang syntaks hvor det trengs

mange linjer med kode for å få opp en webserver. Pakker som ble brukt i API-et kan finnes i "go.mod"-filen.
For å få API-et til å hente ned datapakker hele tiden uten at noen trenger å kjøre det manuelt på sin egen datamaskin, så er det valgt en skyplattform for å kjøre API-et.
Den plattformen som ble valgt til dette er Azure, da oppdragsgiver bruker dette og har mange fagpersoner som kan hjelpe ved eventuelle problemer. Mer om Azure og brukte ressurser i Azure i kapittel 4.1.1 - 4.1.4

Til å lage enkle scripts for å evaluere og visualisere pakketap og signalkvalitet ble programmeringsspråket Python valgt. Python er et avansert programmeringsspråk som ofte blir brukt for å bruke ulike komponenter sammen. (Rossum, 1998)
I python er det enkelt å kalle på API-er, da det bare trengs et par linjer med kode ved hjelp av et innebygd bibliotek. Sammenlignet med for eksempel Java hvor det kreves at du setter opp headers før du gjør kallet, noe som ikke trengs i Python. Som en del av dette er det nødvendig med noen ulike pakker for å utføre oppgavene. Liste over brukte pakker kan finnes i "requirement.txt" på Github.

Git ble valgt som versjonskontrollverktøy for å sikre en effektiv måte å dele og lagre koden på. Dette er gjort med repositories i github. Dette hjalp med å holde god oversikt over endringer på koden, og at flere kunne arbeide likt og ha det på et felles sted. I tillegg ble det lett å holde oppdragsgiver oppdatert på koden. I github er det også lagt inn CI og CD for å automatisk kjøre tester når ny kode blir merget inn i en branch.

2.3.2 Kommunikasjon og veiledning

Kommunikasjonen mellom prosjektmedlemmene skjedde stort sett i en felles kanal på sosiale medier og felles oppmøter på skolen, og møtene med veileder skjedde fysisk på skolen på HVL. Møtene med oppdragsgiveren ble holdt digitalt på teams, og annen kommunikasjon ble benyttet over mail. Det ble brukt god kommunikasjon som sikret jevn fremgang i prosjektet.

Oppdragsgiver stilte med ressurspersoner som hjalp prosjektet med Azure og andre tekniske aspekter. HVL-veileder kom med mange gode spørsmål og tips om det tekniske som gjorde at prosjektet hadde god fremgang.

2.4 Hva er LoRaWAN

2.4.1 LoRaWAN

LoRa står for Long Range og er en radioteknologi for trådløs overføring av data over lange avstander. LoRa benytter seg av lav bitrate som betyr at dataoverføring krever lavt energiforbruk. Fordelen med dette er at sensorer har lang batterilevetid og kan stå lenge uten at en må tenke på å bytte batteri. Kombinasjonen av langdistansekommunikasjon og lavt energiforbruk gjør LoRa attraktivt for en rekke ulike industrielle IoT-applikasjoner (Rush, 2021).

LoRa er basert på Chirp Spread Spectrum (CSS) teknologi. Chirps, som også kalles symboler, er måten data blir sendt på. Denne teknikken koder informasjon på radiobølger ved hjelp av chirp-pulser. Dette gjør at dataoverføringen er svært robust mot forstyrrelser og støy, og har lang rekkevidde. Denne måten å overføre data på er ideell for applikasjoner som skal overføre små mengder data med lav bitrate over lange avstander (Rush, 2021).

> *"Data kan overføres over et større området enn andre trådløse teknologier som Wi-Fi, Bluetooth og ZigBee" (Rush, 2021)*.

LoRaWAN står for Long Range Wide Area Network og er navnet på protokollen som bygger på LoRa radioteknologi. LoRaWAN-nett er satt sammen av sluttnoder og gateways som samler og sender data. Denne dataen blir sendt til en nettverksserver hvor data blir behandlet og lagret. Denne kommunikasjonen er toveis, som vil si at det er mulig å hente ut data fra sensorer, men en kan også sende data fra en endenode som for eksempel en trykknapp som kan fungere som en panikk -eller fallalarm.
Serveren er ansvarlig for å behandle data. Oppgaver serveren har kan være å slette duplikater av datapakker, bekrefte pakkens integritet og utføre andre sikkerhetsmomenter (TECTELIC, 2023).

Nettverksserveren laster opp dataen til en applikasjonsserver. Arbeidsoppgavene til applikasjonsserveren er å tolke data mottatt fra endenodene. Avhengig av hvordan applikasjonen utvikles så kan dataene tolkes både visuelt og/eller analytisk (Rush, 2021).

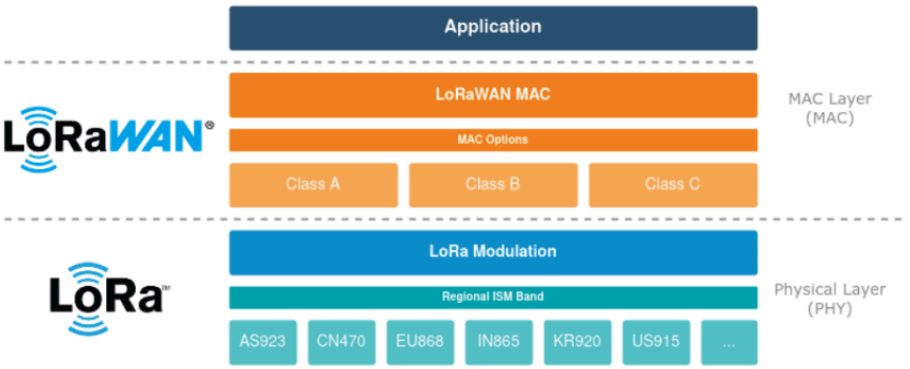
LoRaWAN protokollen har tre modus klasser, disse er A, B og C. Klasse A kommunikasjonmodusen gjør at enheter i denne klassen kun er aktiv i korte perioder. Enheten sender en uplink melding med jevne mellomrom, deretter åpnes det for to korte mottaks vindu slik at enheten kan ta imot data. Nettverket kan i disse periodene sende signaler. Denne modusen er den mest økonomiske, da enheter bare slår seg på ved gitte mellomrom, resten av tiden står enheten i hvilemodus. Dette gjør at enheten sparer energi og batterilevetiden er høyere enn de følgende kommunikasjonsmodusene (TECTELIC, 2023).

Klasse B bygger på klasse A med å periodisk åpne mottaks vinduene for mottak av data. Disse vinduene kalles for *ping slots*. Nettverket sender jevnlig signaler, gjennom gatewayene, som til slutt mottas av endenodene. Signalene gir en tidsreferanse til endenodene slik at de kan justere sine interne klokker i forhold til nettverket. Dette gjør det mulig for nettversserveren å vite når og til hvilken endenode den skal sende signal til. Når en klasse B enhet har riktig tidssynkronisering, går den inn mottaksmodus i sine ping slots for å sikre at den mottar signal som sendes fra nettverket (TECTELIC, 2023).

Denne modusen brukes om det trengs å sende og motta signaler oftere enn i klasse A. Hastigheten på prosessen er også høyere enn i klasse A fordi en slipper å vente på en uplink før en kan sende signal til endenoden som nå har et åpent mottaks vindu.

Til slutt har vi klasse C. Denne klassens kommunikasjonsmodus er den mest pålitelige modusen. Grunnen til dette er at mottakeren av et signal alltid er aktivert. Dette betyr at signal kan mottas når det sendes uten å måtte vente på et åpent vindu hvor mottaker er klar til å motta signal. Basert på dette betyr det at energiforbruket blir mye høyere, som igjen betyr at batterilevetiden er lavere (Welch, 2022).

Figur 2.2 viser nettverkslagene i et LoRaWAN-system fra leverandør til applikasjon.



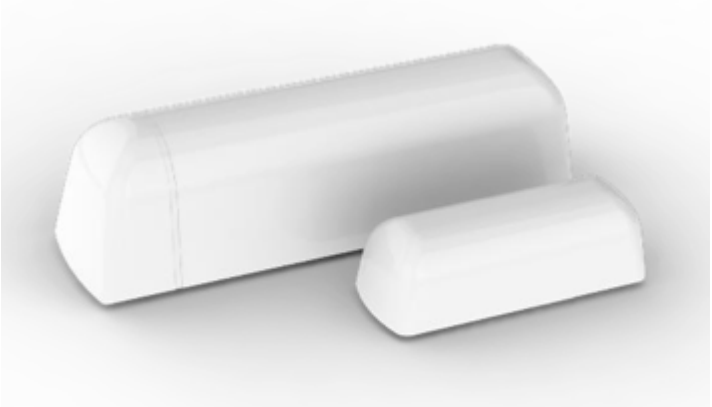
Figur 2.2: Nettverkslag i LoRa teknologien (Pous, 2021)

2.4.2 LoRaWAN endenode

Endenode også kalt for sluttnode beskriver enheter som er plassert i slutten av et nettverk. I typiske tilfeller er endenoden en sensor. Denne sensoren har som jobb å samle inn og/eller overvåke data med jevne mellomrom. Tidsintervallet sensoren måler på varierer, men eksempel på intervaller er 15 minutter mellom hver måling og andre sensorer kan vente en time mellom hver måling. Intervallet varierer basert på hvor ofte det er nødvendig med måling. Kritiske applikasjoner trenger gjerne oftere måling enn mindre kritiske applikasjoner. Et eksempel på en måling hvor en ikke trenger like hyppige oppdateringer som andre kan være mål på hvor mye det regner på en gitt plass.
Når sensoren "venter" på å måle, går den i dvale for å spare strøm. Dette betyr at hyppig måling betyr lavere batteritid (Rush, 2021).

Sensorer som benytter seg av LoRaWAN teknologi kommer i mange varianter og det finnes en rekke forskjellige leverandører for disse. Oppdragsgiveren i prosjektet bruker i hovedsak sensorer levert av Elsys. Elsys leverer sensorer for smarte bygg, "smart-cities" og smart-industri. Sensorene kommer i flere typer som for eksempel ERS2 CO2 modellen som måler Co2 -nivå, temperatur, luftfuktighet, lys og bevegelse (Figur 2.3). Et annet eksempel på en sensor levert av Elsys er EMS Door som måler antall ganger en dør åpnes (Figur 2.4).

Figur 2.3: Bilde av Elsys ERS2 CO2 sensor (Elsys, 2019)



Figur 2.4: Bilde av Elsys EMS Door sensor (Elsys, 2020)

I prosjektet har gruppen fått tilgang til en kontinuerlig datastrøm distribuert av oppdragsgiver. Datastrømmen inneholder sensordata fra sensorene Eidsiva benytter seg av, blant annet sensorene avbildet over i figur 2.3 og 2.4.

2.4.3 LoRaWAN Gateway

LoRaWAN gateways opptrer som et bindeledd mellom sluttnoder og nettverksserveren på nettet. Hovedoppgaven til en gateway er å motta sensordata fra sensorer ved hjelp av LoRa-konsentrator og videresende denne dataen for videre behandling til server via internett eller annen privat nettverksinfrastruktur (Rush, 2021).

En LoRaWAN-gateway kan sende og motta meldinger over lange avstander. Avstandene i urbane omgivelser som byer og lignende er opp til 5 km og i landområder opp til 15 km (Dusuniot, 2022).

Det finnes også signal repeater som forsterker ytelsen til signal i områder med dårlig dekning som for eksempel under bakken eller i tette bygg.

2.4.4 Bruksområder

LoRaWAN nettverk har siden starten i 2015 fått et bredt spekter av bruksområder innenfor IoT-applikasjoner. Bruksområdene varierer, men en gjenganger i områdene LoRaWAN brukes er smarte løsninger.

"Smart cities" eller smartbyer er et mye brukt begrep som betyr smarte løsninger og teknologi brukt i byer i dag. Smartbyer krever sanntidsvisning av flere aspekter ved bylivet. Eksempel på noen aspekter som trenger sanntidsvisning er gatelys, parkeringsmålinger og tømming av søppel. For å oppnå dette blir ofte bredt dekkende trådløse nettverk benyttet for å gjøre tilkobling mellom et stort nettverk av forskjellige IoT-enheter, sensorer og smarte målere som generer og overvåker data. LoRaWAN er en bredt dekkende teknologi som ikke krever mye energi og sensorene har ofte lang levetid. LoRaWAN er derfor en nyttig teknologi å benytte ved utvikling av smarte byer. Ved å utnytte LoRaWAN kan en enkelt visualisere og analysere omfattende sanntidsdata for å dynamisk kunne optimalisere utnyttelse og fordeling av byens ressurser. Oppgaver kan automatiseres og og byer kan overvåke når vedlikehold trengs samtidig som driftskostnader kan reduseres (TECTELIC, 2023).

På samme måte som LoRaWAN kan effektivt kan brukes i smartbyer, kan smartbygg også dra nytte av teknologien. Smarte bygninger som bruker sensorer til å overvåke og generere data vil kunne bidra til å redusere driftskostnader og fordele byggets ressurser på en mer effektiv måte.

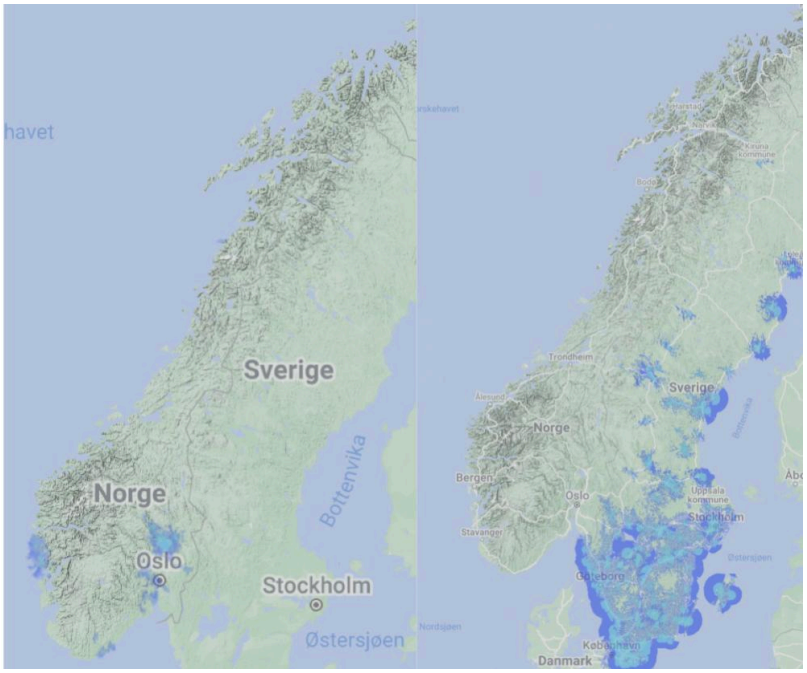
LoRaWAN brukes også innenfor logistikk og transportnæringen. Forsyningskjede- og logistikkselskaper kan dra nytte av den lange rekkevidden til LoRaWAN. Teknologien kan hjelpe selskapene med å spore pakker og andre verdifulle eiendeler. På grunn av visualiseringen av sanntidsdata så kan også pakker som er underveis spores, selv når det er snakk om sendinger som er sendt over store geografiske områder (TECTELIC, 2023).

På samme måte som i smartbyer, kan LoRaWAN benyttes for å redusere driftskostnader, optimalisere bruken av verktøy og personale og gi innsyn til vedlikeholdsbehovet.

Lavt strømforbruk, lang rekkevidde, lavt infrastrukturbehov og mulighet for sanntid visualisering og analysering av data gjør at LoRaWAN har blitt en populær protokoll med en rekke ulike bruksområder.

2.4.5 LoRaWAN i Norge

Bruken av LoRaWAN-protokollen i Norge er i hovedsak tatt i bruk i Innlandet og Bergen. Dette er betraktelig mindre enn andre land. I Figur 2.5 kan en se forskjell på LoRaWAN dekning i Norge og Sverige.



Figur 2.5: Bilde av LoRa dekning i Norge sammenlignet med Sverige. (Netmore, 2024)

3 Design av prosjektet

3.1 Forslag til løsning

I denne delen av rapporten vil alternative løsninger til hvordan vi kan oppnå målet for prosjektet bli diskutert. Tidligere i rapporten har det blitt satt opp 3 delmål. En alternativ løsning vil beskrive hvordan gruppen tenker det er mulig å gå fram for å nå disse 3 delmålene. Til slutt vil de forskjellige løsningene bli vurdert og det vil bli begrunnet hvorfor den valgte løsningen er valgt.

3.1.1 Alternativ løsning 1

I alternativ løsning 1 vil det for det meste bli utviklet scripts for å nå delmålene. For å måle ytelsen til LoRaWAN sensorene ønsker gruppen å utvikle scripts som kjøres over lengre tid. Det skal utvikles et script for å måle ytelsen på pakketap og signalkvalitet til LoRaWAN sensorene. Det skal også utvikles et script som måler spredning faktoren på signalene til LoRaWAN sensorene. For å oppnå målet om å finne ut om dataen fra LoRaWAN sensorene er anvendbare, ønsker gruppen å implementere en enkel applikasjon der en bruker kan se live data fra forskjellige LoRaWAN sensorer.

3.1.2 Alternativ løsning 2

I alternativ løsning 2 vil det bli brukt tidligere forskning og allerede bygde applikasjoner for å oppnå delmålene. LoRaWAN er ingen ny teknologi og det finnes allerede masse data på ytelsen til LoRaWAN og det finnes mange applikasjoner som bruker LoRaWAN data. Derfor kan en mulig løsning på prosjektet være å bruke litteratursøk for å finne tidligere rapporter og forsøk som svarer på det gruppen ønsker å finne ut. For å finne ut om LoRaWAN data er anvendbar kan gruppen se på allerede utviklede applikasjoner som bruker LoRaWAN data.

3.1.3 Alternativ løsning 3

I alternativ løsning 3 vil det bli brukt en sammensetning av løsning 1 og løsning 2. Gruppen vil både utvikle egne script og samle inn resultater samt bruke litteratursøk for å finne relevant litteratur som omhandler bruk av LoRaWAN. Gruppen ønsker å finne eksisterende eksempler på bruk av LoRaWAN samt finne data på ytelsen til LoRaWAN.

3.1.4 Diskusjon av løsningene

Hver av løsningene har fordeler og ulemper. Fordelene med løsning 1 er at LoRaWAN sensorene som blir testet er akkurat den typen sensor oppdragsgiver bruker, samt den blir testet i et område der oppdragsgiver er nysgjerrig på ytelsen til sensorene. Fordelene med løsning 2 er at gruppen vil finne data fra mange forskjellige kilder som vil gi et mer komplett bilde av ytelsen til LoRaWAN sensorene. Gruppen vil få se flere applikasjoner som implementerer LoRaWAN data. Fordelen med løsning 3 er at gruppen vil få fordelene med både løsning 1 og løsning 2. Ulempen med løsning 3 er at den krever mye arbeid og kan føre til at gruppens egne utviklede løsninger og gruppens research blir dårligere enn den hadde vært dersom gruppen hadde fokusert mer i en retning.

3.2 Valgt løsning

Løsningen gruppen har valgt å gå for er alternativ løsning 3. Dette er fordi denne løsningen vil gi et mer komplett bilde og vil oppfylle delmålene gruppen har satt opp på den best mulige måten. Ved valg av løsning 3 vil gruppen utvikle egne scripts og utføre litteratursøk. Gjennom script vil gruppen få data fra samme type LoRaWAN sensor som oppdragsgiver bruker.

3.3 Valg av verktøy

GOLang

Gruppen har valgt å bruke GoLang programmeringsspråket også omtalt som GO. GO brukes til å implementere en service som skal kjøre i Azure og lagre forskjellige sensordata når den blir mottatt. Gruppen skal også bruke GO til å implementere et API som kan hente ut lagret sensordata. Go er et åpent kildekode-programmeringsspråk utviklet av
Google (Barney, 2023). GO er et moderne programmeringsspråk, det ble lansert i 2012 og blir nå brukt av store selskaper som Google Cloud Fare og Netflix. På grunn av at språket er raskt, statisk skrevet og det kan kjøres uten en virtuell maskin er GO et språk bra egnet for å implementere "Microservices" (Barney, 2023). På grunn av dette og fordi Eidsiva også har god erfaring med dette språket og kan tilby hjelp, valgte gruppen å bruke Go for å implementere datainnsamling og API implementering.

Python

For å utvikle scripts som analyserer, evaluerer og visualiserer oppsamlet sensordata, har vi valgt å bruke Python programmeringsspråket. Python er et høyt-nivå, objektorientert programmeringsspråk, det er ikke type fast og er et av de mest populære språkene i verden (Coursera, 2024).

Python har mange biblioteker som NumPy, pandas, Matplotlib, sick it-learn, Tensorflow og Keras som alle er designet for å bli brukt innenfor data analyse og visualisering
(Luna, 2023).

På grunn av dette og fordi Python er et av de mest brukte programmeringsspråkene for dataanalyse, har gruppen valgt å bruke dette programmeringsspråket.

GoLand

Valget av utviklingsmiljø for Go implementasjonen ble GoLand. På grunn av dets brukervennlighet, støtte for forskjellige biblioteker og kraftige verktøy som debugging er dette et veldig bra valg når du skal utvikle Go-kode.

Visual Studio Code

For å utvikle python-scriptene, valgte gruppen å bruke Visual Studio Code. På grunn av Visual Studio Code sin brukervennlighet, fleksibilitet og omfattende utvalg av utvidelser som bidrar til å gjøre arbeidsflyten bedre. Visual Studio Code tilbyr også gode verktøy til bruk for visualisering og matematiske funksjoner, gjennom tidligere nevnte biblioteker som for eksempel pandas.

Azure

I dette prosjektet skal gruppen sette opp et API og et program som mottar og lagrer sensordata. For å gjøre dette trenger vi å sette dem opp i et kjøretidsmiljø. Kjøretidsmiljøet vi har valgt å bruke er Azure. Grunnen til at vi valgte å bruke Azure er fordi Eidsiva har erfaringer med det og kan hjelpe ved problemer. Siden medlemmene av gruppen er studenter får alle 1000kr gratis til å bruke i Azure. Med dette har Eidsiva sagt at det er lettere hvis gruppen selv distribuerer og drifter Azure Resource Groupen med tanke på interne tilganger. Eidsiva har derimot sagt at hvis det går over gratis pengene så er det bare å gjøre et utlegg som de betaler for.

Azure DevOps

For utvikling av scripts og applikasjoner bruker vi Scrum. For å sette opp scrum boards og sprints har vi valgt å bruke Azure Devops. Azure Devops er et sett verktøy lagd for å hjelpe utviklingsteam å planlegge og dokumentere utvikling (Codefresh). Grunnen til at vi valgte å bruke Azure Devops er igjen at vi får det gratis siden vi er studenter, og fordi et gruppemedlem har erfaring med det fra før.

3.4 Prosjektmetodikk

3.4.1 Litteratursøk

3.4.1.1 Litteratursøk metoder

I prosjekter som gjerne ikke har et konkret svar, og baserer seg på forskning og undersøkelse, så er litteratursøk viktig. Når en skal foreta litteratursøk for innhenting av informasjon til en oppgave er det en del komponenter å tenke igjennom. Først må problemstillingen defineres og en må finne ut hvilken informasjon en trenger og hva en skal lete etter. Deretter er det viktig å finne konkrete søkeord for informasjonen en ønsker.

Litteratursøk brukes til forskjellige type oppgaver og det finnes flere måter å utføre litteratursøk på, avhengig av hva en ønsker å finne. Eksempler på litteratursøk er presise - og fullstendige litteratursøk.

Fullstendig litteratursøk, også kalt for systematisk litteratursøk, har som navnet tilsier fokus på fullstendighet. Her vil en prøve å innhente alle relevante artikler for det gjeldende temaet. Dette gjøres ved å gjøre omfattende søk i flere forskjellige databaser. Denne metoden å utføre litteratursøk er tidkrevende, forutsetter god planlegging og strukturert søking. Fullstendig eller systematisk litteratursøk bør gjerne gjøres sammen med en bibliotekar for å spare tid og unødvendig leting (Tvedt, 2021).

