NTNU

Kunnskap for en bedre verden

INSTITUTT FOR ELEKTRONISKE SYSTEMER

BACHELOROPPGAVE

ELEKTRONIKK OG SENSORSYSTEMER

Parameterovervåkning i tunnelanlegg ved bruk av LoRaWAN-basert IoT-system

---

av:

GRUPPE E2317

Czarnocka, Julia

Lapiz, Lourenzo Ray Apellido

Lie, Edvard

Nielsen, Erik Klanwilai

Trondheim, Mai 2023

Bacheloroppgave BIELEKTRO

Oppgavetittel:

Parameterovervåkning i tunnelanlegg ved bruk av LoRaWAN-basert IoT-system

Parameter monitoring in underground construction sites using a LoRaWAN-based IoT system

Forfattere:
Julia Czarnocka
Lourenzo Ray Apellido Lapiz
Edvard Lie
Erik Klanwilai Nielsen

Prosjektnummer:	E2317
Innleveringsdato:	22.05.2021
Gradering:	[x] åpen [ ] lukket

Studium:	Elektroingeniør - BIELEKTRO
Studieretning:	Elektronikk og sensorsystemer
Veileder internt:	Herman Ranes
Institutt:	Institutt for elektroniske systemer
Oppdragsgiver:	Skanska Norge AS
Kontaktperson:	Børge Lundhaug

Sammendrag:

Følgende rapport beskriver gjennomføring av bacheloroppgaven ved studieretningen Elektronikk og Sensorsystemer som tilhører programmet Bachelor i ingeniørfag, Elektro ved Norges Teknisk-Naturvitenskapelige Universitet i Trondheim. Oppgaven ble utført i samarbeid med Skanska Norge AS som presenterte en problemstilling innenfor datakommunikasjon i tunnelanlegg med mål om å effektivisere parameterovervåkning. Dette ble løst ved å implementere et IoT-system som registrerer forholdene i underjordiske anleggsområder. Systemet er bygget rundt en sensorboks som inneholder en oksygensensor, støvsensor, gassensor og temperatur- og fuktighetssensor. Avleste data blir sendt til en tjener på ESP32 ved hjelp av LoRa og deretter lastet opp via WiFi til en skylagringstjeneste Ubidots. Selv om testene av systemet i et tunnelanlegg under sprengning og massetransport ble fullført, kan mangler under testen gjøre sluttresultatet mindre realistisk. Senere ble det gjennomført en fullstendig test av sensorsystemet og LoRa-rekkevidden, hvor systemet fungerte opp til maksimumsrekkevidden på 700 meter. Selv om systemet fungerer bra finnes det ennå forbedringspotensiale som å oppgradere til industrielle sensorer, bygge en sterkere innkapsling, sette opp varslingssystem og implementere fjernstyring med Ubidots.

Abstract:

The following report describes the completion of the bachelor's thesis in the field of Electronics and Sensor Systems, as part of the Bachelor's program in Electrical Engineering at the Norwegian University of Science and Technology (NTNU) in Trondheim. The thesis was carried out in collaboration with Skanska Norge AS, which presented a research problem related to data communication in tunnel facilities, with the aim of improving parameter monitoring. This was achieved by implementing an IoT system that monitors conditions in underground construction areas. The system is built around a device that consists of oxygen, gas, temperature, dust and humidity sensors. The collected data is sent to an ESP32 microcontroller through LoRa and then uploaded to the Ubidots's cloud system through WiFi. Although the tests of the system in a tunnel facility during blasting and mass transport were completed, there may be limitations during the testing phase that could affect the overall realism of the results. Subsequently, a comprehensive test was conducted to verify sensor functionality as well as the measure the LoRa range, which resulted in a maximum range of approximately 700 meters. Even though the system is fully functioning, there are multiple potential improvements to be implemented, such as upgrading the sensors to industrial versions, building a tougher and more resistant sensor chassis, implementing a status indicator and finally remote commands using Ubidots.

Stikkord:	**Keywords:**
IoT, LoRaWAN, tunnelanlegg, Ubidots, Skylagring, Sensorer, Mikrokontroller, Stuff	IoT, LoRaWAN, underground construction sites, Ubidots, Cloud, Sensors, Microcontroller, End of tunnel construction

Sammendrag

Følgende rapport beskriver gjennomføring av bacheloroppgaven ved studieretningen Elektronikk og Sensorsystemer som tilhører programmet Bachelor i ingeniørfag, Elektro ved Norges Teknisk-Naturvitenskapelige Universitet i Trondheim. Oppgaven ble utført i samarbeid med Skanska Norge AS som presenterte en problemstilling innenfor datakommunikasjon i tunnelanlegg med mål om å effektivisere parameterovervåkning. Dette ble løst ved å implementere et IoT-system som registrerer forholdene i underjordiske anleggsområder. Systemet er bygget rundt en sensorboks som inneholder en oksygensensor, støvsensor, gassensor og temperatur- og fuktighetssensor. Avleste data blir sendt til en tjener på ESP32 ved hjelp av LoRa og deretter lastet opp via WiFi til en skylagringstjeneste Ubidots. Selv om testene av systemet i et tunnelanlegg under sprengning og massetransport ble fullført, kan mangler under testen gjøre sluttresultatet mindre realistisk. Senere ble det gjennomført en fullstendig test av sensorsystemet og LoRa-rekkevidden, hvor systemet fungerte opp til maksimumsrekkevidden på 700 meter. Selv om systemet fungerer bra finnes det ennå forbedringspotensiale som å oppgradere til industrielle sensorer, bygge en sterkere innkapsling, sette opp varslingssystem og implementere fjernstyring med Ubidots.