En annen måte å utføre litteratursøk er det som kalles presist litteratursøk. Denne metodikken går ut på å finne færre med mer relevante artikler for temaet en trenger litteratur til. Denne

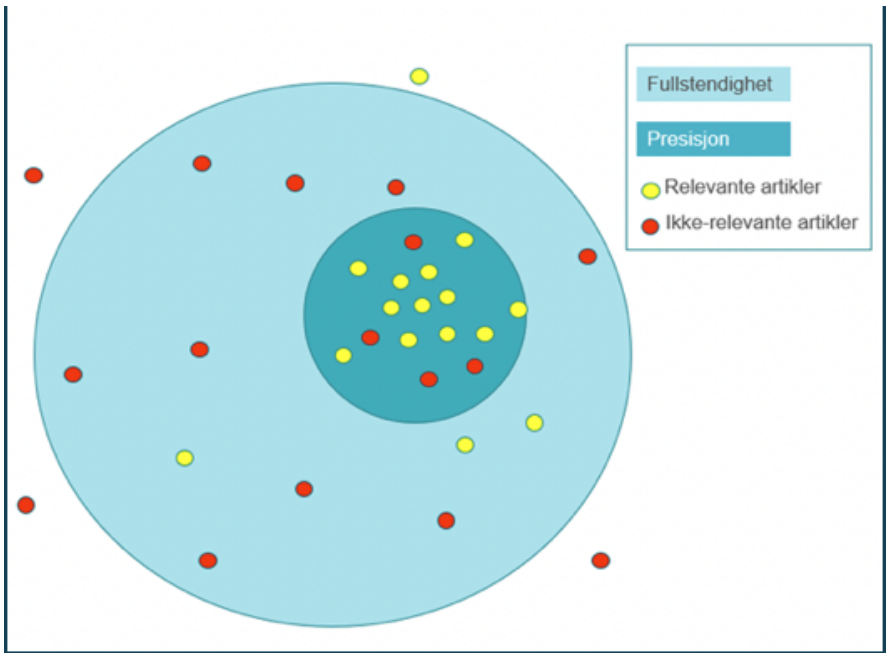
type søk gjøres gjerne i fag - og tema spesifikke databaser for å forhindre upresise og vage artikler som ikke vil være relevante for oppgaven. Nøkkelen i denne metodikken er å utnytte gode og presise søkeord som på en god måte beskriver det en leter etter. En kan og benytte seg av kombinasjoner av nøkkelord for å være enda mer presis. Samtidig som det er ønskelig å være presis så kan en være for presis og søket kan ende med å gi null treff. Da er det viktig å enten korte ned på antall nøkkelord eller rekonstruere hvordan nøkkelordene er sammensatt. Hvordan søkeord er satt sammen har også noe å si for resultatet. Søkeord kan kombineres med kombinasjonsordene AND, OR eller NOT (Tvedt, 2021).

3.4.1.2 Diskusjon av litteratursøk

Fullstendig litteratursøk vil gi bredere forståelse av emnet på kostnad av at det kreves mye tid og ressurser til å gå igjennom alle artikler og tidsskrifter som kommer opp ved denne typen søk. Det fullstendige søket vil også gi treffe på ikke-relevante artikler som en må bruke tid på å lese igjennom. Søket gir mange treff, men ikke alle treffene er nødvendigvis gode. Metodikken kan være nyttig når en trenger en oversikt over det som er publisert i forskningsfeltet og en generell oversikt over temaet.

Presist litteratursøk vil spare mye tid ved å minimere antall artikler en må lese igjennom og gi treff på relevant litteratur. Søket gir presise treff, nyttig informasjon kan bli forbigått i andre artikler då disse ikke inngår i det presise søket.

I illustrasjonen fremvist i figur 3.1 kan en se forskjell på antall og kvalitet på treff ved å benytte seg av de forskjellige metodene å gjøre litteratursøk på.



Figur 3.1: Illustrasjon som sammenligner antall og kvalitet på treff en får ved presist - og fullstendig litteratursøk (Tvedt, 2021).

3.4.1.3 Valg av litteratursøk

Prosjektet går ut på å forske på om LoRaWAN er en protokoll for sending av datapakker som kan benyttes i tidskritiske applikasjoner. Her trenger gruppen generell informasjon om LoRaWAN og benytter seg av fullstendig litteratursøk. Denne metodikken benyttes fordi gruppen trenger oversikt over hvilken forskning som er gjort og for å se et helhetlig bilde om hva LoRaWAN er og hvordan protokollen kan benyttes. Når det gjelder spørsmålet om LoRaWAN kan benyttes i tidskritiske applikasjoner trengs mer presise søk. Her benytter gruppen seg av metodikken presise litteratursøk. En kombinasjon av de vanlige metodikkene av litteratursøket blir benyttet i prosjektet for å spare tid ved å finne presise artikler som er relevante, men og få en helhetlig oversikt over emnet og hva som er gjort av tidligere arbeid.

3.4.1.4 Litteraturssøk metodikk

I prosjektet benyttet gruppen seg av både fullstendig - og presise litteratursøk. For å få oversikt over LoRaWAN som protokoll samtidig som at gruppen ønsket å se hvor mye litteratur som finnes, ble metodikken fullstendig litteratursøk brukt. I denne prosessen startet gruppen med søkeordet "LoRaWAN" for å få en oversikt over protokollen. Senere ønsket gruppen å se hvilke bruksområder LoRaWAN protokollen oftest er benyttet. Kombinasjonen av søkeordet "LoRaWAN" og "bruksområder" ble i denne forbindelse benyttet, med AND

som kombinasjonsord for å få resultater som inneholdt informasjon om LoRaWAN sine bruksområder. Søket ga mange resultater, og dette gav gruppen en god oversikt over hvor protokollen blir benyttet i dag.

Prosjektet skal undersøke om LoRaWAN kan er pålitelig *nok* og kan benyttes i tidskritiske applikasjoner. Gruppen ønsket dermed å søke etter resultater som beskrev LoRaWAN protokollen i tidskritiske applikasjoner. Her ble kombinasjonen av søkeord "LoRaWAN" og "tidskritisk" med kombinasjonsordet AND benyttet. Søket ga få resultater, dermed ble søkeordet "timecritical" brukt for å øke sjansen for flere treff med utenlandsk litteratur og ikke bare norsk litteratur. Dette ga noe flere resultater.

Da helsesektoren ofte kan sees på som en bransje hvor tid kan være kritisk, ble enkelte søkeord fra denne bransjen benyttet i kombinasjon med "LoRaWAN". Søkeord som "helse", "hospital", "healthcare" eller lignende, ble brukt for å få en oversikt over om LoRaWAN brukes i helsesektoren i dag. Dette ga noe flere resultater som gruppen dro nytte av. Mer om hvordan litteratursøket ble gjort i kapittel 4.2.

3.4.2 Utviklingsmetodikk

3.4.2.1 Utviklingsmetoder

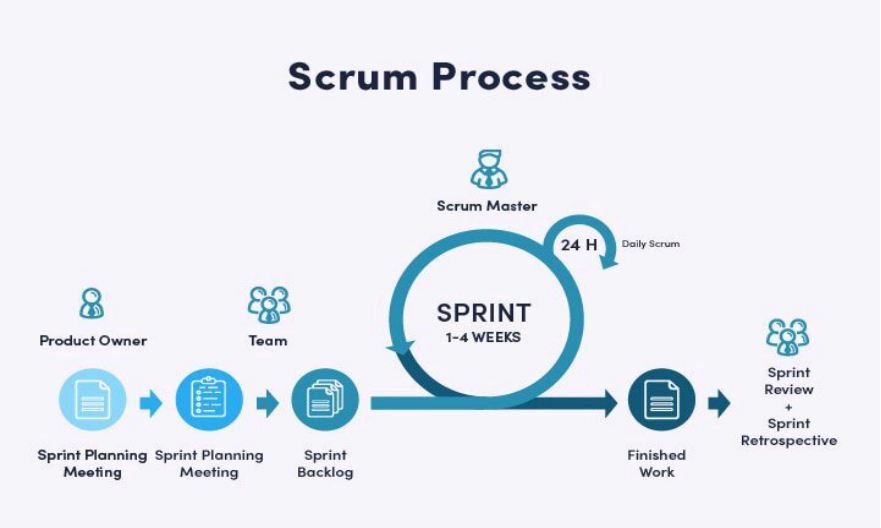
Under oppstartsmøtet med oppdragsgiver ble det gjort klart at gruppen står fritt til å velge utviklingsmetode selv. Prosjektet er i hovedsak en forskningsbasert bacheloroppgave, og utviklingsmetodikken vil ikke være veldig relevant. I den delen av prosjektet hvor gruppen skal lage et API og scripts for å teste signalstyrke og undersøke pakketap bestemte gruppen at det skulle benyttes en smidig (agile) utviklingsprosess på denne delen. Vanlige og mye brukte smidige metoder er Scrum, Kanban og XP (extreme programming). Ofte kan en kombinasjon av flere metoder benyttes.

Scrum er den utviklingsmetoden som er mest populær når det snakkes om smidige utviklingsmetoder. Hovedideen til utviklingsmetoden er å dele opp prosjektet i sykluser eller trinn i utviklingen, kalt for en sprint. En sprint er hvor det jobbes mot et fastsatt mål innenfor en gitt tidsramme (Santo, 2022).

Tidsrammen for en sprint er opp til gruppen som anvender utviklingsmetoden, men vanlig lengde er mellom 1 til 4 uker (Praecipio, 2022)

Poenget er ved å stykke opp arbeidet i disse delmålene som jobbes mot i sprintene, så vil det være fremgang i prosjektet og omfanget av prosjektet føles mer overkommelig.

Ved å benytte seg av Scrum så startes hver dag med et lite stand-up møte hvor gruppen eller teamet som jobber på prosjektet møtes i ca. 15 min for å snakke om fremgang og utfordringer. Dette kalles en "daily scrum". Figur 3.2 illustrerer en scrum prosess.



Figur 3.2: Scrum prosess (Stanke, n.d).

Kanban går ut på å ha arbeidsoppgaver og fremgang på disse visualisert med "kort" på en tavle som kalles "kanban board". Meningen er at hele gruppen eller teamet skal kunne se og legge til arbeidsoppgaver for prosjektet. Tavlen deles som regel inn i 3 kolonner. Disse kolonnene er "to do" eller "gjøremål", "doing" eller "i arbeid" og til slutt "done" eller "ferdig".

Et eksempel på bruk er at et gruppemedlem velger seg et kort fra "to do" kolonnen og flytter den til "doing" kolonnen og skriver gjerne navnet sitt på kortet. Grunnen til dette er at resten av gruppen vet hvem som jobber med hva. Når vedkommende er ferdig med oppgaven så

flyttes kortet til “done” kolonnen. Deretter finner gruppemedlemmet en ny arbeidsoppgave å jobbe med (Santo, 2022). Figur 3.3 illustrerer et kanban board.

Figur 3.3: Eksempel på et kanban board (Santo, 2022).

XP eller “extreme programming” er en annen smidig utviklingsmetode. Denne metoden sitt hovedprinsipp er å hele tiden ha i fokus å få laget noe som virker på enklest mulig måte. Prosjektet deles også her opp i flere iterasjoner, men hver iterasjon løses på enklest mulig måte uten å ha for mye fokus på helheten eller sluttproduktet.

I XP metodikken testes produktet fra dag en og tilbakemeldinger fra kunder og andre relevante personer hentes hyppig og tas til betraktning med en gang. Utviklingsmetoden gir et prosjekt høy effektivitet og produktivitet.

Grupper som benytter seg av XP som utviklingsmetode benytter seg også av andre aktiviteter som par-programmering (Santo, 2022). Figur 3.4 illustrerer en XP loop.

Figur 3.4: Planlegging og tilbakemelding loop (Santo, 2022)

3.4.2.2 Diskusjon av utviklingsmetode

Oppdragsgiver har gitt gruppen fritt spillerom når det kommer til valg av metoder og verktøy benyttet i prosjektet.

Ved sammenligning av de mest populære utviklingsmetodene og prosjektmetodikk har gruppen diskutert for og i mot for hvert alternativ. Diskusjonen er nødvendig for at prosjektet skal ha best mulig flyt og systematisk oppbygging. Ved sammenligning kan en se at XP egner seg til utvikling av produkter hvor deler av produktet skal tas i bruk fort. Derav metodologien: Den enkleste måten å få noe til å fungere. Denne metoden følger og kontinuerlig leveranse (CD, continuous delivery) konseptet. Prosjektet presentert i denne rapporten går ikke ut på å ha kontinuerlig leveranse. Prosjektet tar for seg forskning om LoRaWAN protokoll er egnet for tidskritiske applikasjoner.

Utviklingsmetoden kanban kan være en nyttig utviklingsmetode for prosjektet, da alle gruppemedlemmene kan se hvilken arbeidsoppgaver de andre på gruppen jobber med.

Kanban er derimot mer egnet for større og kontinuerlige prosjekter som ikke nødvendigvis har en fastsatt tidsramme.

Scrum er en utviklingsmetode som egner seg godt til prosjektarbeid med fastsatt tidsramme. Dette prosjektet har en fastsatt tidsramme og er et prosjekt som startes på, det er ikke allerede en påbegynt arbeidsflyt. Scrum er godt egnet til mindre grupper og nystartede prosjekter. Scrum er også en fleksibel metodikk. I et prosjekt som baserer seg på forskning og testing/utforsking er fleksibilitet viktig. Uventede vendinger kan forekomme og ting av ulike grunner kan ta lengre tid enn antatt, derfor er det viktig at problemer kan løses fortløpende. Dette er grunner til at gruppen ble enig i å benytte seg av denne metodikken for utviklingsdelen i prosjektet.

3.4.2.3 Valg av utviklingsmetode

Utvikling

Selv om oppdragsgiveren gav gruppen fritt spillerom til valg av utviklingsmetoder og andre verktøy, anbefalte de oss å bruke Azure sine verktøy. Grunnen til dette er at det er veletablerte metodikker som oppdragsgiver bruker til dagen og dermed kan bidra med hjelp og kunnskap hvis det er behov for det.

På bakgrunn av at Scrum føltes som beste alternativ for prosjektet og at Azure tilbyr gode verktøy for denne utviklingsmetoden, så ble valget enkelt. Azure har verktøy for å lage scrum board som alle gruppemedlemmene har tilgang til digitalt og kan følge fremgangen av sprinteren kontinuerlig.

Et Scrum-prosjekt har vanligvis en Scrum-master og et utviklingsteam. Gruppen består av tre personer og basert på dette er det lite hensiktsmessig med en dedikert scrum master. Videre ble det bestemt at hver sprint skulle vare i to uker. I enden av hver sprint vil gruppen planlegge et møte med oppdragsgiver for en liten rapport for sprinten og hvordan fremgangen i prosjektet går.

Eksperiment

Gruppen vil, i tillegg til å gjøre utvidede litteratursøk på emne, utføre ulike eksperimenter for å kunne underbygge eller motbevise resultatet av søket. På grunnlag av den gitte tidsrammen har gruppen valgt ut tre eksperiment å jobbe med, herav to utviklings eksperiment og et felteksperiment. Utviklings eksperimentene gruppen ønsker å utføre er å programmere to script som evaluerer påliteligheten på signal sendt med LoRaWAN protokollen. Det ene scriptet skal evaluere signalstyrken og det andre scriptet skal undersøke om og eventuelt hvor ofte datapakker går tapt, mer om dette senere i rapporten. Felteksperimentet er valgt for å faktisk se hvordan signaler oppfører seg og blir påvirket av forskjellige parameter som plassering, støy og hindringer. Dette er et prosjekt gruppen vil få til for å utvide eksperimenteringen til flere aspekter (fysisk) i tillegg til å evaluere sensordata som gruppen har tilgang til.

Analyse

I tillegg til litteratursøk vil gruppen analysere resultatene av eksperimentene for å kunne, ved prosjektslutt, komme med en godt begrunnet konklusjon på om LoRaWAN kan benyttes i tidskritiske applikasjoner. Mer om hvordan gruppen har analysert eksperimentene kommer senere i rapporten.

Reproduserbarhet

Gruppen vil lage et API som henter ut, ved hjelp av riktig tilgang og sertifikater, sensordata fra sensorer benyttet av oppdragsgiver. Koden til API'et kan gjenbrukes, men riktig tilgang trengs. Ved benyttelse av koden vil resultatene være noe annerledes enn det som blir forklart og vist i denne rapporten, for da vil resultatene være signal som kommer fra tiden programmet kjøres som er nye signal. Om noen vil lagre data må det opprettes en storage-account i Azure. Om dette ikke er nødvendig kan applikasjonen også kjøres lokalt.
Da må sertifikater legges i en «.env»-fil og datamaskinen er avhengig av å stå på hele tiden for å kontinuerlig hente data, noe som man slipper i en skyløsning, mer om dette i kapittel 4.1.1.. Scriptene utvikles med mulighet for reproduksjon og for mulighet for å kunne gjøre enda mer avanserte evalueringer.

Felteksperimentet kan alle med tilgang til en Field tester reprodusere. Her kan en også utvide og enda grundigere utføre målinger rundt om i Bergen by. En kan og flytte fokusområdet til innlandet hvor gruppen ikke har fått utført felteksperiment.

3.4.3 Prosjektplan

Planleggingen av bachelorprosjektet gjorde gruppen ved hjelp av et GANTT-diagram. GANTT-diagrammet er vist i tabell 3.2. Diagrammet ble brukt til å lage milepæler for hvert hovedpunkt og tilhørende underpunkter. Denne metoden å planlegge et prosjekt på gir gruppen oversikt over tidsplan og alle oppgaver som skal ferdigstilles før prosjektet ferdigstilles.

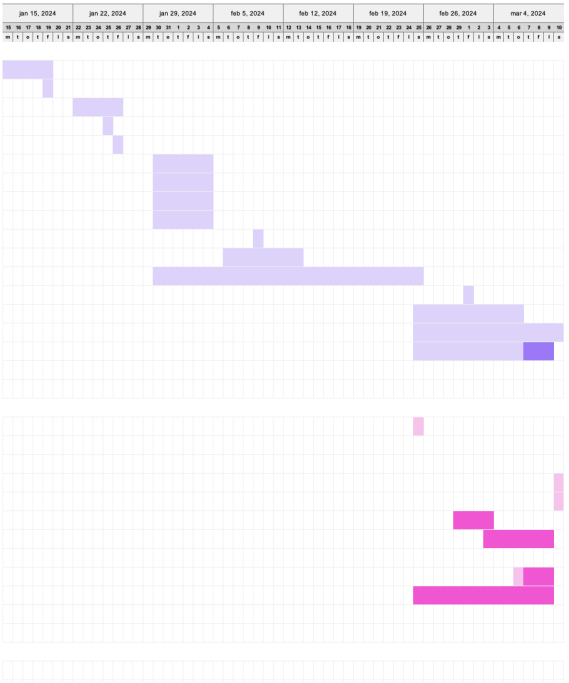
Oppgavene til prosjektet ble som nevnt delt inn i hovedoppgaver. Disse er Dokumentasjon, utvikling og presentasjoner. Som vist i tabell 3.1.

TASK	Time used	START	END
Dokumentasjon			
Intern planlegging av bachelor prosjekt	100 %	1.15.24	1.19.24
Oppstartsmøte med oppdragsgiver (Oblig 1)	100 %	1.19.24	1.19.24
Tolke oppgave og skrive skjema for tolkning til arbeidsgiver	100 %	1.22.24	1.26.24
Statusmøte 1 med Hvl veileder (Oblig 2)	100 %	1.25.24	1.25.24
Godkjenning av tolkning av oppgave fra oppdragsgiver	100 %	1.26.24	1.26.24
Registrere prosjekttittel (En del av Oblig 3)	100 %	1.30.24	2.4.24
Risikoanalyse (En del av Oblig 3)	100 %	1.30.24	2.4.24
Skrive innledning i visjonsdokument (En del av Oblig 3)	100 %	1.30.24	2.4.24
Skrive sammendrag problem og produkt (Visjonsdokument) (En del av Oblig 3)	100 %	1.30.24	2.4.24
Statusmøte 2 med Hvl veileder	100 %	2.9.24	2.9.24
Analyse av tidligere oppgaver (Oblig 5)	100 %	2.6.24	2.13.24
Utkast midtveisrapport pluss dokumentasjon, iterasjon 2 (Oblig 6)	100 %	1.30.24	2.25.24
Statusmøte 3 med Hvl veileder (Oblig 7)	100 %	3.1.24	3.1.24
Midveisrapport ferdig (Oblig 8)	100 %	2.25.24	3.6.24
Diverse dokumenter, iterasjon 3 (Oblig 10)	60 %	2.25.24	4.12.24
Utkast 1 rapport (Oblig 11)	20 %	2.25.24	4.19.24
Utkast 2 rapport (Oblig 12)	0 %	4.19.24	5.3.24
Refleksjonsnotat (oblig 13)	0 %	5.17.24	5.24.24
Utvikling			
Planlegge struktur på API	100 %	2.25.24	2.25.24
Initiere API og få tilgang til oppdragsgivers datapool	100 %	2.25.24	3.10.24
Initiere Azure container for å lagre data fra datapool	90 %	2.25.24	3.10.24
Opprette API - kall for å kunne hente ut data fra datapool	25 %	3.10.24	3.24.24
Skrive test kode for API - kall	25 %	3.10.24	3.24.24
Lage script for å måle signalstyrke	0 %	2.29.24	3.3.24
lage skript for å måle/undersøke tap av datapakker	0 %	3.3.24	3.10.24
Utvikle et script som plotter all data presentert i grafer	0 %	3.10.24	3.17.24
Litteratursøk	7 %	3.6.24	3.20.24
Evaluere	0 %	2.25.24	5.16.24
Presentasjon			
Midtveispresentasjon presentasjon (Oblig 9)	0 %	3.15.24	3.15.24
Sluttpresentasjon	0 %	6.3.24	6.3.24
EXPO	0 %	6.12.24	6.12.24

Tabell 3.1: Hovedoppgaver og tilhørende underoppgaver (Fremgang vist i figuren er fra 18 mars).

Arbeidsflyten og planlagt tidsbruken er visualisert med fargede ruter for hver arbeidsoppgave. Arbeidsflyten er vist i Figur 3.2. Diagrammet viser 8 uker av gangen. Flere bilder som viser diagrammet finnes i vedlegg og i Vedlegg 1.

TASK	Time used	START	END
Dokumentasjon			
Intern planlegging av bachelor prosjekt	100 %	1.15.24	1.19.24
Oppstartsmøte med oppdragsgiver (Oblig 1)	100 %	1.19.24	1.19.24
Tolke oppgave og skrive skjema for tolkning til arbeidsgiver	100 %	1.22.24	1.26.24
Statusmøte 1 med Hvl veileder (Oblig 2)	100 %	1.25.24	1.25.24
Godkjenning av tolkning av oppgave fra oppdragsgiver	100 %	1.26.24	1.26.24
Registrere prosjekttittel (En del av Oblig 3)	100 %	1.30.24	2.4.24
Risikoanalyse (En del av Oblig 3)	100 %	1.30.24	2.4.24
Skrive innledning i visjonsdokument (En del av Oblig 3)	100 %	1.30.24	2.4.24
Skrive sammendrag problem og produkt (Visjonsdokument) (En del av Oblig 3)	100 %	1.30.24	2.4.24
Statusmøte 2 med Hvl veileder	100 %	2.9.24	2.9.24
Analyse av tidligere oppgaver (Oblig 5)	100 %	2.6.24	2.13.24
Utkast midtveisrapport pluss dokumentasjon, iterasjon 2 (Oblig 6)	100 %	1.30.24	2.25.24
Statusmøte 3 med Hvl veileder (Oblig 7)	100 %	3.1.24	3.1.24
Midveisrapport ferdig (Oblig 8)	100 %	2.25.24	3.6.24
Diverse dokumenter, iterasjon 3 (Oblig 10)	60 %	2.25.24	4.12.24
Utkast 1 rapport (Oblig 11)	20 %	2.25.24	4.19.24
Utkast 2 rapport (Oblig 12)	0 %	4.19.24	5.3.24
Refleksjonsnotat (oblig 13)	0 %	5.17.24	5.24.24
Utvikling			
Planlegge struktur på API	100 %	2.25.24	2.25.24
Initiere API og få tilgang til oppdragsgivers datapool	100 %	2.25.24	3.10.24
Initiere Azure container for å lagre data fra datapool	90 %	2.25.24	3.10.24
Opprette API - kall for å kunne hente ut data fra datapool	25 %	3.10.24	3.24.24
Skrive test kode for API - kall	25 %	3.10.24	3.24.24
Lage script for å måle signalstyrke	0 %	2.29.24	3.3.24
lage skript for å måle/undersøke tap av datapakker	0 %	3.3.24	3.10.24
Utvikle et script som plotter all data presentert i grafer	0 %	3.10.24	3.17.24
Litteratursøk	7 %	3.6.24	3.20.24
Evaluere	0 %	2.25.24	5.16.24
Presentasjon			
Midtveispresentasjon presentasjon (Oblig 9)	0 %	3.15.24	3.15.24
Sluttpresentasjon	0 %	6.3.24	6.3.24
EXPO	0 %	6.12.24	6.12.24



Tabell 3.2: GANTT-diagram viser arbeidsflyt de første 8 ukene (Fremgang vist i figuren er fra 18 mars).

3.4.4 Risikovurdering

I prosjektarbeid så medfølges alltid en viss risiko for at noe går galt. Dette kan være mye forskjellig og konsekvensene for hver risiko varierer. Med dette tatt i betraktning, utførte gruppen en risikovurdering basert på en mal fra emnet "DAT191 Bacheloroppgave". En risiko vurderes ved å beskrive risikoen, finne årsak til hvorfor risikoen eventuelt skulle skjedd, sannsynlighet for at risikoen inntreffer, konsekvens om den inntreffer, et risiko-produkt basert på sannsynlighet og konsekvens og til slutt tiltak for å forhindre risikoen.

Sannsynligheten vurderes med en verdi fra 1 til 5 hvor 1 er svært lav sannsynlighet og 5 er svært høy sannsynlighet. På samme måte vurderes konsekvensen med en verdi mellom 1 og 5

hvor 1 er svært lav konsekvens og 5 er svært høy konsekvens. Risiko-produktet er verdi til sannsynlighet multiplisert med verdien til konsekvens.

Risikovurderingen har gruppen fremstilt i den tildelte malen fra emnet og en del er vist i tabell 3.3. Fullstendig risikovurdering finnes i vedlegg til slutt i rapporten. Risikoprodukt-diagram er vist i tabell 3.4. Risikodiagram finnes også i Vedlegg 1.

	Hendelse /Risiko	Årsak	Sannsynlighet	Konsekvens	Risiko-produkt	Tiltak
1	*Applikasjonen blir ikke tatt i bruk.*	*Kunde kan eventuelt lage sin egen applikasjon på oppdragsgivers infrastruktur, og bare bruke deres datapool*	*Høy (4)*	*Svært Lav (1)*	4	*Gjennomføre brukertesting* *Lage en bra nok applikasjon så kunde heller vil bruke den.*
2	*Gruppen klarer ikke levere et produkt som tilfredsstiller kravet for å svare på problemstilling*	*Gruppen har ikke nok teknisk kompetanse* *Gruppen har ikke gjort nok forarbeid* *Gruppen blir overambisiøs og prøver å lage mer enn de klarer å gjennomføre*	*Lav (2)*	*Svært høy (5)*	10	*Jobbe jevnt og lære seg teknologien som blir brukt* *Gjør nok forarbeid og god planlegging* *Sette realistiske mål*

Tabell 3.3: Utdrag fra risikovurderings-diagram

Sannsynlighet						
	Svært Høy	5	10	15	20	25
	Høy (4)	4	8	12	16	20
	Middels (3)	3	6	9	12	15
	Lav (2)	2	4	6	8	10
	Svært Lav	1	2	3	4	5
		Svært Lav (1)	Lav (2)	Middels (3)	Høy (4)	Svært Høy (5)
	Konsekvens					

Tabell 3.4: Risikoprodukt-diagram

3.5 Evalueringsplan

Evaluering er en viktig del av prosjektarbeidet. Hensikten er at en skal kunne lære av hva som gikk bra, men også av hva som gikk mindre bra eller dårlig. En ønsker å skape oversikt over hva som bør taes hensyn til ved et nytt prosjektarbeid.

Det er alltid rom for forbedring og en kan alltid lære av feil en har gjort. Dette kalles erfaring. Sammen med oppdragsgiver har gruppen laget noen delmål som skal hjelpe med å prøve å finne et svar på hovedmålet med prosjektet. Undersøke om LoRaWAN-protokollen er anvendbar i tidskritiske applikasjoner.

Gruppen vil grundig undersøke, ved hjelp av scripts, om signal kommer fram fra sensor til mottaker med riktig signalstyrke eller om signalstyrken svekkes på veien grunnet støy eller andre faktorer. Signalstyrke måles i numeriske tall og kan sammenlignes med hvilke verdier som er ønsket og forventet. Gruppen ønsker og, ved hjelp av scripts, å måle hvor ofte datapakker går tapt innenfor et gitt tidsintervall og deretter evaluere om resultatet opp mot hva bruksområdet skal være. Det å evaluere tall kan ofte være lettere enn å evaluere for eksempel hvor *god* en applikasjon er. Da kreves brukertesting og brukerundersøkelser. Da dette prosjektet ikke er et produktfokusert prosjekt, ville ikke dette vært nødvendig uansett. Metoden gruppen vil evaluere resultatet på trenger ikke nødvendigvis hjelp fra brukere, men kan evalueres av oppdragsgiver for å bestemme om resultatene svarer på hovedoppgaven ved prosjektet.

4 Detaljert løsning

4.1 Design og utvikling

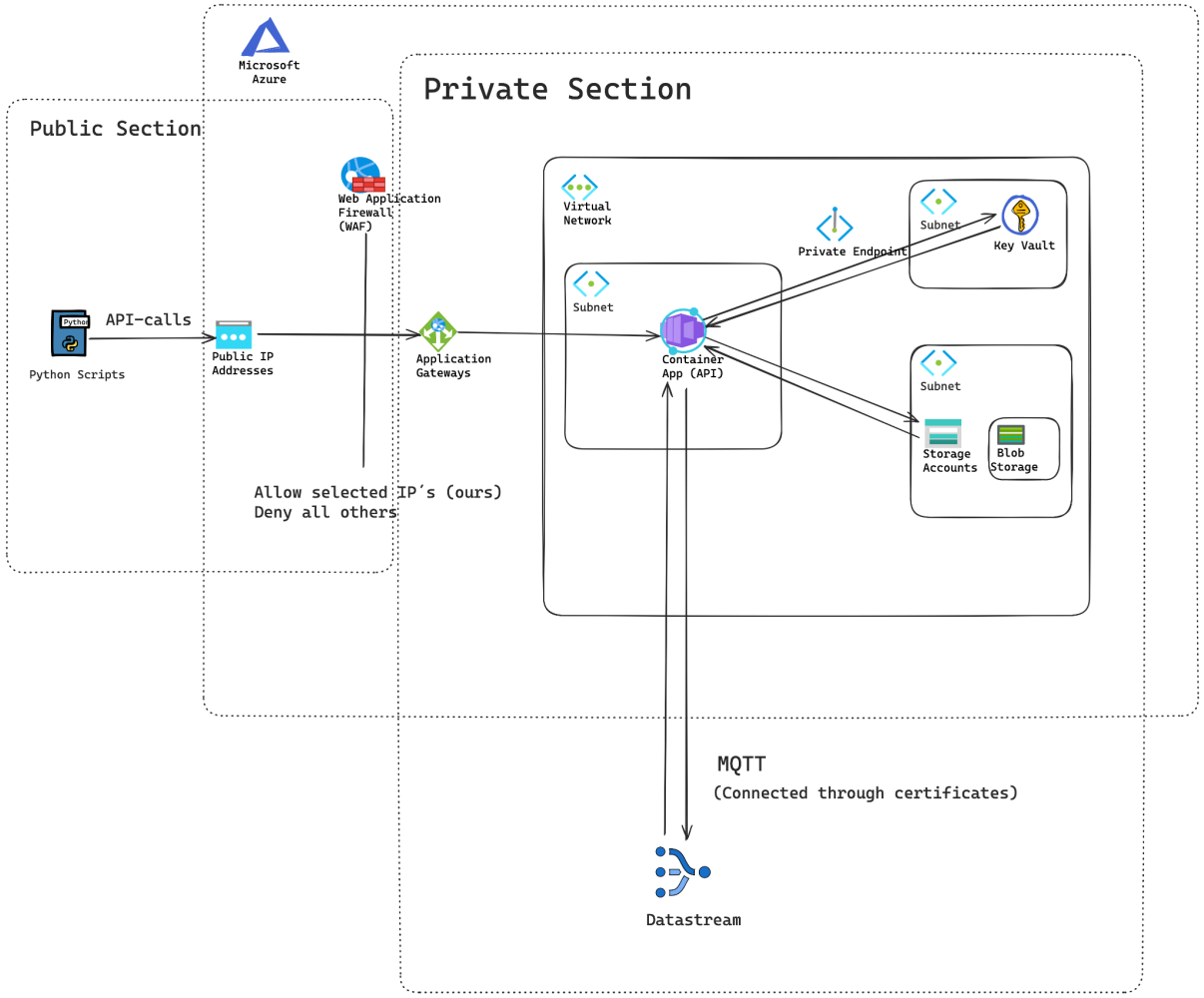
4.1.1 Bakgrunn for design

Gruppen har i tillegg til rapporten, laget systemsdokumentasjonen som beskriver den tekniske løsningen, men vurdert kravsdokumentasjon som ikke nødvendig. Dette fordi løsningen av utviklingsdelen blir beskrevet i detalj i denne rapporten.

Prosjektet har valgt å bruke Azure for å håndtere API-et, lagring og hemmeligheter. Dette er blitt valgt med tanke på å ha en skybasert løsning som gjør at API-et og nedhenting av sensordata skjer hele tiden. Alternativ løsning kunne være å kjøre API-et lokalt på egen maskin for å slippe unna tilganger mellom ressurser, men da ville man ikke nødvendigvis hentet ned data hele døgnet og alle som skulle ha kjørt API-et måtte ha hatt alle hemmeligheter lokalt på egen maskin. Sikkerhetsrisikoen minker også med antall steder hemmelighetene ligger. Azure fokuserer på god sikkerhet for tilganger til de ulike ressursene både internt i Azure miljøet og fra utsiden. Dette er viktig for prosjektet med tanke på at sensordataer og tilganger ikke kommer på avveie.

I selve API-et er det valgt en løsning hvor den “hoster” en egen webserver, henter ned nødvendige hemmeligheter, kobler seg opp mot datastrømmen på MQTT, henter ned dataen og lagrer den i Azure.

4.1.2 Arkitektur



Figur 4.1: Struktur i og utenfor Azure med offentlig og privat seksjon.

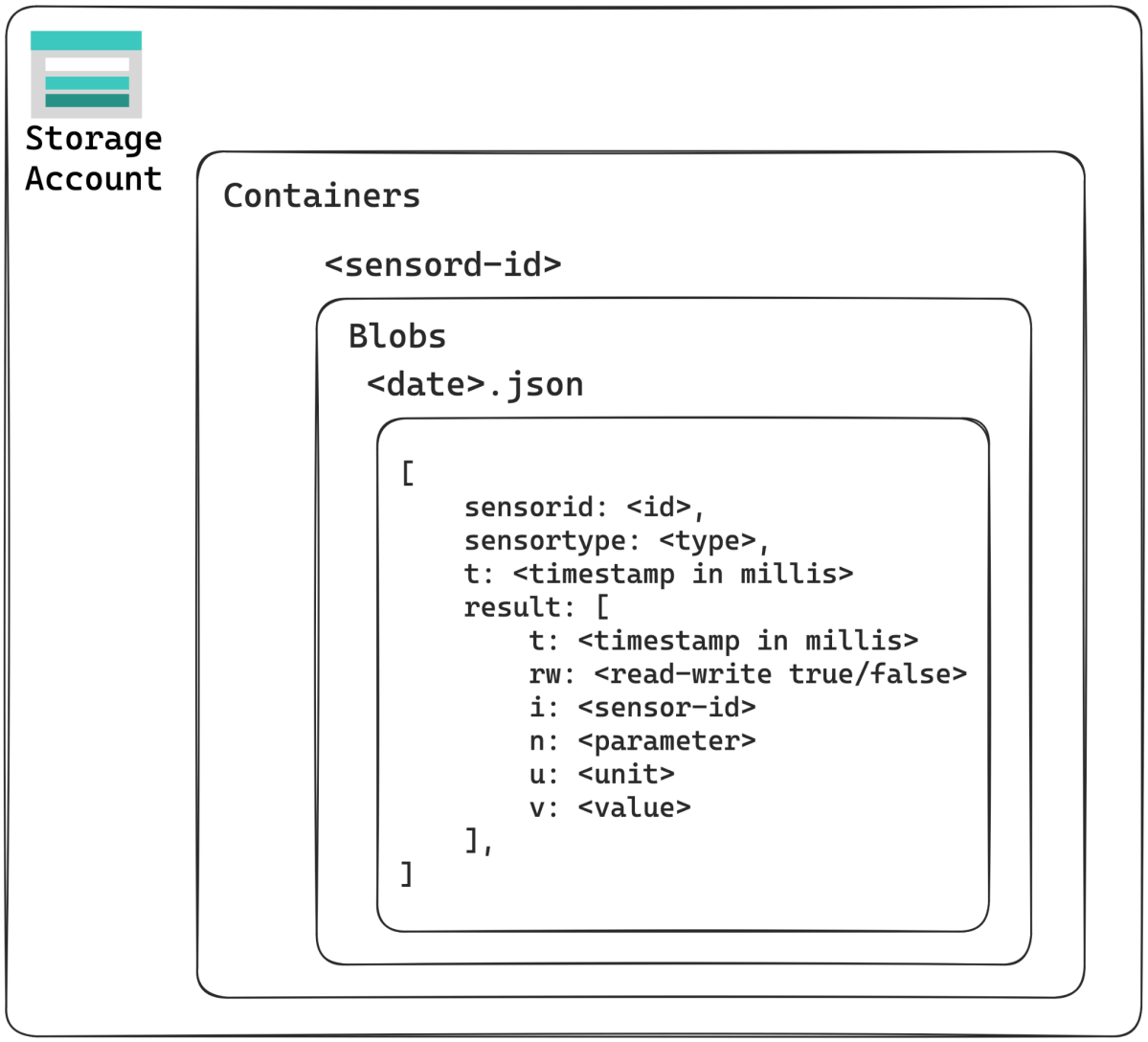
I Azure så er API-et lagt opp i en Container App via github actions som skjer automatisk hver gang det blir pushet til "master"-branchen i github. Container-appen ligger i et subnet under et virtuelt nettverk. Andre nødvendige ressurser har også sine egne subnet i det virtuelle nettverket, som gjør det enkelt for ressurser å snakke med hverandre internt uten å måtte sette opp offentlige tilganger og redusere hvem som kan aksessere hva. For å kunne aksessere API-et så er det satt opp en Application Gateway som har en offentlig IP-adresse ut mot internett. Denne Application Gatewayen vil være offentlig så hvem som helst kan aksessere API-et, så for å redusere tilganger til den offentlige IP-adressen er det satt opp en Web Application Firewall (WAF) som forbyr alle IP-adresser som ikke er like de som er satt opp i

WAF-en. Hele strukturen vises i figur 4.1. Deretter vil python-scriptene kalle på API-et via den offentlige IP-adressen og prosessere dataene og gjøre kalkulasjoner og analyser.

4.1.3 Lagring av sensordata - Storage Account

Storage Account i Azure ble valgt som løsning for lagring av data overfor en database da det egner seg til rask prototyping, enkel lagring av JSON-struktur som datapakkene kommer på og medlemmene av prosjektet ønsker å teste ny teknologi. Oppsettet i en Storage Account er som et filsystem på en datamaskin, det blir opprettet mapper og filer i mappene. Prosjektet har valgt løsningen ved å opprette mapper for hver sensor-id og inne i mappene så er det filer som tilsvarer hver dag, navngitt med datoen. I filene for hver dag så blir det lagt inn en JSON-liste hvor en inngang i listen tilsvarer en måling på et spesifikt klokkeslett på den dagen. Dette kan sees i figur 4.2.

Med å bruke JSON- struktur i lagringen så blir det lettere å hente ut data og bruke denne med en gang. En slipper å måtte drive med veldig mye dataprosessering for å få det på riktig struktur som man måtte gjort hvis man hadde brukt en database.



Figur 4.2: Struktur i Storage Account.

4.1.4 API - Klient

Gruppen har satt opp et API via en application gateway med en offentlig IP-adresse, som er beskyttet av en Web Application firewall. Mot dette APIet kan det gjøres API kall som returnerer sensordata på JSON format. Gruppen ønsker å analysere denne sensor-dataen ved hjelp av python scripts. For å aksessere denne dataen utviklet gruppen en API-klient i python. Denne API klienten vil utføre kall til APIet ved hjelp av Python- pakken: requests. Denne pakken ble valgt å bruke for den lar deg utføre HTTP/1.1 forespørsler ekstremt enkelt, den har egne metoder for å legge til "query"-strenger til urlen din, så du slipper å forme urlen selv. API klienten utfører et HTTPS api kall og får tilbake JSON data. API klienten kan utføre 5 forskjellige api kall. Disse er:

- `fetch_status, fetch_all_sensor_data`
- `fetch_by_sensor_id(id), fetch_by_sensor_id_and_date(id,date)`
- `fetch_by_sensor_id_data_and time Interval(id,date,start Time,endTime)`

Fetch_status vil returnere et JSON objekt som inneholder informasjon om hvor lenge serveren har kjørt.
Fetch_all_sensor_data vil returnere et JSON objekt som inneholder alle sensordata som har blitt plukket opp av applikasjonen vår og lagret i databasen. Fetch_by_sensor_id(id) vil returnere et JSON objekt som inneholder all sensor data for en sensor med gitt id. Fetch_by_sensor_id_and_date(id,date) vil returnere et JSON objekt som inneholder all sensor data for en sensor med gitt id på en gitt dato. Fetch_by_sensor_id_date_and_timeInterval(id,date,startTime, endTime) vil returnere et JSON objekt som inneholder all sensor data for en sensor med gitt id, på en gitt dato mellom gitt starttid og sluttid.

4.1.5 Testkode

Gruppen har i prosjektet prøvd å skrive mest mulig testkode for å få høyest mulig testdekning for å forsikre om at programmene oppfører seg som ønsket. Dette er gjort ved å skrive enhetstester til nesten hver metode. Det var ikke alle testene som funket da det ble kjørt gjennom github actions på grunn av manglende tilganger opp til Azure.

4.2 LoRaWAN i tidskritiske applikasjoner

I dette kapittelet valgte gruppen å fokusere på tre hovedpunkter, helse, industri og annen litteratur om LoRaWAN som valg av teknologi. Dette ble fokusområdene for denne delen av prosjektet grunnet prosjektets tidsperspektiv og tid som trengtes til andre deler av prosjektet. Da gruppen består av tre medlemmer ble det også valgt tre hovedpunkter av logiske grunner. Kapittelet ser på hvordan LoRaWAN kan brukes i de forskjellige sektorene, men også hvordan de tidskritiske aspektene av sektorene kan eller kan ikke benytte LoRaWAN system.

I tillegg til resultatene vil det også bli presentert i dette kapittelet hvordan gruppen kom frem til resultatet.

4.2.1 LoRaWAN i Helse

Litteratursøket i denne delen ble delt opp i ulike fokusområder. Dette er områder som kan sies å være tidskritiske innenfor helsesektoren. Som nevnt tidligere, oppdaget gruppen at engelske søk gav flest resultater. Første steg i dette kapittelet var å finne ut hva som gjør at LoRaWAN kan være et bra valg av IoT-system innenfor helse med fokus på de tidskritiske aspektene. I denne delen var det ønskelig å se hva som er praksis i dag med LoRaWAN som IoT-system i helsesektoren. Med følgende søkestreng: (LoRaWAN) AND (IoT smart healthcare) AND (practices) gav det noen resultater på Google. Artikkelen "*7 Best Practices for Leveraging LoRa in IoT Smart Healthcare*" ble tatt i bruk i denne delen av kapittelet da den gav et overordnet bilde av LoRaWAN i helsesektoren på et generelt plan. Google ble også brukt til neste del som omhandler utrykningstjenester og tidskritisk kommunikasjon med helsehjelp ved ulykker og lignende. Her ble søkestrengen: (LoRaWAN) AND (critical messaging) benyttet og gav artikkelen "*Is LoRaWAN a viable option for critical messaging?*" fra nettstedet Issuu. Artikkelen er opprinnelig fra "Critical Comms". Søkestrengen gir tilgang til artikkelen fra begge nettstedene. I delen som omhandler eldrehjelp ønsket gruppen å finne hjelpemidler som benytter LoRaWAN-teknologien. Her ble søkestrengene: (LoRaWAN) AND (GPS watch) AND (senior care) og (LoRaWAN) AND (emergency button). Resultatet av disse søkene ga artikkelen "*LoRaWAN GPS Smartwatch for Smart Senior Care"* og eksempel på panikk-knapp i "*LoRaWAN®-Based Panic Button - LW004-PB*". Siste delen i dette kapittelet omhandler bruken av LoRaWAN på sykehus. Søkestrengen som ga resultatene i denne delen var: (LoRaWAN) AND (Smart healthcare) AND (use). Dette ga artiklene *"Leveraging LoRa in IoT Smart Healthcare"* og igjen *"LoRa in IoT Smart Healthcare: 7 Best Practices to Know"*

Hvorfor helse er tidskritisk

Helsesektoren kan beskrives som tidskritisk på bakgrunn av at tiden det tar å få hjelp kan være avgjørende for utfallet av en skade, sykdom eller annen form for helsemessig hjelp personer skulle trenge.

LoRaWAN er på papiret svært egnet innenfor helsesektoren på grunn av dens lave strømforbruk og med det lave kostnader, lange rekkevidde og bred dekning. LoRaWAN benytter seg dessuten av en spesiell spredninsteknologi som tillater terminaler med forskjellige spredningssekvenser å sende meldinger på samme frekvens uten å forstyrre hverandre. Dette er en evne som IoT-enheter ønskes å ha når det sendes mange meldinger samtidig, for eksempel på et sykehus. Et annet positivt aspekt hvorfor LoRaWAN kan være et godt IoT-system i helsesektoren er den billige infrastrukturen og muligheter for å integrere med andre teknologier (Gerard, 2022).

Utrykning/akutt hjelp

Når en nødsituasjon oppstår og vi mennesker trenger akutt hjelp er det avgjørende at kontakt med nødetater blir opprettet fortløpende. Dette kan være med å avgjøre utfallet av situasjonen. Denne typen kontakt kalles “critical messaging" som oversatt til norsk betyr kritisk melding. Denne meldingen bør mottas av nødetater innen ti sekunder eller mindre. Teknologi som sørger for at nødetatene får melding innen ti sekunder må være pålitelige og ha høy datahastighet. Teknologi som ikke er pålitelig eller at datapakker har en tendens til å gå tapt, kan ha fatale konsekvenser ved en nødsituasjon. Derfor er det så viktig å forske på LoRaWAN sin pålitelighet og effektivitet før en vurderer protokollen i applikasjoner eller systemer som tilkaller nødetater.

Et eksempel på hvor LoRaWAN protokollen benyttes for å kontakte nødetater, her brannvesenet, er i Frankrike, nærmere bestemt de franske Alpene. Denne brannavdelingen dekker 6925 kvadratkilometer bestående av 154 500 innbyggere og består av 1548 frivillige brannmenn.

Hver brannmann bærer en enhet som kalles Birdy Slim IoT og blir varslet på denne enheten via standard POCSAG-paging. Når melding mottas , sendes en bekreftelse tilbake over LoRaWAN-nettverket. Deretter kan den mottakende brannmannen svare med en melding om han eller hun responderer på meldingen eller ikke til CAD (Computer Aided Design)-operatøren slik at det er tydelig hvor mange som responderer på brannen eller ikke.
Hvis bekreftelsen ikke mottas av CAD-en, sendes meldingen deretter ut via LoRaWAN i tilfelle pageren var utenfor POGSAG-rekkevidde. I tillegg til dette blir GPS-posisjonen sendt via LoRaWAN for å vise respondentens posisjon i forhold til brannstasjonen når en alarm blir sendt ut (Welch, 2022).

I dette eksempelet kan en se at LoRaWAN benyttes sammen med andre teknologier for å sikre at respondenten får melding selv om en bekreftelse ikke blir sendt til CAD-en. I tillegg blir LoRaWAN protokollen benyttet til GPS måling på grunn av dens lange rekkevidde og pålitelighet.

Aldershjem og eldrehjelp

Noen eldre i dagens samfunn klarer ikke på lik linje som alle andre å ta vare på seg selv. Noen trenger hjelp til dagligdagse ting grunnet fysisk for, andre kan på grunn av sykdommer som demens og Alzheimers, ha behov for å bli passet på av helsepersonell og til slutt så kan noen ha behov for hjelp ved fall osv. Disse punktene er alle forskjellige behov for hjelp, men tiden det tar for å få hjelp kan beskrives som kritisk. Derfor er eldrehjelp en tidskritisk ordning.

Alzheimer og demens er en sykdom som kan ramme eldre. Denne sykdommen rammer mennesker forskjellig og noen trenger ofte mer hjelp enn andre. Det som ofte går igjen med mennesker som har denne sykdommen er stedsans. Om en bor på et demenssenter eller om en er i tidlig stadium av sykdommen og bor hjemme, så kan en gå utenfor huset og miste fatning om hvorfor en dro eller hvor en er. Her kan LoRaWAN benyttes i form av GPS-sporing. På grunn av ens lange rekkevidde og lave energiforbruk kan eldre med Alzheimer eller demens bruke for eksempel en klokke som benytter seg av protokollen til å sende GPS-lokasjon av bæreren.

LoRaWAN GPS smartklokke, som leveres av blant annet Hiotech.net, er et eksempel på en enhet som kan hjelpe å lokalisere eldre som har gått seg bort og samtidig gir bæreren mulighet til å be om hjelp hvis en ikke vet hvor en er eller faller på turen. Pårørende kan se lokasjonen til bæreren og samtidig se tilstand til bærer som ved fall og bærer ikke har mulighet til å tilkalle hjelp fra klokken, kan pårørende eller de som har ansvar for vedkommende se om personen har falt eller har skadet seg. Enheten brukt i dette eksempelet benytter LoRaWAN siden rekkevidden er lang og da det er snakk om lite datamengde i en GPS-lokasjon eller SOS-melding, er LoRaWAN på papiret et fornuftig valg av teknologi. Ifølge hiotech.net så har klokken en ukes batteritid, dette på grunn av lavt energiforbruk på teknologi (Hiotech, n.d.).

Noen eldre bor på eldrehjem og behovet for kontinuerlig GPS-sporing er kanskje ikke nødvendig. Her kan LoRaWAN sin teknologi også komme til nytte. Beboerene på eldrehjemmet kan få tildelt en panikk-knapp som kan benyttes ved fall eller annet behov for hjelp. Når en beboer trykker på panikk-knappen så vil en uplink sendes til nærmeste gateway og denne gatewayen vil ved mottatt signal fra en beboer varsle personell om hendelsen og de kan respondere på meldingen. Noen panikk-knapper sender også lokasjonen hvor knappen blir trykket på, for eksempel MOKO Smart sin LW004-PB (MOKO SMART, 2023).

Andre måter LoRaWAN protokollen kan brukes i sammenheng med eldrehjelp er å benytte seg av sensorer som måler temperatur slik at ikke de eldre skal fryse. En annen sensor som kan være nyttig er dør-sensor som måler når noen drar, da kan pårørende eller sykepleiere varsles om beboerne ikke har kommet tilbake når en skal. Disse punktene viser at LoRaWAN har et bredt bruksområde, men punktene er ikke nødvendigvis tidskritiske, derfor går ikke rapporten nærmere inn på disse bruksområdene.

Sykehus

Sykehus bruker i dag mye IoT-systemer for å utføre ulike målinger og for å ha kontroll på pasienter og ansatte. IoT-systemer blir benyttet for å kunne sende beskjeder, lokasjoner og alarmer til flere sentraler og fagfolk som drar nytte av informasjonen. LoRaWAN er et godt alternativ til bruk på sykehus.

Mange målinger og overvåkinger gjøres hver dag og flere ganger om dagen på sykehus. Noen tidskritiske og noen ikke. Eksempel på ikke-tidskritiske målinger og overvåking hvor LoRaWAN kan være effektivt er smart energi overvåking. Dette er overvåking over energibruk på et sykehus. Med god overvåking kan ressurser effektivt bli benyttet for å senke energibruken og dermed minimere kostnader for sykehuset. Et annet eksempel på ikke-tidskritisk overvåking kan være å holde styr på medisinsk avfall og for å sørge for best mulig avfallsbehandling som igjen kan være med på å mer effektivisere ressurser sykehuset har til behandling av avfall. Rapporten går ikke dypere inn på disse punktene, men gruppen vil få frem et poeng om at LoRaWAN kan benyttes til andre ikke-tidskritiske aspekter innenfor helsesektoren og.

Når det kommer til målinger og overvåkinger på sykehus hvor tiden er kritisk så er det mange. For eksempel skal medisin ha en spesifikk temperatur for å ikke bli ødelagt eller

miste sin ønskede funksjon. Derfor er det svært viktig at kjøleskap eller kjølerommene har riktig temperatur til enhver tid. LoRaWAN kan benyttes til å overvåke temperaturen og gi alarm til sentralen eller en enhet fagpersonell kan bære, om temperaturen går over en gitt grenseverdi. Dette gjør at fagpersonell kan respondere raskt og mulig redde medisin for potensielt veldig mye penger, for som en vet er medisin dyrt. I dette eksempelet vil lang rekkevidde være viktig for å sikre at riktig personell blir nådd ved temperaturendringer. LoRaWAN er et godt alternativ på grunn av dens lange rekkevidde og mulighet for sanntidsmåling som kan overvåkes av fagpersoner, pluss mulighet med alarm sending ved nådd grenseverdi på temperaturen i kjøleskap eller kjølerom (Gerard, 2022)..

Et annet eksempel hvor LoRaWAN kan være nyttig på et sykehus er mulighet for å overvåke helsetilstanden hos pasienter. På en side kan en pasient få utdelt en trykknapp på samme måte som i kapittelet om eldrehjelp. Her kan pasienten trykke om noe ikke er som det skal og kan på den måten få tilsyn raskt på helsepersonell. LoRaWAN sin mulighet til å gi sanntidsmålinger, gir helsepersonell mulighet til å overvåke tilstand til pasienter og kan respondere raskt om noe skjer med pasienten som gjør at vedkommende trenger tilsyn eller hjelp (MINEWSEMI, 2022).

Hvorfor LoRaWAN kan brukes og er pålitelig i helsesektoren

"Smart helse" er, ifølge Grand view Research, estimert til å vokse fra en 153,6 mrd. industri til en 484,4 mrd. industri i 2030 (Gerard, 2022).

Dette tyder på at behovet for gode IoT-løsninger er ønskelig og ettertraktet i helsesektoren. Som nevnt tidligere i rapporten stiller LoRaWAN teknologien sterkt i denne økningen av smarte løsninger innenfor helse på grunn av rekkevidde og lavt energiforbruk. For å gjøre LoRaWAN systemet mest mulig pålitelig må infrastrukturer settes opp på en optimalisert måte for å dra best mulig nytte av systemet. Dette skrives mer om i kapittel 4.3.3. hvor gruppen har testet hva som avgjør god kontakt og klare signal fra gateway. Systemet vil kunne bidra til pålitelig måling og overvåking, lavere energiforbruk og driftskostnader, optimalisere andre ressurser og samtidig være med å bidra til en smartere hverdag for helsepersonell.

4.2.2 LoRaWAN i Industri, logistikk og transport

I dette delkapittelet er det gjort litteratursøk om LoRaWAN innenfor industri, logistikk og transport. For å finne kilder om industri er det blitt gjort søk i Google på: "LoRaWAN industrial and time critical" for å få opp relevante kilder som handler om LoRaWAN i industrien og tidskritisk. Her er det blitt gjort søk med AND-operatoren. Søket er gjort på engelsk da det ikke er like mange kilder om LoRaWAN på norsk som gjør at man får større utslag på søket. Det samme er gjort for logistikk og transport, "LoRaWAN logistics and time critical" og "LoRaWAN transport and time critical". Oppslagene som slår ut på logistikk og transport inneholder mange av de samme kildene, så man ser at logistikk og transport i LoRaWAN går om hverandre.

Industri:

LoRaWAN i industri kan beskrives som tidskritisk ved at det er avgjørende hvor lang tid det tar å fikse opp i hendelser som at en fryserdør står oppe, strømbrudd, lav og høy temperatur i rom og gasslekkasjer. Her er det ikke like tidskritisk som i helse med tanke på liv og død, men det krever at det ikke går alt for lang tid før det blir fikset opp i for å unngå økonomiske tap eller andre problem.

I industrien er det ofte bygg med tykke vegger laget av materialer som kan blokkere pakker, men det gjør ingenting om noen datapakker går tapt her eller der, da man bare trenger å se tendensen til at f.eks en fryser blir varm. Her kan det f.eks brukes KI for å predikere hva pakken skulle ha vært.

IIoT, Industrial-IoT, begynner å bli stadig mer i bruk i verden til overvåkning av bygg, måling av strømbruk, smartbyer, smart industri, og olje og gass. I New York ser man bruk av LoRaWAN til industriell bruk, der det er satt opp LoRaWAN for å overvåke gasslekkasjer og antall gasslekkasjer som stoppes har blitt redusert med 6 ganger mer av hva det var tidligere (Fremont, C, 2023).

Her er tidsintervallet for tidskritisk medium høyt da dette ikke må fikses opp i med en gang, men er noe som er viktig å gjøre før det eventuelt er en flamme i nærheten og gassen tar fyr.

Arduino, som er en stor produsent og bidragsyter for utdanningssektoren som støtter oppunder utviklere og innovatører, begynner nå å få sine enheter inn på LoRaWAN teknologien. De satser på å få flere til å begynne med IIoT, derfor har de lagd skreddersydde

sensorer for krevende miljøer (The Things Industries-a, 2023). I denne artikkelen er det ikke noe bevis på at LoRaWAN kan funke i tidskritiske applikasjoner, men det viser at LoRaWAN i industrien øker betraktelig de neste årene og tidskritisk vil være mer relevant å se mer på.

Logistikk / transport:
Tidskritisk i logistikk og transport handler om utvalgte varer som har en streng frist, dette kan være en pasient som trenger medisiner eller utstyr for å bli frisk, eller varer som er tidssensitive. Dette er veldig viktig for varer som f.eks er innenfor helse, der det er blodprøver, medisiner og lignende som ikke tåler å være ute av kulden over en tidsperiode.

LoRaWAN teknologien er veldig lovende for logistikk og transport med tanke på sin lave energibruk, lang kommunikasjons avstand og tillater sikker og bidireksjonal kommunikasjon (The Things Industries-b, 2023).
På grunn av at logistikk og transport, som i industrien, har store bygninger med tykke vegger og gjerne også er under bakken, er det behov for en signal extender / repeater for å gjøre at det er dekning inne i byggene. Dette ble observert i felt (kommer i kapittel 4.3.3) da det ble målt signalstyrke under bakken, inne i høye bygg og lignende, som viste at under bakken er det vanskelig å få dekning uten en extender / repeater.

I India er tidskritisk logistikk og transport ganske sentralt, men India står overfor ulike utfordringer, inkludert utilstrekkelig infrastruktur, mangel på standardisering og begrenset teknologiadopsjon (LinkedIn, 2023). For å overkomme denne utfordringen har logistikken i India blitt endret til teknologi og innovasjon. De bruker teknologi som IoT GPS-tracker, lagerstyringssystemer og kunstig intelligens til blant annet å redusere leveringstid og optimalisere kjøreruter. Med LoRaWAN GPS-tracker kan man få oppdateringer på transporten i sanntid, noe som gjør at man kan følge med på om den kjører rett vei og om det eventuelt skulle skje noe på veien. Skulle det skje noe på veien vet man nøyaktig hvor man skal rykke ut og hjelpe.

På varehus kan man sette opp et smartlager for å hjelpe med logistikken. Der kan man sette opp temperaturmålere i hvert rom for å se at rommene har riktig temperatur, bevegelsessensorer for å se om det er innbrudd i lageret, vibreringssensor for rasfare inne på lageret og en dørsensor for å se om en dør er åpen eller lukket. Dette er små overvåkninger som hjelper de ansatte og varsler fort om det skjer noe.

Hvorfor LoRaWAN i industri, logistikk og transport?

I følge Abhresh Sugandhi vil IoT løsninger øke sikkerhet og effektivitet innen industri, logistikk og transport. Dette er noe LoRaWAN er en veldig god kandidat for. I industri, logistikk og transport må sensorene være på steder langt unna eller steder som er vanskelige å komme til. Med LoRaWAN sin lange rekkevidde og gjennomtrengningsevne vil det være en bra løsning for problemet (TWTG, 2023).

4.2.3 Annen litteratur om LoRaWAN

I dette kapittelet er det blitt utført litteratursøk for å finne relevant litteratur om bruk av LoRaWAN i tidskritiske applikasjoner. Målet med dette søket var ikke bare å finne grunner til å bruke LoRaWAN, men også finne potensielle problemer som kan oppstå når man bruker LoRaWAN. Databasen som har blitt søkt i er Google. Den første søkestrengen som ble brukt var "pros and cons using LoRaWAN in time critical applications". Da ble artiklene "*What are the advantages and disadvantages of using the LoRaWAN network?*" og "*Is LoRaWAN a viable option for critical messaging?*" funnet. Videre siden dette kapittelet ønsker å fokusere på utfordringer med å bruke LoRaWAN ble det søkt på "*disadvantages LoRaWAN*" da ble artikkelen "*LoRaWAN*" funnet.

Det finnes mange motargumenter for å bruke LoRaWAN i tidskritiske applikasjoner. Mange av fordelene med LoRaWAN som langveis kommunikasjon og lavt strømbruk bærer med seg ulemper som kan gjøre det utfordrende å bruke LoRaWAN i tidskritiske applikasjoner.

> *"LoRaWAN sitt "lytt før du snakker"-prinsipp, introduserer ventetid i kommunikasjonen, noe som gjør den mindre egnet for sanntids applikasjoner som krever øyeblikkelig svar"* (Macnam, 2023, gruppemedlem sin oversettelse).

I mange tidskritiske applikasjoner vil rask kommunikasjon være viktig. Som for eksempel ved trykknappen diskutert i kapittel 4.2.1 er det veldig viktig at beskjeden om at denne knappen er trykket på kommer raskt frem. En annen ting som kan føre til problemer med bruk av LoRaWAN er spekteret de sender meldinger på.

> "*Siden LoRaWAN opererer i med ulisensierte frekvensbånd, er nettverket mottakelig mot interferens fra andre enheter som bruker samme spekteret, noe som potensielt kan påvirke nettverks ytelse og stabilitet*" (Macnam, 2023, gruppemedlem sin oversettelse).

I tidskritiske applikasjoner er det viktig at man kan stole på at alle sendte meldinger kommer frem dit de skal, og at det er korrekt melding som kommer fram. LoRaWAN sin bruk av åpne frekvensbånd som også kan bli brukt av andre enheter kan føre til uforutsette ulemper som vil være uønskelig i en tidskritisk applikasjon. LoRaWAN er ikke egnet for store data payloads. Payload begrenset til 0.3 - 5.5kBps for bruk i noen tidskritiske applikasjoner vil ikke dette være et problem, som for eksempel med denne helse trykk knappen da payload i denne meldingen ikke er så stor. (Meaney, n.d.) Denne begrensningen kan derimot bli et problem innenfor andre tidskritiske applikasjoner som krever større payload.

> "*Akkurat nå har den kanskje ikke samme dekning som andre teknologier og har en maks melding lengde på 80-120 karakterer*" (Welch, 2022, gruppemedlem sin oversettelse).

LoRaWAN er ikke designet for kontinuerlig overvåkning, utenom klasse C enheter. Det finnes 3 klasser LoRaWAN sensorer klasse A,B og C. Av disse er det kun klasse C som vil fungere til kontinuerlig overvåkning, som for eksempel til et paging system til nødetater. Klasse C LoRaWAN sensorer har støtte for kontinuerleg overvåkning, men har på grunn av dette dårligere batteritid (Meaney, n.d.).

4.3 Eksperiment

4.3.1 Eksperimentering med signalstyrke

4.3.1.1 Signal-to-noise ratio

Signal-to-noise ratio er som navnet tilsier, forholdet mellom signal og støy. Signal-to-noise ratio er ofte beskrevet som SNR eller S/N. SNR er en måling som sammenligner signalnivået med støy-nivået, og er ofte målt i måleenheten (dB). Det er ønskelig med en høy SNR verdi, da dette betyr at det er rikelig med brukbar informasjon i signalet, selv med støy tatt i betraktning (Altunian, 2021).

Støy kan kalles for "uønsket informasjon" og oppstår vanligvis av elektromagnetisk felt eller annen form for elektronikk. Støy forekommer ofte når det snakkes om signaler og sensorer, da sensorer er bygget opp av elektronikk. Hvis støy-verdien blir for høy kan dette "overdøve" signal-informasjonen og mottaker kan risikere å ikke kunne tolke signalet slik det var tenkt. Dette er grunnlag til at SNR er en parameter viktig å ta i betraktning når en skal jobbe med å undersøke pålitelighet og brukbarhet til signaler.

4.3.1.2 Received Signal Strength Indicator

Received Signal Strength Indicator, referert til som rssi, er et mål på hvor sterkt signalet mottatt fra senderen er. Rssi er målt i desibel-milliwatt (dBm), og verdien er gitt som negativ verdi. Det er ønskelig med en verdi nær null. En rssi verdi nær null betyr at mottatt signal er sterkt (Thethingsnetwork-a, n.d).

Rssi er viktig når en skal undersøke hvor godt mottaker kan "høre" signalet fra senderen. LoRa sin dokumentasjon forteller at et minimumskrav for at mottaker skal kunne "høre" et signal, er -120 dBm, men høyere er ønskelig. En rssi verdi på for eksempel -30 dBm betyr at det er et svært godt mottakelig signal (LoRa, 2018).

4.3.1.3 Spreading factor

Spredning faktoren (SF) kontrollerer symbolene og dermed hastigheten på dataoverføringen. Høy SF verdi gir bredere frekvensspredning, noe som gjør signal mer robust og mindre mottakelig for støy og andre forstyrrelser. Høy SF betyr dog tregere datahastigheter. Lav SF verdi gir lavere frekvensspredning, men datahastigheten er høyere (Thethingsnetwork-b, n-d).

LoRa benytter seg av spredning faktorene SF7 til SF12. SF12 gir høyere spredning og lavere hastighet på dataoverføring. SF7 gir lavere spredning, men høyere hastighet på dataoverføring (Thethingsnetwork-c, n-d).

4.3.1.4 Script som måler og evaluerer signalstyrke

En del av evaluering av kvaliteten på signal som sendes med LoRaWAN protokoll var å lage python script. Scriptet anvender SNR-verdier, rssi-verdier og ser hvilken SF som benyttes, og

sammenligner disse med ønskede grenseverdier. Mer om evalueringsmetoden kommer i kapittel 5.1.1.

Som mer beskrevet i kapittel 4.1.4, så ble det utviklet en API-klient som henter ut data live fra Azure applikasjonen til gruppen etter hvert som data blir sendt fra sensor.

Denne klienten ble brukt i scriptet laget for å evaluere og visualisere signalstyrke-verdiene. I Github-repsitoryet ble det opprettet en mappe som gruppen kalte “Signalstyrke”. Denne mappen inneholder to filer. En main.py-fil og en main.ipynb-fil.

Main.py-filen benytter API-klienten til å hente ut sensordata som inneholder nyttig informasjon som SNR, rssi og SF for hvert signal. Deretter sammenlignes disse verdiene med grenserverdier som beskriver hva som er et godt og “hørbart” signal. Disse sammenligningene blir gjort ved hjelp av if-setninger som avgjør hvilken “klasse” signalet tilhører, vist i figur 4.3. Deretter blir resultatet av hvert signal fra en sensor skrevet ut i terminalen med forskjellige kommentarer ut fra de ulike verdiene på signalet. SF målingen sjekker om signalet er innenfor gitte SF-verdier. Her forventes samtlige signal at de er innenfor, men det kan likevel være interessant å se hva for spreading faktorer som blir benyttet. Rssi målingene sjekker om rssi-verdien er innenfor minimumsgrensen og om styrken på mottatt signal er veldig sterk. Disse klassifiseres i: “RSSI value is lower than minimum”, “RSSI value is ok” og “RSSI is good, the signal is strong”. Til slutt sjekker også scriptet snr verdien og gir basert på verdiene følgende tilbakemeldinger: “SNR value is to low and the signal is to corrupted to demodulate”, "SNR is lower than 0 but, LoRa can demodulate the signal" og "SNR is good, the received signal is less corrupted".

```
for sensor_data in sensor_data_list:
      for item in sensor_data["result"]:

          if item["n"] == "SF": #For elsys sensorer
              sf = item["v"]
              if sf < sf_min or sf > sf_max:
                  print("SF is out of range", sf)
              elif sf >= sf_min and sf <= sf_max:
                  print("SF is in range", sf)

          if item["n"] == "rssi":
```

```
                rssi = item["v"]
                if rssi < rssi_minimum:
                    print("RSSI value is lower than minimum", rssi)
                elif rssi > rssi_minimum and rssi < rssi_good:
                    print("RSSI value is ok", rssi)
                elif rssi > rssi_good:
                    print("RSSI is good, the signal is strong", rssi)

 if item["n"] == "snr":
                snr = item["v"]
                if snr < snr_minimum:
                    print("SNR value is to low and the signal is to
corrupted to demodulate", snr)
                elif snr > snr_minimum and snr < 0:
                    print("SNR is lower than 0 but, LoRa can demodulate
the signal", snr)
                elif snr > snr_wanted:
                    print("SNR is good, the recieved signal is less
corrupted", snr)

                print("")
```

Figur 4.3: IF-setningene, fra main.py i signalstyrke mappen, som sjekker hva for klasse signalene tilhører ut ifra måleresultat.

Main.ipynb benytter samme API-klient klassen for å hente ut sensordata. Scriptet anvender Matplotlib-biblioteket for å kunne visualisere data. Scriptet som er skrevet i en jupyter notebook henter ut ønskede verdier fra sensordataen, herav SNR, rssi og SF og sammen med grenseverdiene skriver ut et diagram for hver verdi. Her plotter skriptet hver signal og deres verdier i diagrammet og visualisere alle tilgjengelige signal fra Azure applikasjonen. Grenseverdien visualiseres med en stiplete oransje strek. Med grenseverdien visualisert som en strek så er det lett for gruppen å se hva for signal som er innenfor ønskede grenser og hva for signal som er utenfor. Gruppen har valgt å gi diagrammene en overskrift som sier tydelig hva for verdi diagrammet plotter og en egen farge slik at det er lett å skille de forskjellige diagrammene fra hverandre.

4.3.1.5 Evalueringsmetode Signalstyrke

Utvikling av script for å evaluere kvaliteten på signalstyrken trenger threshold-verdier eller grenseverdi. Dette er verdier som det ønskes at målte verdier er innenfor.

SNR er en måleverdi på forholdet mellom signal og støy. Mer om SNR finnes i kapittel 4.3.1.1. En SNR-verdi på null kalles for "støy-bunnen" og verdier over null er over bunnivået og verdier under null er under bunnivået. Vanligvis vil signalet ikke kunne tolkes eller "høres" om SNR-verdien er under bunnivået, altså SNR < 0. LoRaWAN-teknologien kan derimot tolke signal hvor SNR-verdien er under null.
LoRaWAN kan demodulere eller gjenvinne signal med SNR verdi helt ned til SNR = - 20. Typiske SNR-verdier ved LoRaWAN signaler er mellom -20dB og +10dB (LoRa, 2018).
-20dB er dermed brukt som grenseverdi for å evaluere kvaliteten på signal gruppen henter fra datastrømmen.

RSSI som kan leses mer om i kapittel 4.3.1.2, er et mål på styrken til signalet som mottas. Som nevnt tidligere så måles rssi i negative verdier, hvor en verdi nærmere null betyr sterkere signal. Minimumskravet for at signalet som mottas skal være sterkt nok er -120 dBm. Derfor brukes -120dBm som grenseverdi i evalueringen for å kunne konkludere om signal hentet fra datastrømmen mottas sterkt nok til å kunne anvendes.

LoRa benytter seg i hovedsak av seks spreading factor-verdier. Disse verdiene er SF7 til SF12 og alle i mellom disse ytterpunktene. Det forventes at sensordata som gruppen mottar, alle benytter en av disse SF-verdiene uten unntak. Gruppen velger likevel å undersøke spreading factor, for å se hvilke av verdiene som oftest er i bruk. Som nevnt i kapittel 4.3.1.3 så betyr lav SF lavere frekvensspredning og høyere datahastighet, og høy SF betyr større frekvensspredning og lavere datahastighet. Vanligvis benytter LoRa seg av høyere SF når signalet er svakt.

Sensor data som hentes ut fra datastrømmen gruppen har fått tilgang til, sender informasjon om overordnede verdier. Med dette ville gruppen lage et python-script som henter ut disse verdiene fra sensordata som mottas. Disse verdiene sammenlignes med de gitte grenseverdiene. Det skrives ut hvor *bra* et signal er ut fra verdiene på signalet og dette blir i

tillegg visualisert i en graf sånn at en lett kan se hvordan faktiske signaler stilles i forhold til grenseverdier.

4.3.2 Eksperimentering med tap av datapakker

4.3.2.1 Pakketap

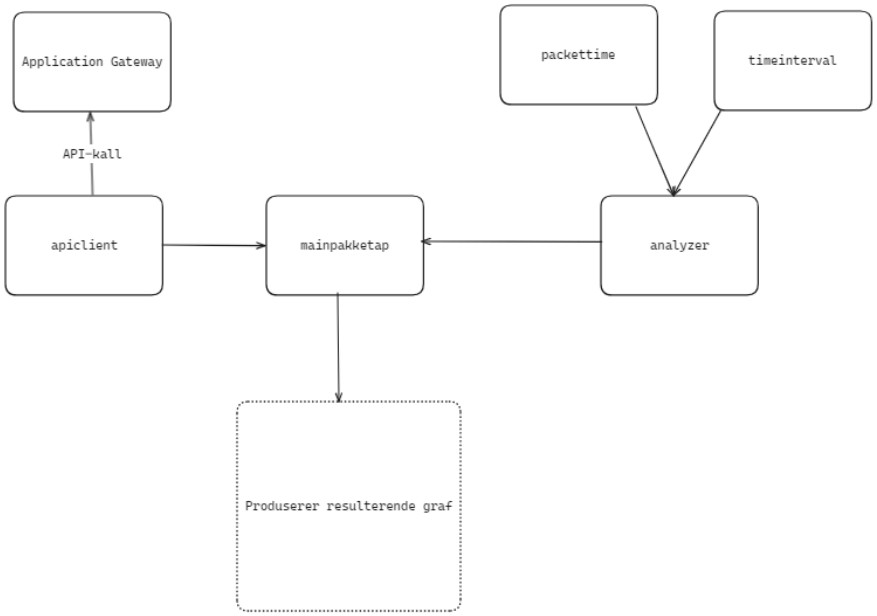
Pakketap er når en eller flere sendte pakker, ikke når destinasjonen sin (Gillis,2021). I en tidskritisk applikasjon som for eksempel til bruk i en nødknapp er dette noe som må unngås. Pakketap kan ha flere årsaker. De mest normale årsakene til at pakketap skjer er dårlig signalstyrke, system-støy, programvarefeil og overbelastede nettverksnoder. (Gillis,2021). Konsekvensene av pakketap i en tidskritisk applikasjon vil være veldig betydelig. Da en tidskritisk applikasjon er en applikasjon som krever at data som skal behandles blir behandlet er det veldig viktig at alle pakker og dermed all data kommer frem og kan bli behandlet av applikasjonen.

4.3.2.2 Script for måling av pakketap

Gruppen ønsker å analysere hvor mye pakketap som finnes ved bruk av LoRaWAN sensorer. Gruppen har fått tilgang på en kontinuerlig datastrøm med sensordata fra LoRaWAN sensorer som Eidsiva har ute hos kunder. Gruppen har også satt opp en service som lagrer alle sensordata i en database og lar deg utføre API-kall for å hente ned lagret data på JSON format. En lagring vil være på denne formen, grundigere vist i figur 4.2:

```
{
 "i": <sensor-id>,
 "sensortype": <sensor-type>,
 "t": <timestamp>,
 "result": []
}
```

Målet med scriptet er å bruke denne "t" verdien som representerer tidspunktet pakken ble sendt på for å finne ut hvor mange pakker som har gått tapt i sensor dataene som gruppen har lagret. Siden vi for noen utvalgte sensorer vet hvor ofte det skal sendes pakker, kan disse to verdiene brukes til å finne ut om en pakke har gått tapt eller ikke.



Figur 4.4: Overordnet arkitektur av hvordan pakketap resultat produseres.

For å hente ned sensordata, analysere denne dataen og deretter visualisere resultatet av analysen, er det utviklet en python-applikasjon. Denne python-applikasjonen er bygd opp av forskjellige klasser som jobber sammen, dette er visualisert i figur 4.4. Klassene som er utviklet er de to hjelpeklassene packettime og timeinterval. Klassen som er ansvarlig for selve analysen av sensordata er analyzer. Klassen ansvarlig for å hente ned sensordata som er lagret er apiclient. I mainpakketap kjøres all koden som trengs for å analysere sensordata og produsere resulterende graf.

Packettime er et objekt som representerer en sensormåling. Packettime klassen har to attributter. Disse to attributtene er id og timestamp. Id representerer en sensor-id, timestamp representerer tidspunktet da pakken ble sendt.

```
    def __init__(self, id: str, timestamp: int):
        self.id = id
        self.timestamp = timestamp
```

Figur 4.5: Viser konstruktøren for packettime klassen

TimeInterval er et objekt som har tre attributter: hours, minutes og seconds. Som vist i figur 4.5.

```
    def __init__(self, hours: int, minutes: int, seconds: int):
        self.hours = hours
        self.minutes = minutes
        self.seconds = seconds
```

Figur 4.6: viser konstruktøren for timeinterval klassen

Dette blir brukt i analyzer klassen for å sjekke om pakker blir sendt med riktige intervaller. For eksempel dersom en vet at en sensor skal sende en pakke hvert tiende minutt, lages et TimeInterval objekt som representerer ti minutter og så blir dette brukt i analyzer klassen.

Analyzer er en python klasse som har i oppgave å analysere JSON data og finne ut hvor mye pakketap som er i dataen. Klassen inneholder metodene:

- `get_packets_before(self, JsonData, time_interval)`
- `get_number_of_lost_packets(packets_before)`
- `get_timestamps(data)`

Klassen inneholder hjelpemetoden `get_time_stamps(data)`, vist i figur 4.6. Denne metoden går igjennom den gitte json dataen og omgjør hver måling til et packettime objekt og returnerer dermed en representasjon av python dataen som en liste med packettime objekt.

```
    def get_timestamps(self, data) -> List[PacketTime]:
        timestamps = []
        for item in data:
            i = item['i']
            t = item['t']
            timestamps.append(PacketTime(i, t))

        return timestamps
```

Figur 4.7: viser get_time_stamps() metoden. Denne metoden tar in json data og returnerer en liste med packettime objekt

Klassen inneholder også en hjelpemetode for å konvertere et TimeInterval til millisekund, som vist i figur 4.8, dette er nødvendig fordi "t" attributten til sensorene representerer et tidspunkt ved å være antall millisekund siden 1.Januar 1970.

```
    # Takes in a dateTime object and returns an int
 def from_timeinterval_to_int(self, timeinterval: TimeInterval) -> int:
        hours_to_milliseconds = timeinterval.hours * 3600000
        minutes_to_milliseconds = timeinterval.minutes * 60000
        seconds_to_milliseconds = timeinterval.seconds * 1000

        return hours_to_milliseconds + minutes_to_milliseconds +
seconds_to_milliseconds
```

Figur 4.8: viser `from_timeinterval_to_int()` metoden, denne metoden tar inn et timeinterval objekt og returnerer verdien av timeinterval objektet i millisekunder.

Metoden `get_packets_before(JsonData, TimeInterval)` , vist i figur 4.9, tar inn et sett med Json data og et timeinterval objekt. JsonData er den dataen som blir analysert. TimeInterval objektet representerer hvor ofte det skal være sendt pakker.

```
     # Returns a list of PacketTime objects
    # The list that is returned contains the PacketTime objects that
were sent
    # before a packet was lost
    def get_packets_before(self, json_data, time_interval:
TimeInterval) -> List[PacketTime]:
        packet_times = self.get_timestamps(json_data)
        packets_before: List[PacketTime] = []
        prev_entry: PacketTime | None = None

        for packet_time_obj in packet_times:
            if (prev_entry is not None and
(packet_time_obj.get_timestamp() - prev_entry.get_timestamp() >

self.from_timeinterval_to_int(time_interval))):
                difference: int = packet_time_obj.get_timestamp() -
prev_entry.get_timestamp() //
self.from_timeinterval_to_int(time_interval)
                if (difference > 1):
                    for _ in range(0, difference):
                        packets_before.append(prev_entry)
                else:
```

```
                    packets_before.append(prev_entry)

            prev_entry = packet_time_obj

        return packets_before
```

Figur 4.9: Viser `get_packets_before()` metoden, denne metoden tar inn Json data og returnerer en liste med packettime objekt. Disse packettime objektene er de pakkene som ble sendt før det gikk tapt en pakke.

Metoden starter med å hente ut en liste av packettime objekt fra den gitte Json dataen. Deretter startes en gjennomgang av dataene, for hvert objekt i listen sjekkes det om tiden mellom den nåværende pakken og forrige pakke er større enn det angitte intervallet er større enn det angitte tidsintervallet. Dersom forskjellen er større enn angitt legges objektet til packetsBefore listen. Dersom det er gått tapt mer enn en datapakke mellom den forrige og den nåværende datapakken, vil den forrige datapakken legges til flere ganger. Til slutt returneres packets_before listen som inneholder de pakkene som ble sendt før det gikk tapt pakker.

Klassen inneholder også metoden `get_number_of_lost_packets(packets_before)`, vist i figur 4.10. Denne metoden tar inn listen som returneres av get_packets_before() metoden og returnerer antall elementer i listen, dette vil representere antall tapte pakker.

```
    # Takes in a list of packet_times and returns the number of
    # entries in the list
 def get_number_of_lost_packets(self, packets_before: List[PacketTime])
-> int:
        return len(packets_before)
```

Figur 4.10: viser `get_number_of_lost_packets()` metoden, denne tar inn en liste og returnerer lengden av listen.

Til slutt har vi MainPakketap scriptet, vist i figur 4.11 til 4.13. Dette scriptet bruker analyzer klassen og apiclient klassen for å analysere et sett med sensorer der gruppen har fått vite fra oppdragsgiver hvor ofte de skal sende pakker.

Scriptet starter med å definere id til de sensorene som skal analyseres.

```
# Timeinterval 1H
sensorid_1 = "0004a30b00eddc20"
# Timeinterval 30 min
sensorid_2 = "a81758fffe06a058"
# Timeinterval 10 min
sensorid_3 = "a81758fffe090a28"
# Timeinterval 10 min
sensorid_4 = "a81758fffe090a2d"
# Timeinterval 10 min
sensorid_5 = "a81758fffe091918"
# Timeinterval 10 min
sensorid_6 = "a81758fffe091919"
# Timeinterval 10 min
sensorid_7 = "a81758fffe094c79"
# Timeinterval 10 min
sensorid_8 = "a81758fffe094c7a"
```

Figur 4.11: viser starten på MainPakketap. Her defineres de forskjellige sensorene.

Deretter blir all data som er lagret på de forskjellige sensorene hentet ned ved hjelp av API klienten.

```
client = APIClient()



json_data_sensor_1 = client.fetch_by_sensor_id(sensorid_1)
json_data_sensor_2 = client.fetch_by_sensor_id(sensorid_2)
json_data_sensor_3 = client.fetch_by_sensor_id(sensorid_3)
json_data_sensor_4 = client.fetch_by_sensor_id(sensorid_4)
json_data_sensor_5 = client.fetch_by_sensor_id(sensorid_5)
json_data_sensor_6 = client.fetch_by_sensor_id(sensorid_6)
json_data_sensor_7 = client.fetch_by_sensor_id(sensorid_7)
json_data_sensor_8 = client.fetch_by_sensor_id(sensorid_8)
```

Figur 4.12: viser hvordan json data for de forskjellige sensorene hentes

Deretter opprettes en instans av analyzer klassen og dataen for alle sensordataene analyseres .

```
analyserer = Analyzer()
one_hour_interval = TimeInterval(1, 0, 40)
half_hour_interval = TimeInterval(0, 30, 40)
```

```
ten_minute_interval = TimeInterval(0, 10, 40)


packets_before_sensor_1: List[PacketTime] =
analyserer.get_packets_before(json_data_sensor_1, one_hour_interval)
packets_before_sensor_2: List[PacketTime] =
analyserer.get_packets_before(json_data_sensor_2, half_hour_interval)
packets_before_sensor_3: List[PacketTime] =
analyserer.get_packets_before(json_data_sensor_3, ten_minute_interval)
packets_before_sensor_4: List[PacketTime] =
analyserer.get_packets_before(json_data_sensor_4, ten_minute_interval)
packets_before_sensor_5: List[PacketTime] =
analyserer.get_packets_before(json_data_sensor_5, ten_minute_interval)
packets_before_sensor_6: List[PacketTime] =
analyserer.get_packets_before(json_data_sensor_6, ten_minute_interval)
packets_before_sensor_7: List[PacketTime] =
analyserer.get_packets_before(json_data_sensor_7, ten_minute_interval)
packets_before_sensor_8: List[PacketTime] =
analyserer.get_packets_before(json_data_sensor_8, ten_minute_interval)


number_lost_packets_sensor_1: int =
analyserer.get_number_of_lost_packets(packets_before_sensor_1)
number_lost_packets_sensor_2: int =
analyserer.get_number_of_lost_packets(packets_before_sensor_2)
number_lost_packets_sensor_3: int =
analyserer.get_number_of_lost_packets(packets_before_sensor_3)
number_lost_packets_sensor_4: int =
analyserer.get_number_of_lost_packets(packets_before_sensor_4)
number_lost_packets_sensor_5: int =
analyserer.get_number_of_lost_packets(packets_before_sensor_5)
number_lost_packets_sensor_6: int =
analyserer.get_number_of_lost_packets(packets_before_sensor_6)
number_lost_packets_sensor_7: int =
analyserer.get_number_of_lost_packets(packets_before_sensor_7)
number_lost_packets_sensor_8: int =
analyserer.get_number_of_lost_packets(packets_before_sensor_8)

percent_lost_packets_sensor_1: float =
analyserer.get_percentage_of_packet_loss(json_data_sensor_1,
packets_before_sensor_1)
```

```
percent_lost_packets_sensor_2: float =
analyserer.get_percentage_of_packet_loss(json_data_sensor_2,
packets_before_sensor_2)
percent_lost_packets_sensor_3: float =
analyserer.get_percentage_of_packet_loss(json_data_sensor_3,
packets_before_sensor_3)
percent_lost_packets_sensor_4: float =
analyserer.get_percentage_of_packet_loss(json_data_sensor_4,
packets_before_sensor_4)
percent_lost_packets_sensor_5: float =
analyserer.get_percentage_of_packet_loss(json_data_sensor_5,
packets_before_sensor_5)
percent_lost_packets_sensor_6: float =
analyserer.get_percentage_of_packet_loss(json_data_sensor_6,
packets_before_sensor_6)
percent_lost_packets_sensor_7: float =
analyserer.get_percentage_of_packet_loss(json_data_sensor_7,
packets_before_sensor_7)
percent_lost_packets_sensor_8: float =
analyserer.get_percentage_of_packet_loss(json_data_sensor_8,
packets_before_sensor_8)
```

Figur 4.13: viser hvordan analyzer klassen blir brukt for å finne antall tapte pakker og hvor mange prosent pakker som går tapt for hver sensor.

4.3.2.3 Evalueringsmetode Tap av datapakker

Gruppen ønsker å evaluere om antall tapte pakker som går tapt er for høy til at LoRaWAN kan brukes innenfor tidskritiske applikasjoner. Hvor mye pakketap som er akseptabelt varierer fra applikasjon til applikasjon. For de fleste applikasjoner regnes pakketap på 1-2% som akseptabelt(Gillis,2021), men tidskritiske applikasjoner krever noen gang lavere.

4.3.3 Måle signal med signalmåler i Bergen

Tirsdag 09.04.2024 dro gruppen til Bergen sentrum for å måle signalstyrken fra gateway i Bergen. Dette eksperimentet gikk ut på å teste hvordan plassering av sensorer i forhold til

gateway har noe å si for signalstyrken. Gruppen definerte først hovedpunktene som skulle sjekkes, mer om dette under, deretter ble det laget en plan på hvordan eksperimentet skulle foregå. Hvert hovedpunkt skulle sjekkes fra flere steder, det vil si at gruppen ville se hvordan signalstyrken er for eksempel under bakkeplan fra flere steder. Dette for å sjekke om et resultat er tilfeldig for akkurat denne plassen eller om det måles like resultater flere steder for hvert hovedpunkt. Til slutt ble gruppen enig i å trykke på knappen av Field testeren fem ganger og se på et overordnet snitt hvordan signalstyrken er på denne plassen.

Eksperimentet ble delt inn i fire hovedpunkter:

- **Nærhet til gateway**

  Her stod gruppen så nær gateway (Gikk ut i fra gateway på Ulrikken) som mulig i Bergen sentrum. Gruppen gikk også opp på taket på parkeringen på Bergen busstasjon. Poenget med denne testen var å se hvordan signalverdiene varierte når de ikke hadde noen tydelig "synlig" forstyrrelse.

- **Mellom bygg (Ikke klar bane til gateway)**

  Gruppen ønsket også å se hvor mye bygninger hadde å si for hvor bra signalet er. Derfor spaserte gruppen rundt om i Bergen sentrum og målte. Noen ganger bak et stort bygg, andre ganger bak flere bygg i forhold til gateway.

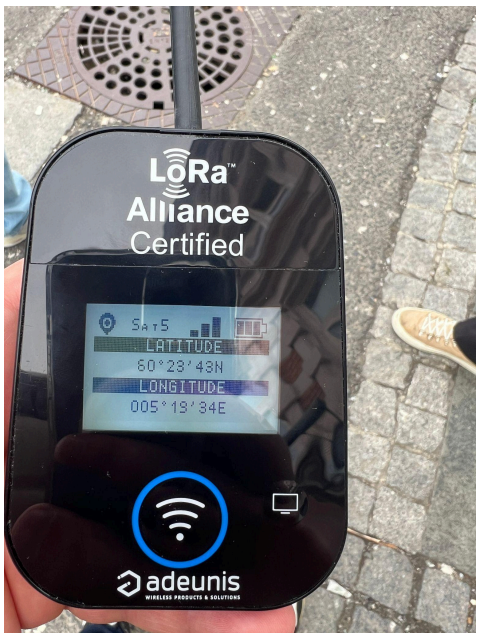
- **Under bakkenivå**

  Gruppen mistenkte at signalet blir noe svakere eller ikke eksisterende om en sensor befinner seg under bakkeplan. Gruppen gikk derfor ved flere anledninger i underetasjen på bygg for å teste. Eksempel på steder som er under bakkeplan er underetasjen på Galleriet og klostergarasjen.

- **Utenfor rekkevidde (Gardermoen)**

  Gruppen ville også se hvordan signalet er om en befinner seg utenfor dekningskartet til LoRaWAN. Spørsmålet gruppen stilte seg var om det var ingen signal eller om det var et svakt signal.

Gruppen benyttet en dekningsmåler som heter Field Test Device (ARF8123AA), vist i figur 4.14. Denne oppfører seg som en sensor og sender en datapakke til nærmeste gateway og gir den ut verdi på signalstyrke.



Figur 4.14: Bilde av Field testeren gruppen benyttet til eksperimentet.

Gjennomføring av eksperiment

Utstyr:

Field Test Device (ARF8123AA)

Lokasjoner:

- I midten av torgallmenningen,
- Bergen sentrum.
- 3.etasje Galleriet Bergen sentrum.
- 6. etasje Galleriet Bergen sentrum.
- 0.etasje Galleriet Bergen sentrum.
- Backstube bryggen Bergen.
- Rema 1000 Kløverhuset.
- Klostergarasjen.
- Rådhuset Bergen sentrum.
- Toppen av parkeringshuset Bergen bystasjon.

Framgangsmåte:

Gruppen tok med seg måleren til alle de forskjellige lokasjonene. Dermed ble det notert ned klokkeslett og det ble trykket 5 ganger på knappen ved hver lokasjon. Senere ble dataene hentet ned fra en portal og analysert.

5 Resultater

5.1 Prosjektresultat

5.1.1 Resultat av litteratursøk

Gjennom bachelorprosjektet våren 2024 er det blitt utført litteratursøk for å bedre forstå LoRaWAN sine bruksområder samt utforske om det er gjort tidligere arbeid med LoRaWAN som teknologi brukt i tidskritiske applikasjoner og sektorer.
Gruppen erfarte tidlig i denne prosessen at det er lite omfattende nasjonal litteratur på området. Dette er forståelig med bakgrunn i figur 2.5. fra kapittel 2.4.5. Dermed ble det gjort større søk i internasjonale artikler, noe som gav mer omfattende litteratur på området gruppen ville se nærmere på. Da LoRaWAN ble lansert i 2015, er det en relativt ny protokoll, men en kan se at den allerede har mange bruksområder og samtidig kan en se at protokollen kan være et godt valg i mange av dagens industrier. Smartbyer og smarte løsninger vil være et økende satsningsområde rundt om i verden i nærmeste og mer fjern fremtid. LoRaWAN viser gode tendenser til pålitelighet og energibruk.

LoRaWAN har mange områder der protokollen er i bruk eller kan taes i bruk. I helsesektoren er det mange områder hvor tiden er kritisk, eksempelvis så er det viktig at pasienter får tidlig hjelp samtidig som medisiner er dyrebart og behandling og oppbevaring av disse må gjøres på riktig måte for å unngå store økonomiske tap. Her viser LoRaWAN protokollen seg til å på papiret være et godt valg til system på grunn av påliteligheten og energiforbruket.

Industri- og transportsektoren kan også benytte seg av LoRaWAN protokollen på grunn av dens lange rekkevidde. GPS sporing kan være tidskritisk for at viktige pakker og leveranser skal komme frem til riktig tid, men også at pakker og lignende skal finnes igjen hvis disse forsvinner eller går tapt.

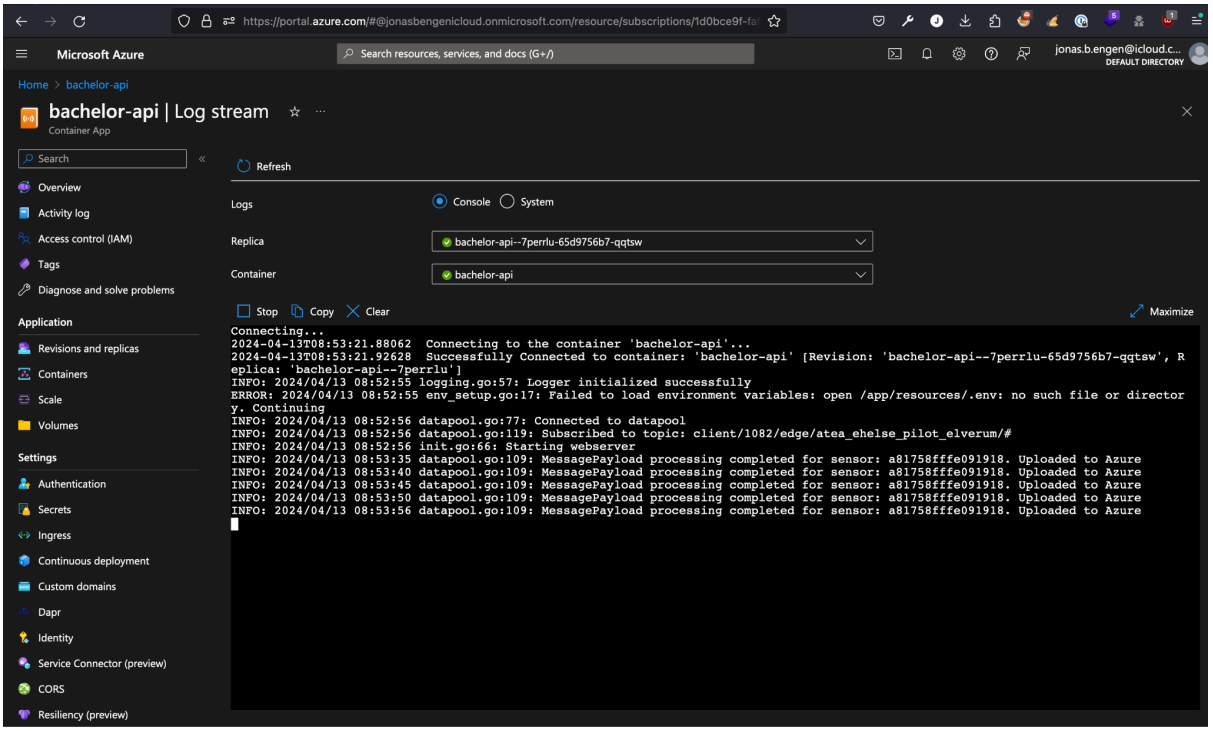
Gruppen har funnet ut at LoRaWAN har bra potensialet og kan brukes i tidskritiske applikasjoner så lenge størrelsen på data er lav. Når det kommer til tidskritiske systemer og funksjoner er det ofte snakk om alarmer, GPS-lokasjon eller små meldinger. Dette gjør at LoRaWAN er et utmerket valg av teknologi. Valg av klasse i systemet bør ved alarm og

sos-signal være klasse C for å alltid kunne motta signal. Til mindre tidskritiske funksjoner kan en bruke de mer energibesparende klassene A eller B.

Gruppen har og funnet ut at Norge har liten infrastruktur (men stadig økende) til bruk av teknologien. Infrastrukturen må bygges ut for å øke avstandene teknologien kan brukes for å dekke større områder. Infrastruktur bygges av oppdragsgiver Eidsiva da de ønsker å utvide bruken av LoRaWAN teknologien. Per nå er områdene begrenset i hovedsak til Bergen og innlandet, men som en ser i andre land så er mye større del av landet dekket. Litteratursøket har gitt gruppen grunn til å tro at LoRaWAN er en teknologi verdt å se på når en tenker på å sette opp nye IoT systemer for tidskritiske applikasjoner.

5.1.2 API som henter ut data fra kontinuerlig datastrøm

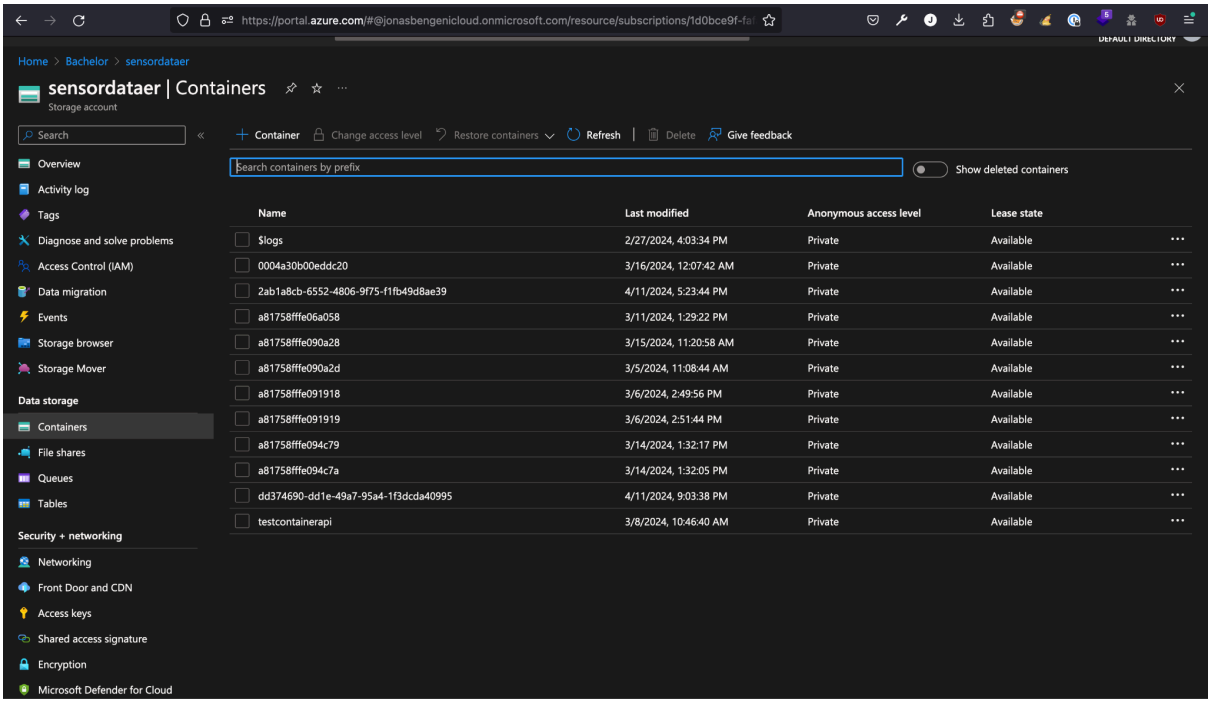
Gruppen lagde et API som henter ut signaldata fra en datastrøm for å kunne se nærmere på ytelse og pålitelighet for LoRaWAN. API-et som henter ut data fra en kontinuerlig MQTT-strøm kjøres med å skrive kommandoen “cd /app” og “go run ./cmd”. Da vil programmet initialisere loggeren, hente ned environment fil med nødvendige hemmeligheter, noe den ikke vil gjøre når den ligger i azure siden hemmelighetene ligger da i et keyvault, koble seg opp til MQTT-datastrømmen med hemmeligheter som ligger i environment fil eller i keyvault og subscribe på riktige topics. Etter dette vil den starte en egen webserver som gjør det mulig å kunne gjøre API-kall.
Som dere ser i figur 5.1 når det kommer sensordata fra MQTT-strømmen blir den prosessert og lastet opp i azure. Når opplastingen er ferdig vil den logge dette i konsollet.



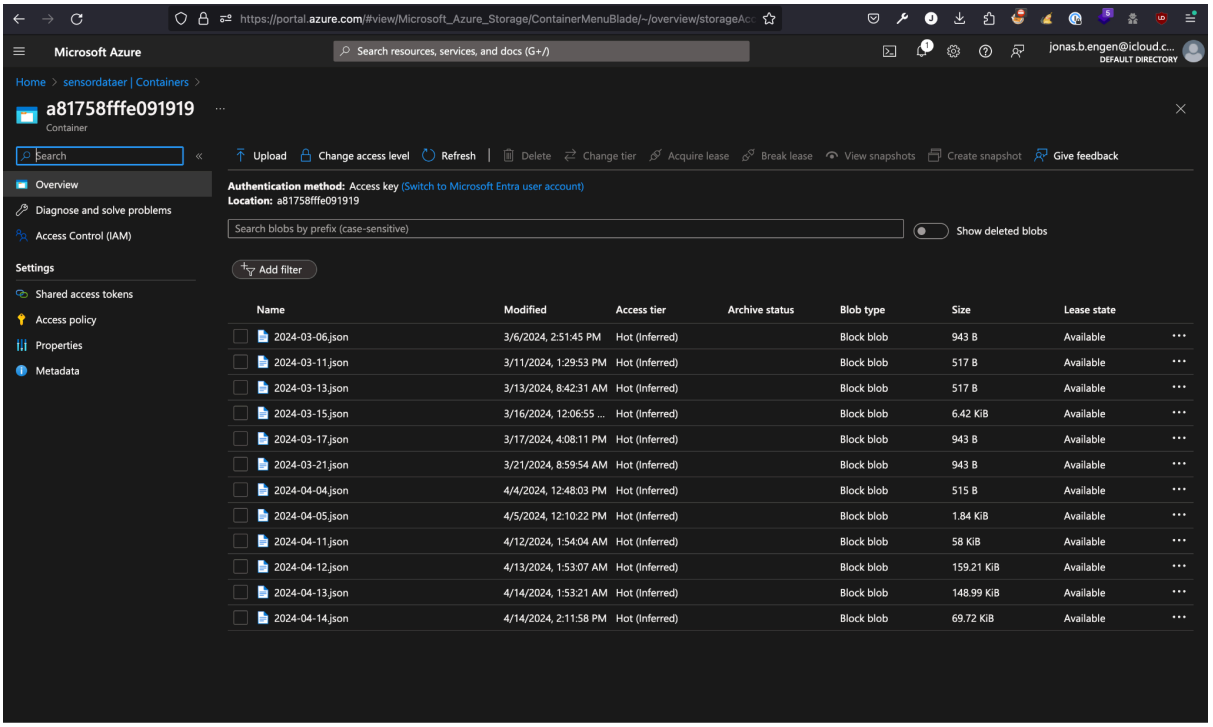
Figur 5.1: Utklipp fra terminalen i Azure ved oppstart av API og mottakelse av sensordata fra kontinuerlig datastrøm

5.1.3 Data lagret i Azure

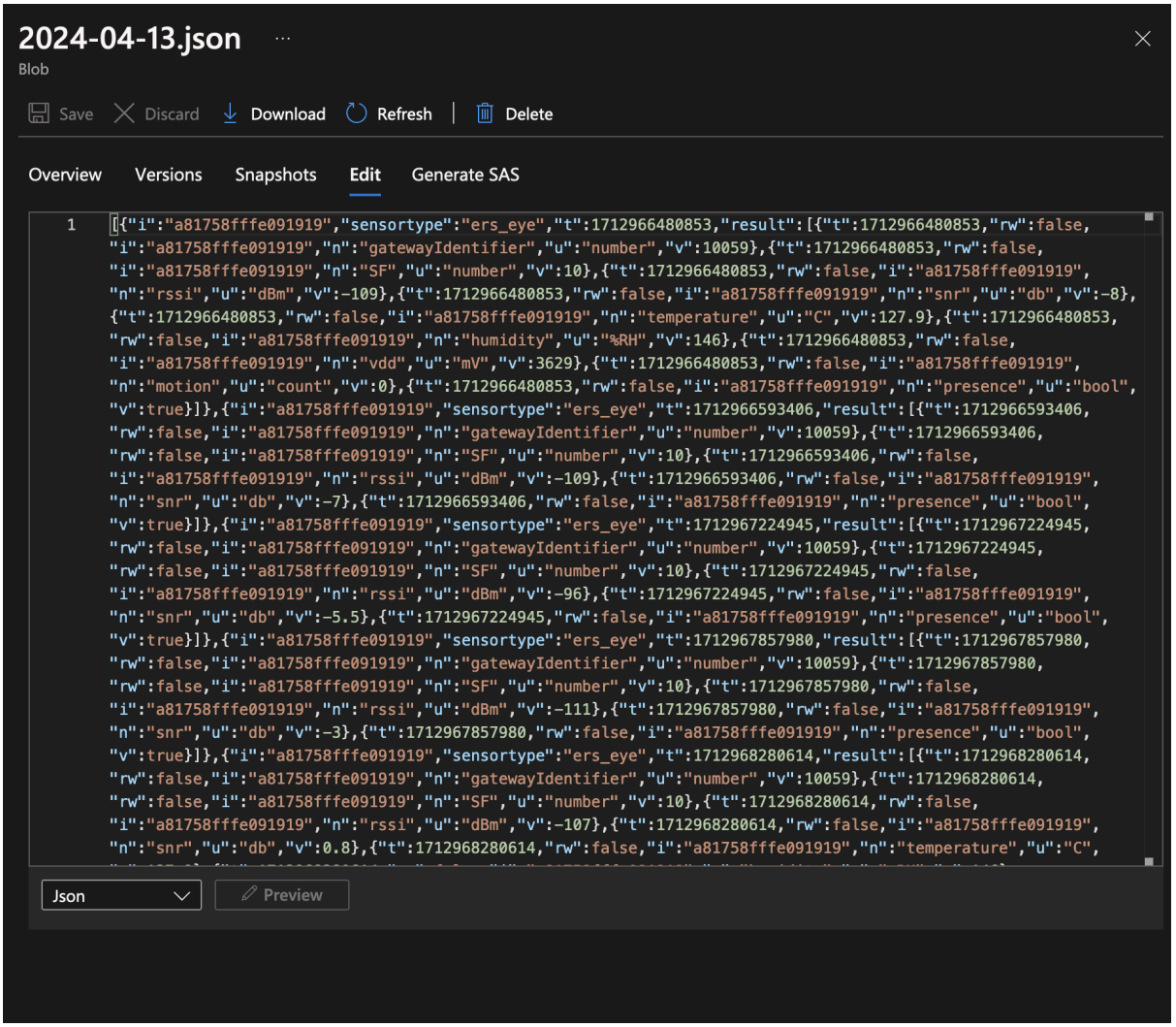
Dataen som er lagret i Azure er lagret i en Storage Account. I storage accounten kan du ha mapper som kalles “containers” hvor prosjektet har valgt en løsning ved å navngi dem med sensor-id til sensoren, vist i figur 5.2. Inne i hver mappe blir det lagret filer som i azure kalles “blobs” som er navngitt med dato som er vist i figur 5.3. Går man inn i en fil så er det en liste med målinger hvor hvert element i lista er ulike tidspunkt i løpet av den dagen, vist i figur 5.4.



Figur 5.2: Utklipp fra Containere i Storage Account i Azure



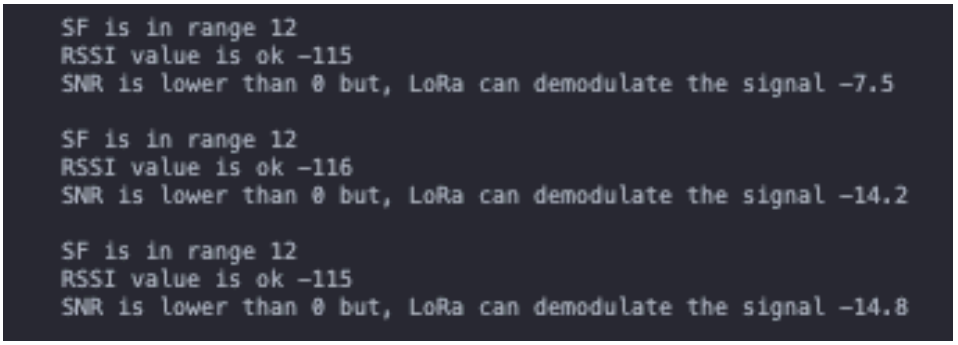
Figur 5.3: Utklipp av Blobs i en Container



Figur 5.4: Utklipp fra innholdet i en Blob i Azure

5.1.4 Script som måler signalstyrke

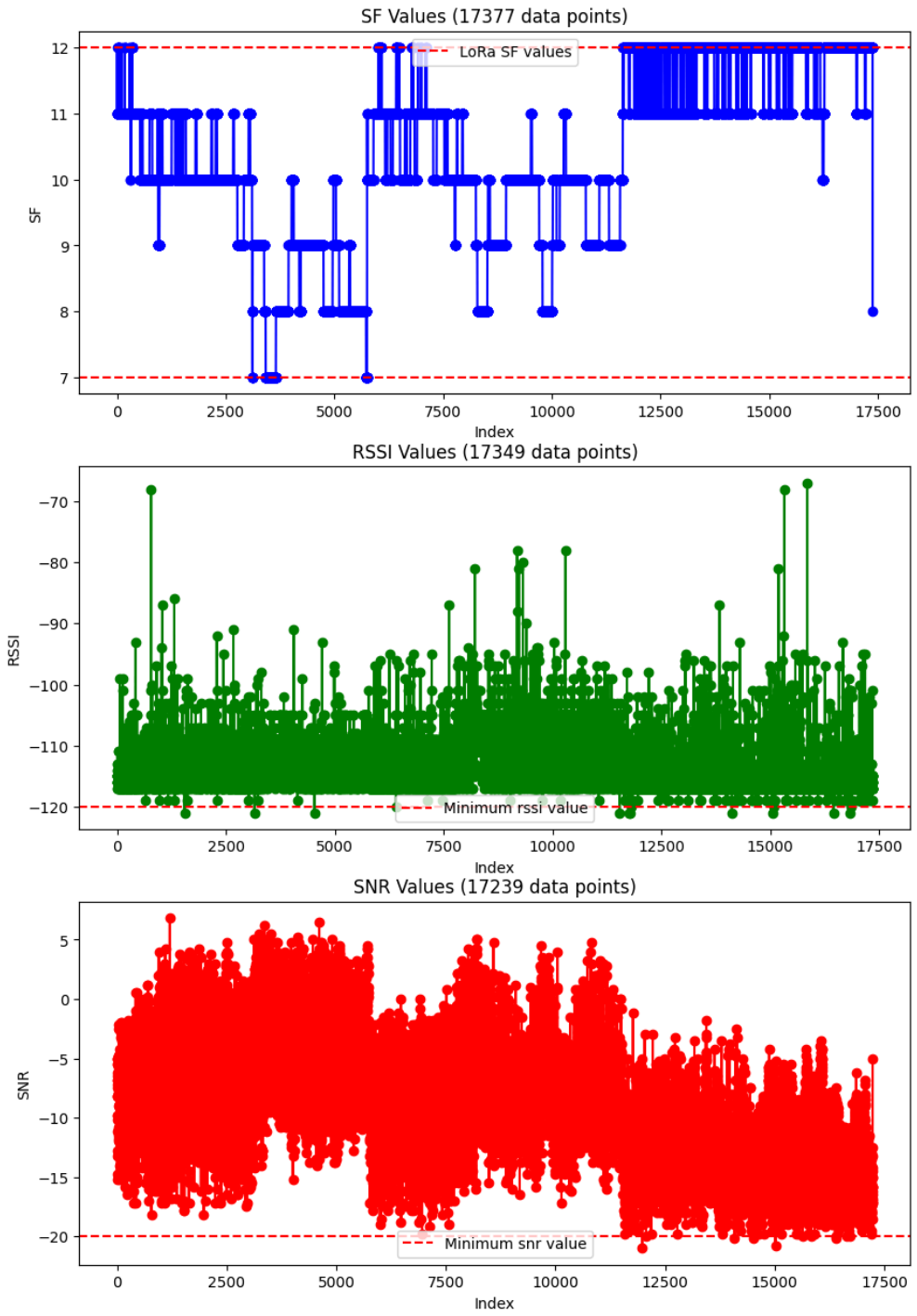
Scriptet main.py i signalstyrke mappen skriver resultatene pr. signal i terminalen. Betingelsene blir sjekket for hvert signal og klassen signalet tilhører, avhengig av resultat, blir skrevet ut i terminalen, vist i figur 5.5. Hvert signalresultat blir delt med et mellomrom i terminalen for å på en enkel måte kunne skille mellom signalene.



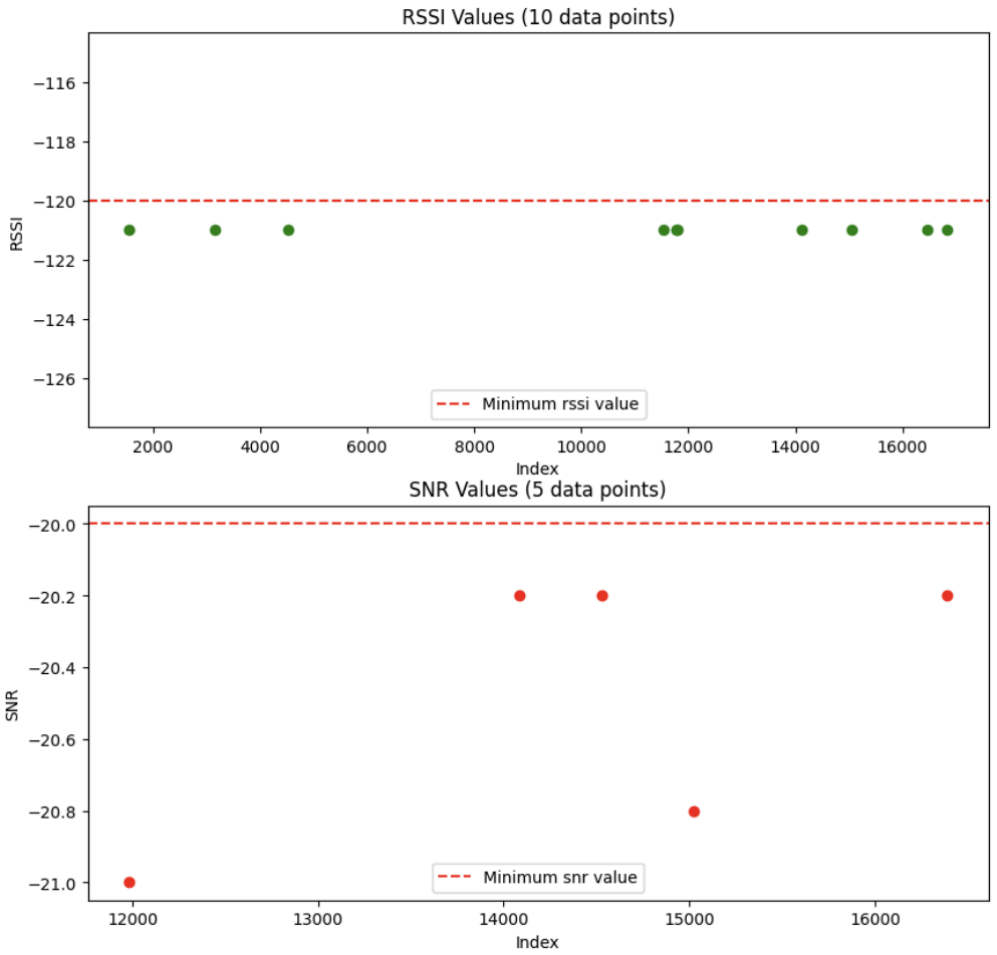
Figur 5.5: Utklipp fra terminalen som har sjekket SNR, SF og rssi verdiene til alle tilgjengelige signal.

Scriptet main.ipynb i signalstyrke mappen visualiserer, med hjelp av matplotlib-biblioteket, resultatene i tre forskjellige diagram. Første graf visualiserer hvilken SF-verdi, signalene benytter seg av med en blå graf. Graf nummer to er grønn og visualiserer rssi-verdiene til signalene. Her er grenseverdien satt til -120dBm og vises med en oransje stiplede linje. Til slutt visualiseres SNR-verdien med en oransje graf og igjen en oransje stiplede linje viser grenseverdien til denne målingen. Mer om grenseverdier og hvorfor de er satt til det de er i kapittel 4.3.1.5. Figurene 5.6 og 5.7 viser hvordan diagrammene og grafene ser ut. På tidspunktet Mandag 29.04 ble siste måling utført, da var det 17349 signal mottatt i Azure applikasjonen, derav 17349 punkter i grafen som måler rssi verdiene og 17239 signal i grafen som viser SNR verdiene. Grunnen til hvorfor antallet er ulikt er uvisst da gruppen ikke har annen tilgang til sensorene enn sensordata som gruppen henter inn fra oppdragsgiverens datastrøm.

Dette scriptet visualiserer også alle signalene som har verdier under grenseverdiene. Grunnlag for dette er at når det kommer mer og mer data inn i scriptet som visualiserer alle signal, kan det være vanskelig å se tydelig hvor mange signal som er under grensen.



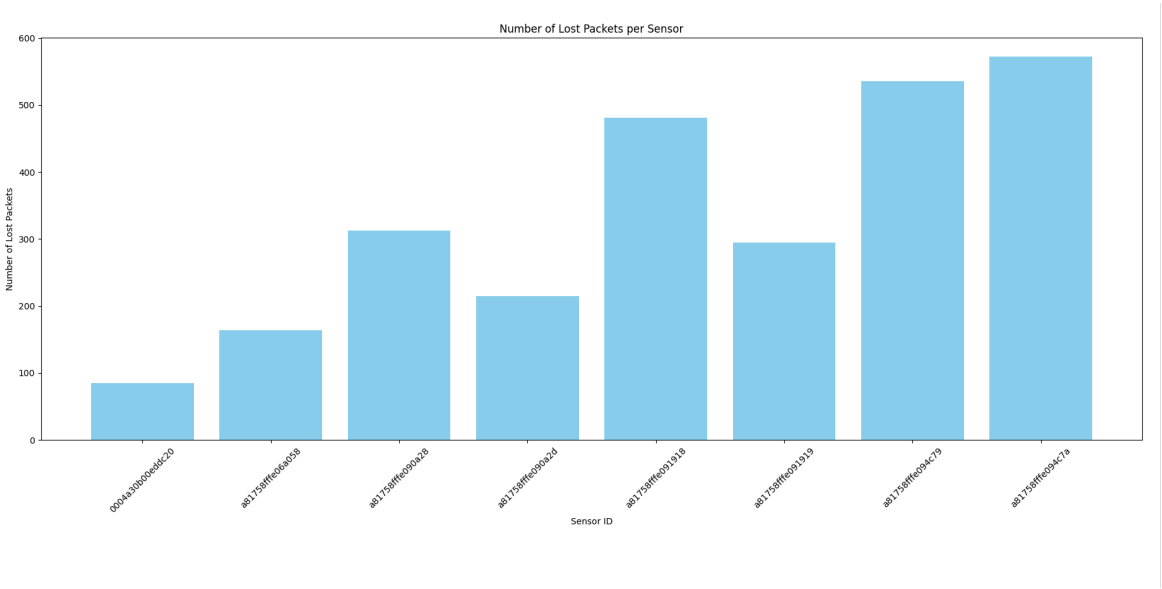
Figur 5.6: Diagram som visualiserer resultat fra sensordata (Mandag 29.04).



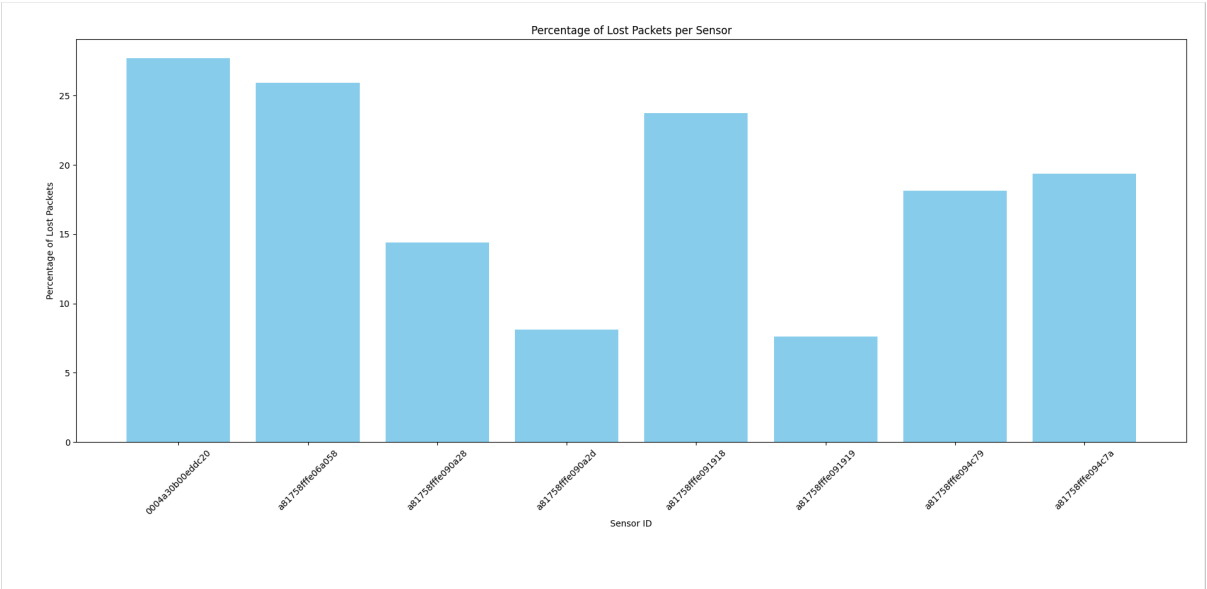
Figur 5.7: Diagram som visualiserer resultatene fra sensordata som er under grenseverdi (Mandag 29.04).

5.1.5 Script som måler tap av datapakker

Scriptet som måler tapte datapakker, måler 8 forskjellige sensorer. Scriptet produserer 2 grafer. En graf som viser antall tapte pakker, kan ses i figur 5.8, og en graf som viser hvor mange prosent av pakkene som går tapt, kan ses i figur 5.9.



Figur 5.8: Diagram som visualiserer resultatene for hvor mange datapakker som går tapt(Mandag 29.04).



Figur 5.9: Diagram som visualiserer resultatene for hvor mange prosent av datapakker som går tapt(Mandag 29.04).

5.2 Evalueringsresultat

Er signalstyrken god nok?

I dette eksperimentet ble det, avhengig av tiden en kjører scriptet, visualisert mange datapunkter hvor hvert datapunkt representerer et signal fra en sensor. Først og fremst kan en tydelig se at alle signalene har benyttet seg av en av de vanlige spreading faktorene LoRaWAN teknologien benytter seg av, SF7 til SF12. Dette var som forventet, men fortsatt interessant. Som en kan se på skjermbilde i figur 5.6 fra kapittel 5.1.4 så er største del av signalene sendt med en spredningsfaktor på 10 og oppover. SF10 og SF12 ser ut til å være mest brukte. Dette kan tyde på at endenoder eller sensorer er plassert langt fra gateways og trenger større spredning for å kunne treffe ønsket mottaker.

En kan også tydelig se at det er et fåtall signal som har rssi verdi lavere enn grenseverdien til LoRaWAN.
Siste måling i dette eksperimentet ble utført mandag 29.04 og her ble det målt 17349 signal hvor 10 signal ble målt under grenseverdien, dette tilsvarer ca. 0.058%.

Når det kommer til SNR-målingen, på tidspunktet bildet i figur 5.6 ble tatt i kapittel 5.1.4, opplever fem signal for mye støy til å demoduleres. Som nevnt tidligere kan LoRaWAN signal demoduleres eller gjenvinnes selv med mye støy. Signal-to-noise forhold kan være så lavt som -20 dB. På siste måling ble det målt 5 signal med for dårlig SNR verdi som tilsvarer ca. 0,03%.

Prosjektet er avgrenset til signalene hentet ut fra oppdragsgiverens datastrøm. Dette betyr at gruppen ikke helt sikkert kan si hva som gjør at fåtallet med dårlige verdier får disse verdiene. Avstand kan spille en rolle, står denne endenoden en plass som er svært langt fra nærmeste gateway? Står endenoden i et parkeringshus inne i et fjell langt fra nærmeste extender/repeater? Det kan også være samme endenode som sender dårlige signal flere ganger. Det er mange faktorer som kan gjøre at en endenode kan gi dårlig utslag uten at det kan sies konkret hva dette er i dette prosjektet. Gruppen sjekket flere ganger i løpet av prosjektet og kan konkludere dette eksperimentet med å se at selv om antall signaler øker betraktelig, øker ikke antall signal med rssi -og SNR verdier under grenseverdi. Dette underbygger teorien om at signalstyrken til LoRaWAN system er pålitelig.

Går datapakker tapt?

Ut fra resultatene ser det ut som at det går veldig mange pakker tapt. Men disse resultatene er veldig misvisende. Originalt var planen for å måle hvor mange datapakker som går tapt som følger. Eidsiva har en kontinuerlig MQTT strøm der det sendes sensordata. Denne sensordataen inneholder et timestamp som sier når den ble sendt. Gruppen har i tillegg informasjon om hvor ofte en sensor skal sende data.

Planen var å utvikle en applikasjon som "lyttet" på denne strømmen og lagret hver sensordata som ble sendt i en database. Dermed skulle det utvikles et python script som gikk gjennom alle de lagrede dataene og fant der hvor forskjellen mellom to tidsstempel var større enn det intervallet for hvor ofte en sensor skulle sende data. For eksempel for sensor med id "0004a30b00eddc20" vet vi at den skal sende data med en times intervall mellom hver sending. Da vil scriptet gå igjennom all data lagret for denne sensoren og finne antall ganger hvor forskjellen mellom to tidsstempel er større enn en time. Dette scriptet ble utviklet og fungerer som det skal, men det er problemer med dataen som analyseres.

Applikasjonen vi utviklet for å "lytte" på datastrømmen fungerer ikke helt som ønsket. Datastrømmen er designet slik at den sparker ut brukere dersom de har lyttet for lenge. Dette betyr at ved jevne mellomrom blir "lytteren" vår sparket ut av datastrømmen, det er satt inn en mekanisme for å koble på lytteren igjen , men dette tar ca. 1 minutt. Dette betyr at alle pakker som blir sendt dette minuttet hvor lytteren ikke er koblet på, ikke blir lagret i databasen. Dette fører til at python scriptet tror at disse pakkene er tapt under sending, selv om de blir sendt. En løsning på dette kunne vært å utnytte loggingen som var implementert i API-et. Dersom denne dataen hadde vært tilgjengelig kunne det blitt lagt inn i analysen, og pakker som manglet som skulle blitt sendt når lytteren var avkoblet kunne blitt ignorert.

Selv om resultatet ikke er helt tilregnelig, kan vi fortsatt få litt verdi ut av det. På de to sensorene som har mest pakketap, ser vi at signalstyrken er veldig dårlig. Dette tyder på at sensorene er dårlig plassert. Dersom du har rette tilganger kan du hente ut ekstra metadata fra sensorene. I denne metadataen finnes en frame-counter. Denne frame-counteren fungerer som en teller der den ved hver pakke som blir sendt øker telleren med en. Dersom gruppen hadde hatt tilgang til denne frame-counteren hadde den vært ideell å bruke for å se om det gikk tapt datapakker på veien. Siden arbeidsgiver ønsket at gruppen skulle utføre denne oppgaven uten noen eleverte tilganger kunne ikke gruppen bruke denne framecounteren og valgte dermed å

utvikle en egen løsning. Ut fra resultatene vi har fått er pakketapet altfor høyt, men ettersom løsningen vår ikke fungerer som vi hadde håpet, er ikke disse resultatene gode nok til å konkludere med.

Felteksperiment resultater

- **Nærhet til gateway**

  Når gruppen befant seg i nærheten eller med "klar bane" til en gateway, var måleverdiene som forventet normale. Alle verdier var bedre enn grenseverdiene for at signalet skal kunne tolkes og utnyttes av mottaker. Beste resultat gruppen målte var på taket på Bergen busstasjon. Her ble det målt en rssi verdi på -71 dBm. Ellers ble det målt gode rssi verdier når målingene skjedde uten "synlige" hindringer mellom tester og gateway. Verdiene varierte, men gruppen observerte et snitt av rssi verdiene på denne testen var på rundt -85 dBm til -100 dBm.

- **Under bakkenivå**

  Da gruppen befant seg under bakkeplan var det mange testtrykk som ikke ga noe signal. Om det ble observert signaler var målingene helt opp mot grenseverdi eller over. Verste måling var i klostergarasjen sitt heishus. Her ble det målt en rssi verdi på -132 dBm. Dette er en verdi som betyr at signalet er for dårlig for at mottaker skal kunne tolke det.

- **Mellom bygg (Ikke klar bane til gateway)**

  Da gruppen befant seg mellom bygg eller med synlig hindring mellom tester og gateway, var signalene også tolkbare for mottaker, altså under grenseverdi. Her testet gruppen flere plasser men observerte ikke noe sted hvor rssi var over -120 dBm. Plassene testene ble foretatt var eksempelvis 3 etg. på Galleriet, 6 etg. på Galleriet, Egon på bryggen og utenfor rådhuset for å nevne noen.

- **Utenfor rekkevidde (Gardemoen)**

  Et at gruppens medlemmer var på Gardermoen i løpet av våren. Vedkommende tok med seg testeren og testet om det var noe signal på denne flyplassen. Gardermoen er utenfor rekkevidden i følge et dekningskart til netmore. Dette ville gruppen bekrefte og det stemte. Ingen signal ble målt.

5.3 Prosjektgjennomføring

Prosjektet ble gjennomført på våren 2024 fra januar 2024 til mai 2024. Gruppen har hatt jevnlige møter for å planlegge fremgang og fordele oppgaver/fokusområder for videre arbeid. Da bachelorprosjektet er et 20 studiepoengs-fag ble det satt et timebudsjett på ca. 25 timer i uken per gruppemedlem. Dette tilsvarer 66% av en 100% arbeidsuke. Faktisk brukte timer gjennom våren er 545. Ukentlige timelister for prosjektet finnes i prosjekthåndbok (vedlegg 1). Gjenstående etter innlevering av rapport er refleksjonsnotat, sluttpresentasjon, lage EXPO-plakat og selve EXPO hvor prosjektet vises frem.

6 Diskusjon

6.1 Prosess

Gruppen opprettet sprinter på to uker, hvor det ble laget en plan om forventningene for denne sprinten. Gruppen hadde gjevne standups, men ikke alltid fysisk. Noen av dagene møtte gruppen fysisk for diskusjon og grundig gjennomgang, ofte i slutten av en sprint, andre standups foregikk digitalt for å sjekke status hos alle gruppemedlemmene. Denne metodikken fungerte bra i utviklingsdelen men gruppen benyttet den også i forskningsdelen. Her og bestemte gruppen seg for å fortsette å sette opp sprinter for hva vi ønsket å fokusere på og få til de neste to ukene. Her også med jevne standups for å diskutere fremgang enten fysisk eller digitalt, også hadde vi et større møte i slutten av sprintene for å diskutere resultatene for hver gruppe medlem. I tillegg til små standup møter og større sprint møter hadde også gruppen i snitt to fysiske møter i uken for å jobbe sammen. Denne måten å jobbe på fungerte bra og hele gruppen hadde alltid oversikt over fremgangen i prosjektet. Gruppen har fulgt GANTT-diagrammet så godt det lar seg gjøre, noen oppgaver har overlappet da enkelte ting har tatt litt lengre tid enn forventet, men alt i alt har prosjektet gått etter planen.

Scriptene som ble utviklet for å analysere tap av datapakker og for å måle signalstyrke startet utvikling midt i prosjektløpet. Begge scriptene ble fullført og begge scriptene fungerer for seg selv som planlagt. Det oppstod et problem med applikasjonen som ble lagd for å lytte på MQTT datastrømmen og lagre sensordata i databasen. Problemet var at “lytteren” med jevne mellomrom ble sparket ut av datastrømmen og dermed måtte bruke cirka ett minutt på å koble seg på igjen. Gruppens manglende kunnskap om MQTT er grunnen til at dette problemet oppstod/ikke ble oppdaget tidligere. Dette fører til misvisende data med tanke på tap av datapakker. Dette var et problem som dessverre ble oppdaget veldig sent i prosjektløpet og dermed førte til at resultatet som ble produsert av pakketap-analysen ikke er brukbart når en skal konkludere.

Etterhvert som oppgaven ble helt konkret og de første obligatoriske innleveringene var levert, økte mengden skriveøkter. Forskningsdelen av prosjektet var mest tidkrevende, så her ble det en del individuelt arbeid for å få fokusert på skrivingen. På denne tiden ble fremgang diskusjonene gjort digitalt, men gruppen fortsatte med faste fysiske møter en gang i uken. I

perioden med fokus på skriving ble disse fysiske møtene noe kortere for å prioritere tid til rapporten. Når rapporten begynte å ta form, økte hyppigheten med fysiske møter for å diskutere og samarbeide om hva som står i rapporten.

6.2 Forbedringer

Dersom dette prosjektet skulle gjennomføres på nytt over samme tidsperiode, er gruppen enig om at en konkret plan på hva vi faktisk skal gjøre kunne komme godt med. Selv om prosjektet hadde LoRaWAN som tema, ga oppdragsgiver gruppen litt frie tøyler til hva oppgaven skulle være. Dette førte til en vag startplan fra gruppen sin side som ble forandret etterhvert før den til slutt, sammen med oppdragsgiver, ble helt konkret bestemt til nåværende oppgave.

For å analysere tap av datapakker, med nåværende kunnskap om hvordan MQTT fungerer, måtte det blitt tatt en annerledes tilnærming fra start. Istedenfor å bruke tidsstempelet som ligger i en sensordata, burde heller gruppen fått ekstra tilganger fra oppdragsgiver og brukt frame-counteren som finnes i metadata hos en sensor dersom du har rette tilganger. Dette hadde gjort at scriptet kunne analysert data fra flere sensorer, ikke bare sensorer som sender data med bestemt tidsintervall. Dataen som pakketap scriptet skulle analysert måtte også blitt hentet inn på en mer optimal måte. Enten ved at det hadde blitt utviklet en bedre “lytter”, som ikke ble sparket ut med jevne mellomrom og dermed måtte bruke tid til å koble seg opp.

Sett tilbake på oppgavegjennomføringen, er gruppen sin tilgang til sensordata avgrenset til datastrømmen gruppen har fått tilgang til av oppdragsgiver. Her kunne gruppen kanskje som et tilskudd til eksperimenteringen, bruke en sensor og “koble” sensordata til en fysisk enhet gruppen har tilgang til. Dette kunne kanskje gitt gruppen en bredere forståelse på hardware som sender sensordata til datastrømmen istedenfor å kun bruke data fra sensorer gruppen ikke “ser”. Da datastrømmen gir gruppen nok data å jobbe og eksperimentere med, ville dette bare vært et interessant tillegg til prosjektet.

6.3 Bruk av kunstig intelligens

Kunstig intelligens eller KI er et hurtig voksende felt som blir mer og mer brukt og er med på å gjøre arbeidsoppgaver i forskjellige sektorer mer effektive. På samme måte kan og trolig vil KI implementeres innenfor verden av Internet Of Things (IoT).

Kan kunstig intelligens være med å gjøre LoRaWAN systemer enda mer pålitelig og effektiv? Om en har et system som måler for eksempel temperatur, kan en KI modell være nyttig. Dette er fordi en modell kan trenes på data fra sensorer. Denne treningen kan være med å hjelpe modellen å predikere en manglende datapakke. Dette vil være med på å holde overvåkingen av for eksempel temperaturen i et badebasseng mer sømløst og oversiktlig, uten "hull" i historikken til bassengtemperaturen.
Når det kommer til tap av datapakker i tidskritiske applikasjoner vil trolig KI ikke kunne hjelpe på samme måte. Om alle signal kommer frem til mottaker i en lengre periode som en KI modell er trent på vil modellen predikere følgende signal som ok selvom denne pakken går tapt. Dette kan være kritisk da den tapte pakken kan inneholde en alarm, for eksempel fra en fryser som står åpen. Alarmen vil dermed ikke komme frem til fagpersonell som skal ha denne alarmen og ting som medisiner, mat og andre elementer kan bli ødelagt. Dette kan føre til ødeleggelser som kan skade bedrifter økonomisk og tidsmessig.

Der gruppen ser at KI kan utnyttes effektivt er når det kommer til trening av data som skjer "før" en alarm går. Her kan en trene en modell til å gjenkjenne trenden på data før dataen treffer grenseverdien som utløser alarm. Dette vil være med å gi personell bedre tid på seg til å rette opp en potensiell kritisk feil før den nødvendigvis blir tidskritisk. Når modellen oppdager en trend hvor data pleier å nå grenseverdier for alarm, kan en foralarm gå slik at problemet kan sjekkes. Denne måten å håndtere tidskritiske systemer er også med på å lære hva som pleier å utløse diverse alarmer eller feilmeldinger, noe som kan effektivisere utbedring av systemer.

Da kunstig intelligens ikke er en del av oppgaven, går ikke denne rapporten grundig inn på temaet da dette medfører omfattende arbeid og ressurser. Gruppen ville derimot vise at KI, som er et høyaktuelt tema i dag, kan være med på å enda bedre være med å lage pålitelige systemer sammen med LoRaWAN.

7 Konklusjon og videre arbeid

7.1 Konklusjon av prosjektmål

Bachelorprosjektet hadde som mål å finne ut om LoRaWAN kan brukes innenfor tidskritiske applikasjoner. Prosjektet tok sikte på å finne ut hvor mye datapakker som gikk tapt og hvor bra signalene og dekningen må være. I den anledning er det ønskelig å besvare følgende problemstilling:

Er LoRaWAN egnet til bruk innenfor tidskritiske applikasjoner?

I forbindelse med prosjektet ble det også satt opp noen delmål og forskningsspørsmål for å få en lettere tilnærming til å svare på problemstillingen. Forskningsspørsmålene som ble satt opp var:

1. Hva er fordelene og ulempene ved bruk av LoRaWAN innenfor tidskritiske applikasjoner?
2. Er LoRaWAN-protokollen pålitelig?

Etter hva dette prosjektet har funnet ut, så ble målene nådd for signalstyrke. Gruppen kom ikke helt i mål med å nøyaktig finne ut hvor ofte datapakker går tapt. Ifølge målingene på signalstyrken er LoRaWAN-protokollen pålitelig nok til å kunne brukes i tidskritiske applikasjoner med tanke på hva som ble funnet ut i dette eksperimentet. Eksperimenteringen med pakketap ble ikke helt som planlagt grunnet at lytteren blir kastet ut fra MQTT, snakket om i punkt 5.2. Dette bør derfor undersøkes nærmere. Gruppen oppdaget dette litt for seint for å kunne utbedre. Utover avgrensningene prosjektet har satt finnes det selvfølgelig spesifikke scenarier hvor protokollen må undersøkes enda mer.

IoT-løsninger og tidskritiske applikasjoner er stadig i utvikling som krever at protokollene må holde følge. Dersom dette ikke er tilfelle, kan det hende at spesifikke protokoller bare egner seg til visse løsninger. På bakgrunn av eksperimentering og forskning som gruppen har foretatt seg, kan en se at LoRaWAN er aktuelt å bruke på steder der det er vanskelig å komme til og over lange avstander, *også* hvor det er tidskritiske elementer til stede.

7.2 Hvem er resultatet nyttig for

Resultatet av oppgaven er nyttig for oppdragsgiver og andre aktører som ønsker å utvide bruken av LoRaWAN-teknologien i tidskritiske sektorer i Norge. Oppgaven kan gi oppdragsgiver og andre virksomheter bredere innsikt i påliteligheten til teknologien og gir et svar på om det er verdt å jobbe videre med.

7.3 Videre arbeid

Videre arbeid for prosjektet kan være å gå dypere inn i LoRaWAN og eksperimentere mer på teknologien. Gruppen måtte på grunn av tiden gitt til prosjektet, velge ut et par eksperimenter gruppen skulle fokusere på for å kunne komme med en konklusjon ved prosjektslutt.

Eksempler på andre eksperimenter kan være å teste nøyaktigheten på GPS-måling ved bruk av LoRaWAN GPS-måler eller nødknapp. Her kan en ha med en nøyaktig GPS og sammenligne koordinater hvor nødknappen er og hvor LoRaWAN systemet sier at den er. Dette kunne vært interessant for å enda sikrere evaluere påliteligheten og nøyaktigheten til teknologien.

Gruppen har i dette prosjektet sett overordnet på teknologien for å kunne konkludere om den er pålitelig nok til å kunne benyttes i tidskritiske applikasjoner. Gruppen har derimot ikke gått noe dypere inn på infrastrukturen som må til for å ha et effektivt og pålitelig system som skal benyttes i tidskritiske virksomheter. Videre arbeid kan være å utforske og ikke minst teste hva som fungerer og hva som fungerer mindre bra med infrastruktur til systemer. Her vil det medføre jobb ved å se nærmere på utstyr og hjelpemiddel som til sammen utgjør et best mulig system med en så god ytelse som overhodet mulig.

Faktisk testing er også en del av videre arbeid. Her kan en teste noe tidskritisk, i et velfungerende system, som for eksempel en fryser. Da kan en teste påliteligheten, om systemet og teknologien kan brukes og ikke minst stoles på i praksis.

8 Referanser

- Altibox (n.d.) *IoT levert via LoRaWAN: Long range, low power*. Tilgjengelig fra: https://www.altibox.no/bedrift/iot/lorawan/(Hentet: 05.02.2024).
- Altunian, G. (2021) *What Is Signal-to-Noise Ratio and Why Does It Matter?*. Hentet fra: https://www.lifewire.com/signal-to-noise-ratio-3134701 (Hentet: 15.03.2024).
- Barney,N. (2023, februar). Go programming language. Tilgjengelig fra: https://www.techtarget.com/searchitoperations/definition/Go-programming-language (hentet: 04.03.2024)
- Catherwood, P. A., Steele, D., Little, M., Mccomb, S. og Mclaughlin, J. (2018) *A Community-Based IoT Personalized Wireless Healthcare Solution Trial* Tilgjengelig fra: https://ieeexplore.ieee.org/document/8355907 (Hentet 17.02.2024)
- Codefresh, (n,d) What is Azure Devops? Tilgjengelig fra: https://codefresh.io/learn/azure-devops/ (hentet: 04.03.2024)
- Coursera (2023, 3. april) What is Python used for? A beginners guide. Tilgjengelig fra: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python (Hentet 03.03.2024)
- Dusuniot (2022) *LoraWAN-gatewayer: Hvor langt kan de overføre?* Tilgjengelig fra: https://www.dusuniot.com/no/blog/lorawan-gateways-how-far-can-they-transmit/ (Hentet: 24.04.2024).
- Eidsiva-a (n.d) *Om Eidsiva.* Tilgjengelig fra: https://www.eidsiva.no/om-eidsiva/ (Hentet: 05.02.2024).
- Eidsiva-b-c (n.d) *Fiber og digitale løsninger.* Tilgjengelig fra: https://www.eidsiva.no/om-eidsiva/bredband/ (Hentet: 05.02.2024).
- Elsys (2019) *ESR CO2.* Hentet fra: https://elsys.se/public/datasheets/ERS_CO2_datasheet.pdf (Hentet: 18.03.2024).
- Elsys (2020) *EMS Door.* Hentet fra: https://elsys.se/public/datasheets/EMS_Door_datasheet.pdf (Hentet: 18.03.2024).
- Fremont, C (2023) *LoRaWAN Leads the Global Industrial Evolution and Drives Industry 5.0's Sustainability, Efficiency and Quality of Life Priotities.* Tilgjengelig fra: https://lora-alliance.org/lora-alliance-press-release/lorawan-leads-the-global-industrial-evolution-and-drives-industry-5-0s-sustainability-efficiency-and-quality-of-life-priorities/ (Hentet 09.04.2024) Søkeord google: LoRaWAN time critical industrial

- Guido von Rossum (1998) *Glue it all together with Python* Tilgjenglig fra: https://www.python.org/doc/essays/omg-darpa-mcc-position/ (Hentet 16.02.2024)
- Gerard, S. (2022) *7 Best Practices for Leveraging LoRa in IoT Smart Healthcare.* Tilgjengelig fra: https://www.mokolora.com/lora-iot-smart-healthcare/ (Hentet: 10.04.2024).
- Luna, J (2023, mars) Top programming languages for Data Scientists in 2023. Tilgjengelig fra https://www.datacamp.com/blog/top-programming-languages-for-data-scientists-in-2022 (Hentet 03.03.2024)
- Hiotech (n.d.) *LoRaWAN GPS Smartwatch.* Tilgjengelig fra: https://www.hiotech.net/product/lora_gps_watch_for_smart_senior_care-en.html (Hentet: 09.04.2024)
- LoRa (2018) *LoRa* Tilgjengelig fra: https://lora.readthedocs.io/en/latest/#rssi (Hentet: 15.03.2024).
- Macnam (2023, 2.desember) What are the advantages and disadvantages of using the LoRaWAN network? Tilgjengelig fra: https://medium.com/@digital.macnman/what-are-the-advantages-and-disadvantages-of-using-the-lorawan-network-c40193901087 (Hentet 08.04.2024)
- Mea, V. D., Popescu, M.H., Gonano, D., Petaros, T., Emili, I. og Fattori, M. G. (2020) *A Communication Infrastructure for the Health and Social Care Internet of Things: Proof-of-Concept Study* Tilgjengelig fra: https://medinform.jmir.org/2020/2/e14583 (Hentet 17.02.2024)
- Meaney(n.d) *LoraWan*. Tilgjengelig fra: https://my.avnet.com/wcm/connect/0a02b164-a434-4346-a801-c35885961b4f/lorawan.pdf?MOD=AJPERES&CVID=n4XMN3N&attachment=false&id=1585845692618 (Hentet 2024, 8. april)
- MINESEMI (2022) *Leveraging LoRa in IoT Smart Healthcare* Tilgjengelig fra: https://en.minewsemi.com/blog/leveraging-loRa-in-iot-smart-healthcare (Hentet:07.05.24).
- MOKO Smart (2023) *LoRaWAN®-Based Panic Button. Tilgjengelig fra:* https://www.mokosmart.com/lorawan-button-lw004-pb/ (Hentet: 07.05.24).
- MQTT (n.d) *MQTT* Tilgjengelig fra: https://mqtt.org/ (Hentet 06.05.2024)

- Netmore (2024) *netmoreLoRaWAN.* Tilgjengelig fra: https://portal.blink.services/net/netmore-coverage?network=NORWAY&type=NET_E (Hentet: 05.04.2024)
- NTNU (n.d) rest. Tilgjengelig fra: https://folk.ntnu.no/olso/wu/rest/rest.html (Hentet 30.4.2024)
- PAFEX Prakash Air Freight India Pvt Ltd (2023) *Time Critical Logistics* Tilgjengelig fra: https://www.linkedin.com/pulse/time-critical-logistics-pafex-1f/ (Hentet 10.04.2024)
- Pous, Marc *Run your LoRaWAN gateway with your own The Things Stack network server on a Raspberry Pi and balena* Tilgjengelig fra: https://dev.to/mpous/run-your-lorawan-gateway-with-your-own-the-things-stack-network-server-on-a-raspberry-pi-and-balena-4o4n (Hentet 06.06.2024)
- Praecipio (2022) *What Is A Sprint In Scrum?* Tilgjengelig fra: https://www.praecipio.com/resources/articles/what-is-a-sprint-in-scrum (Hentet: 16.02.2024).
- Ramsøy, C (2022) *En kort introduksjon til Scrum.* Tilgjengelig fra: https://www.visma.no/blogg/en-kort-introduksjon-til-scrum/ (Hentet 30.04.2024)
- Rush, C. (2021) *Hva er LoRa- og LoRaWAN-teknologi?* Tilgjengelig fra: https://knowhow.distrelec.com/no/telekommunikasjon/hva-er-lora-og-lorawan-teknologi/ (Hentet 18.03.2024).
- Santo, D. E. (2022) *Top 5 main Agile methodologies: advantages and disadvantages.* Tilgjengelig fra: https://www.xpand-it.com/blog/top-5-agile-methodologies/ (Hentet: 16.01.2024).
- Store Norske Liksikon (SNL) *API* Tilgjengelig fra: https://snl.no/API (Hentet 06.05.2024)
- Sokogskriv (2024) *Søketeknikker.* Tilgjengelig fra: https://www.sokogskriv.no/soking/soketeknikker.html#kombinasjon-med-not (Hentet: 04.03.24).
- Stanke, B. (n.d) *Kanban vs Scrum: Which Agile Workflow Framework is Best.* Tilgjengelig fra: https://www.bobstanke.com/blog/kanban-vs-scrum-which-agile-workflow-framework-is-best (Hentet: 16.02.2024)

- TECTELIC Communications inc (2023) *LoRaWAN - Most Common Applications and Use Cases.* Hentet fra: https://www.iotforall.com/lorawan-most-common-applications-and-use-cases (Hentet: 18.03.2024).
- The Things Industries (2022) Why you should use LoRa technology and LoRaWAN for your next IoT use case. Tilgjengelig fra: https://www.thethingsindustries.com/news/why-you-should-use-lora-technology-and-lorawan-for-your-next-iot-use-case/ (hentet: 15.04.2024)
- The Things Industries-a (2023) *Accelerating Industrial IoT with LoRaWAN* Tilgjengelig fra: https://www.thethingsindustries.com/news/accelerating-industrial-iot-with-lorawan/ (Hentet 06.05.2024) Søkeord google: LoRaWAN logistics and time critical
- The Things Industries-b (2023) *Global Logistics solutions with IoT and LoRaWAN* Tilgjengelig fra: https://www.thethingsindustries.com/news/global-logistics-solutions-with-iot-and-lorawan/ (Hentet 10.04.2024) Søkeord google: LoRaWAN logistics tand ime critical
- Thethingsnetwork (n.d) *RSSI and SNR.* Hentet fra: https://www.thethingsnetwork.org/docs/lorawan/rssi-and-snr/ (Hentet: 15.03.2024).
- Thethingsnetwork (n.d). *Spreading Factors.* Hentet fra: https://www.thethingsnetwork.org/docs/lorawan/spreading-factors/ (Hentet: 15.03.2024).
- Tvedt, M. N. (2021) *Presisjon eller fullstendighet? Om ulike typer litteratursøk.* Tilgjengelig fra: https://blogg.hvl.no/bibl/presisjon-eller-fullstendighet-om-ulike-typer-litteratursok/ (Hentet: 04.03.2024).
- TWTG (2023) *Why LoRaWAN is The Idustrial IoT Enabler You Can´t Ignore (⅘)* Tilgjengelig fra: https://www.linkedin.com/pulse/why-lorawan-industrial-iot-enabler-you-cant-ignore-45-twtg/ (Hentet 10.04.2024) Søkeord google: LoRaWAN logistics and time critical
- UIO (2017) *Hva er micro services?* Tilgjengelig fra: https://www.uio.no/tjenester/it/utvikling/integrasjonsarkitektur/hjelp/hva-er-micro-services.html (Hentet 30.4.2024)
- Unipi Technology *Integration of an IoT indoor air quality sensor with LoRa communication* Telgjengelig fra:

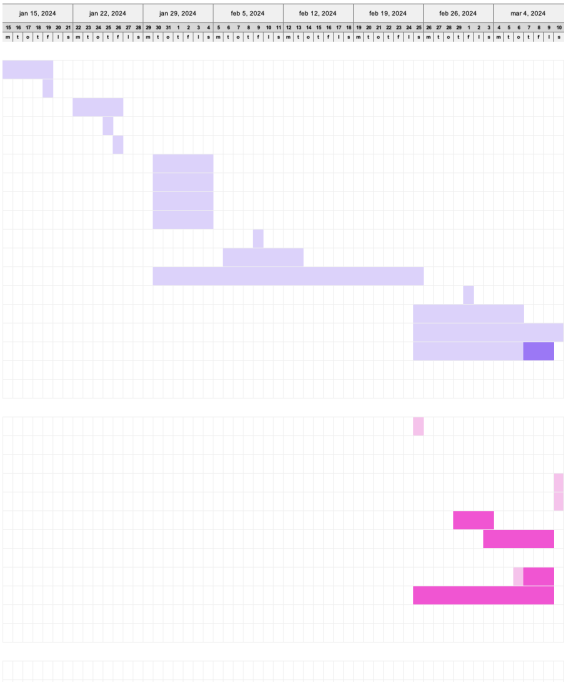
https://www.unipi.technology/case_study/integration-of-an-iot-indoor-air-quality-sensor-with-lora-communication-364 (Hentet 06.05.2024)

- Welch, B. (2022) *Is LoRaWAN a viable option for critical messaging?* Tilgjengelig fra: https://issuu.com/westwick-farrowmedia/docs/critical_comms_jul_aug_2022/s/16336896 (Hentet: 06.04.2024).
- Wikipedia (2022) *Bitrate* Tilgjengelig fra: https://no.wikipedia.org/wiki/Bitrate (Hentet 30.04.2024)
- Wikipedia (2024) *Tingenes Internett* Tilgjengelig fra: https://no.wikipedia.org/wiki/Tingenes_internett (Hentet 29.04.2024)
- Wikipedia (2024) *Programmeringsgrensesnitt* Tilgjengelig fra: https://no.wikipedia.org/wiki/Programmeringsgrensesnitt (Hentet 30.04.2024)
- Wikipedia(2024) *JSON* Tilgjengelig fra: https://no.wikipedia.org/wiki/JSON (Hentet 30.04.2024)

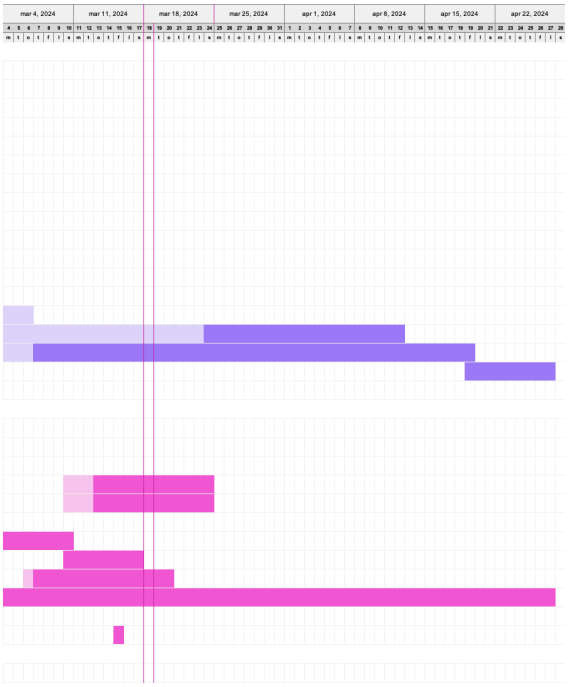
9 Vedlegg

- GANTT-diagram:

TASK	Time used	START	END
Dokumentasjon			
Intern planlegging av bachelor prosjekt	100 %	1.15.24	1.19.24
Oppstartsmøte med oppdragsgiver (Oblig 1)	100 %	1.19.24	1.19.24
Tolke oppgave og skrive skjema for tolkning til arbeidsgiver	100 %	1.22.24	1.26.24
Statusmøte 1 med Hvl veileder (Oblig 2)	100 %	1.25.24	1.25.24
Godkjenning av tolkning av oppgave fra oppdragsgiver	100 %	1.26.24	1.26.24
Registrere prosjekttittel (En del av Oblig 3)	100 %	1.30.24	2.4.24
Risikoanalyse (En del av Oblig 3)	100 %	1.30.24	2.4.24
Skrive innledning i visjonsdokument (En del av Oblig 3)	100 %	1.30.24	2.4.24
Skrive sammendrag problem og produkt (Visjonsdokument) (En del av Oblig 3)	100 %	1.30.24	2.4.24
Statusmøte 2 med Hvl veileder	100 %	2.9.24	2.9.24
Analyse av tidligere oppgaver (Oblig 5)	100 %	2.6.24	2.13.24
Utkast midtveisrapport pluss dokumentasjon, iterasjon 2 (Oblig 6)	100 %	1.30.24	2.25.24
Statusmøte 3 med Hvl veileder (Oblig 7)	100 %	3.1.24	3.1.24
Midveisrapport ferdig (Oblig 8)	100 %	2.25.24	3.6.24
Diverse dokumenter, iterasjon 3 (Oblig 10)	60 %	2.25.24	4.12.24
Utkast 1 rapport (Oblig 11)	20 %	2.25.24	4.19.24
Utkast 2 rapport (Oblig 12)	0 %	4.19.24	5.3.24
Refleksjonsnotat (oblig 13)	0 %	5.17.24	5.24.24
Utvikling			
Planlegge struktur på API	100 %	2.25.24	2.25.24
Initiere API og få tilgang til oppdragsgivers datapool	100 %	2.25.24	3.10.24
Initiere Azure container for å lagre data fra datapool	90 %	2.25.24	3.10.24
Opprette API - kall for å kunne hente ut data fra datapool	25 %	3.10.24	3.24.24
Skrive test kode for API - kall	25 %	3.10.24	3.24.24
Lage script for å måle signalstyrke	0 %	2.29.24	3.3.24
lage skript for å måle/undersøke tap av datapakker	0 %	3.3.24	3.10.24
Utvikle et script som plotter all data presentert i grafer	0 %	3.10.24	3.17.24
Litteratursøk	7 %	3.6.24	3.20.24
Evaluere	0 %	2.25.24	5.16.24
Presentasjon			
Midtveispresentasjon presentasjon (Oblig 9)	0 %	3.15.24	3.15.24
Sluttpresentasjon	0 %	6.3.24	6.3.24
EXPO	0 %	6.12.24	6.12.24



TASK	Time used	START	END
Dokumentasjon			
Intern planlegging av bachelor prosjekt	100 %	1.15.24	1.19.24
Oppstartsmøte med oppdragsgiver (Oblig 1)	100 %	1.19.24	1.19.24
Tolke oppgave og skrive skjema for tolkning til arbeidsgiver	100 %	1.22.24	1.26.24
Statusmøte 1 med Hvl veileder (Oblig 2)	100 %	1.25.24	1.25.24
Godkjenning av tolkning av oppgave fra oppdragsgiver	100 %	1.26.24	1.26.24
Registrere prosjekttittel (En del av Oblig 3)	100 %	1.30.24	2.4.24
Risikoanalyse (En del av Oblig 3)	100 %	1.30.24	2.4.24
Skrive innledning i visjonsdokument (En del av Oblig 3)	100 %	1.30.24	2.4.24
Skrive sammendrag problem og produkt (Visjonsdokument) (En del av Oblig 3)	100 %	1.30.24	2.4.24
Statusmøte 2 med Hvl veileder	100 %	2.9.24	2.9.24
Analyse av tidligere oppgaver (Oblig 5)	100 %	2.6.24	2.13.24
Utkast midtveisrapport pluss dokumentasjon, iterasjon 2 (Oblig 6)	100 %	1.30.24	2.25.24
Statusmøte 3 med Hvl veileder (Oblig 7)	100 %	3.1.24	3.1.24
Midveisrapport ferdig (Oblig 8)	100 %	2.25.24	3.6.24
Diverse dokumenter, iterasjon 3 (Oblig 10)	60 %	2.25.24	4.12.24
Utkast 1 rapport (Oblig 11)	20 %	2.25.24	4.19.24
Utkast 2 rapport (Oblig 12)	0 %	4.19.24	5.3.24
Refleksjonsnotat (oblig 13)	0 %	5.17.24	5.24.24
Utvikling			
Planlegge struktur på API	100 %	2.25.24	2.25.24
Initiere API og få tilgang til oppdragsgivers datapool	100 %	2.25.24	3.10.24
Initiere Azure container for å lagre data fra datapool	90 %	2.25.24	3.10.24
Opprette API - kall for å kunne hente ut data fra datapool	25 %	3.10.24	3.24.24
Skrive test kode for API - kall	25 %	3.10.24	3.24.24
Lage script for å måle signalstyrke	0 %	2.29.24	3.3.24
lage skript for å måle/undersøke tap av datapakker	0 %	3.3.24	3.10.24
Utvikle et script som plotter all data presentert i grafer	0 %	3.10.24	3.17.24
Litteratursøk	7 %	3.6.24	3.20.24
Evaluere	0 %	2.25.24	5.16.24
Presentasjon			
Midtveispresentasjon presentasjon (Oblig 9)	0 %	3.15.24	3.15.24
Sluttpresentasjon	0 %	6.3.24	6.3.24
EXPO	0 %	6.12.24	6.12.24



TASK	Time used	START	END
Dokumentasjon			
Intern planlegging av bachelor prosjekt	100 %	1.15.24	1.19.24
Oppstartsmøte med oppdragsgiver (Oblig 1)	100 %	1.19.24	1.19.24
Tolke oppgave og skrive skjema for tolkning til arbeidsgiver	100 %	1.22.24	1.26.24
Statusmøte 1 med Hvl veileder (Oblig 2)	100 %	1.25.24	1.25.24
Godkjenning av tolkning av oppgave fra oppdragsgiver	100 %	1.26.24	1.26.24
Registrere prosjekttittel (En del av Oblig 3)	100 %	1.30.24	2.4.24
Risikoanalyse (En del av Oblig 3)	100 %	1.30.24	2.4.24
Skrive innledning i visjonsdokument (En del av Oblig 3)	100 %	1.30.24	2.4.24
Skrive sammendrag problem og produkt (Visjonsdokument) (En del av Oblig 3)	100 %	1.30.24	2.4.24
Statusmøte 2 med Hvl veileder	100 %	2.9.24	2.9.24
Analyse av tidligere oppgaver (Oblig 5)	100 %	2.6.24	2.13.24
Utkast midtveisrapport pluss dokumentasjon, iterasjon 2 (Oblig 6)	100 %	1.30.24	2.25.24
Statusmøte 3 med Hvl veileder (Oblig 7)	100 %	3.1.24	3.1.24
Midveisrapport ferdig (Oblig 8)	100 %	2.25.24	3.6.24
Diverse dokumenter, iterasjon 3 (Oblig 10)	60 %	2.25.24	4.12.24
Utkast 1 rapport (Oblig 11)	20 %	2.25.24	4.19.24
Utkast 2 rapport (Oblig 12)	0 %	4.19.24	5.3.24
Refleksjonsnotat (oblig 13)	0 %	5.17.24	5.24.24
Utvikling			
Planlegge struktur på API	100 %	2.25.24	2.25.24
Initiere API og få tilgang til oppdragsgivers datapool	100 %	2.25.24	3.10.24
Initiere Azure container for å lagre data fra datapool	90 %	2.25.24	3.10.24
Opprette API - kall for å kunne hente ut data fra datapool	25 %	3.10.24	3.24.24
Skrive test kode for API - kall	25 %	3.10.24	3.24.24
Lage script for å måle signalstyrke	0 %	2.29.24	3.3.24
lage skript for å måle/undersøke tap av datapakker	0 %	3.3.24	3.10.24
Utvikle et script som plotter all data presentert i grafer	0 %	3.10.24	3.17.24
Litteratursøk	7 %	3.6.24	3.20.24
Evaluere	0 %	2.25.24	5.16.24
Presentasjon			
Midtveispresentasjon presentasjon (Oblig 9)	0 %	3.15.24	3.15.24
Sluttpresentasjon	0 %	6.3.24	6.3.24
EXPO	0 %	6.12.24	6.12.24

- Risikovurderings diagram:

	Hendelse /Risiko	Årsak	Sannsynlighet	Konsekvens	Risiko-produkt	Tiltak
1	*Applikasjonen blir ikke tatt i bruk.*	*Kunde kan eventuelt lage sin egen applikasjon på oppdragsgivers infrastruktur, og bare bruke deres datapool*	*Høy (4)*	*Svært Lav (1)*	4	*Gjennomføre brukertesting* *Lage en bra nok applikasjon så kunde heller vil bruke den.*
2	*Gruppen klarer ikke levere et produkt som tilfredsstiller kravet for å svare på problemstilling*	*Gruppen har ikke nok teknisk kompetanse* *Gruppen har ikke gjort nok forarbeid* *Gruppen blir overambisiøs og prøver å lage mer enn de klarer å gjennomføre*	*Lav (2)*	*Svært høy (5)*	10	*Jobbe jevnt og lære seg teknologien som blir brukt* *Gjør nok forarbeid og god planlegging* *Sette realistiske mål*
3	*Frafall i gruppen og fremgang går ned (Mere arbeid på friske gruppemedlemmer)*	*Sykdom eller andre helse årsaker til å ikke kunne jobbe med oppgaven* *Personlige årsaker* *Familiære årsaker*	*Svært lav (1)*	*Høy (4)*	4	*Holde seg frisk* *Unngå situasjoner som kan skade deg eller gjøre deg syk.* *Prøve å gjøre arbeidsoppgaver hjemmefra.*
4	*Interne konflikter*	*Dårlig samarbeid i gruppen, arbeidsmengden blir ikke fordelt likt.*	*Svært lav (1)*	*Svært høy (5)*	5	*God kommunikasjon innad i grupppen.* *Dele arbeidsmengden på en måte alle blir fornøyde*
5	*Oppdragsgiver går konkurs*	*Oppdragsgiver slutter å tjene penger og må melde konkurs*	*Svært lav (1)*	*Svært høy (5)*	5	*Utenfor gruppens kontroll*
6	*Oppdragsgiver blir oppkjøpt*	*Holder på å gå konkurs* *Vil slå seg sammen med en annen bedrift*	*Svært lav (1)*	*Lav (2)*	2	*Utenfor gruppens kontroll*
7	*Får ikke nok testdata.*	*Det er vanskelig å få tak i relevant testdata på grunn av mangel på tilgang.* *Nettet er nede*	*Lav (2)*	*Høy(4)*	8	*Finn ut hva testdata vi trenger tidlig sånn at man kan begynne å planlegge tidlig.*
8	*Tekniske problemer*	*Menneskelig svikt* *programvarefeil*	*Middels (3)*	*Middels(3)*	9	

- Risikoprodukt diagram

Sannsynlighet						
	Svært Høy	5	10	15	20	25
	Høy (4)	4	8	12	16	20
	Middels (3)	3	6	9	12	15
	Lav (2)	2	4	6	8	10
	Svært Lav	1	2	3	4	5
		Svært Lav (1)	Lav (2)	Middels (3)	Høy (4)	Svært Høy (5)
	Konsekvens					

- Vedlegg 1. Prosjekthåndbok
- Vedlegg 2. Visjonsdokument
- Vedlegg 3. Systemdokumentasjon
- Vedlegg 4. Kildekode API
- Vedlegg 5. Kildekode Scripts