Abstract

The following report describes the completion of the bachelor's thesis in the field of Electronics and Sensor Systems, as part of the Bachelor's program in Electrical Engineering at the Norwegian University of Science and Technology (NTNU) in Trondheim. The thesis was carried out in collaboration with Skanska Norge AS, which presented a research problem related to data communication in tunnel facilities, with the aim of improving parameter monitoring. This was achieved by implementing an IoT system that monitors conditions in underground construction areas. The system is built around a device that consists of oxygen, gas, temperature, dust, and humidity sensors. The collected data is sent to an ESP32 microcontroller through LoRa and then uploaded to the Ubidots Cloud system through WiFi. Although the tests of the system in a tunnel facility during detonation and mass transport were completed, there may be limitations during the testing phase that could affect the overall realism of the results. Subsequently, a comprehensive test was conducted to verify sensor functionality as well as the measure the LoRa range, which resulted in a maximum range of approximately 700 meters. Even though the system is fully functioning, there are multiple potential improvements to be implemented, such as upgrading the sensors to industrial versions, building a tougher and more resistant sensor chassis, implementing a status indicator and finally remote commands using Ubidots.

Forord

Det er med stor glede og takknemlighet at vi nå presenterer denne bacheloroppgaven. Dette prosjektet har vært en reise fylt med læring, utfordringer og vekst, og det har vært et privilegium å ha så mange fantastiske mennesker rundt oss som har bidratt til vår suksess.

Gjennom arbeidet med denne oppgaven har hver enkel av oss fått gleden av å bli kjent med fantastiske Lourenzo, Erik, Edvard og Julia. Vi ønsker å rette en dypfølt takk til hver av oss for engasjementet, samarbeidet og ikke minst vennskap. Vi har lært utrolig mye fra hverandre og opplevd en utvikling både faglig og personlig som vi er stolte av.

Videre ønsker vi å rette en hjertelig takk til Børge Lundhaug, Audun Kvalnes og Gudmund Dyrland fra Skanska Norge AS. Vi er takknemlige for at dere alltid stilte opp og svarte på våre spørsmål, var tålmodige med oss og ga oss den tilliten vi trengte for å gjennomføre prosjektet på best mulig måte.

En spesiell takk rettes også til Herman Ranes, vår veileder. Din støtte, oppmuntring og faglige veiledning har vært uvurderlig gjennom hele prosessen. Du har pushet oss til å jobbe systematisk og strekke oss lenger, ga oss verdifulle tips og var en inspirasjon for oss. Vi er takknemlige for at vi har hatt muligheten til å jobbe sammen med deg.

Videre ønsker vi å takke Lars Onsager for all underholdningen og de lystige øyeblikkene som bidro til å gjøre arbeidsmiljøet både trivelig og energisk. Vi retter også en spesiell takk til Marius Hammer for helgebrevet som alltid fikk oss til å smile og ga oss en velfortjent pause fra oppgaveskrivingen. Til slutt ønsker vi å takke Even Christiansen for hjelp med utstyr og programmering.

Å fullføre bachelorgraden har vært en prosess fylt med blod, svette, tårer og korona og vi hadde ikke klart oss uten våre brødre og søstre i India som har bidratt med utellelige YouTube-videoer om alt fra hvorfor man ikke skal koble katoden på en LED mot strømforsyningen til dimensjonering av PID-regulatorer i frekvensplanet. Når vi satt alene på hybelen uten noe håp om å noensinne lykkes i livet så var det dere som har vist oss lyset i tunnelen og forklart sakte men sikkert alle hemmelighetene av elektroingeniøryrket.

Det har vært en utrolig morsom og lærerik opplevelse å arbeide med problemstillingen i denne oppgaven. Vi har fått en verdifull smaksprøve av arbeidslivet og erfart betydningen av et sterkt og samarbeidsorientert team. Vi håper at denne rapporten kan legge grunnlaget for videreutvikling og gi verdifull innsikt for fremtidige prosjekter innenfor fagfeltet. Til slutt vil vi rette en stor takk til alle som har bidratt til denne oppgaven på ulike måter. Uten deres støtte og innsats ville ikke denne oppgaven vært mulig. Vi håper at vårt arbeid kan være til nytte og inspirasjon for andre, slik som vi har blitt inspirert av dere.

Tusen takk!

Innhold

Figurliste

1. Innledning

1.1. Bakgrunn

Spredningen av IoT-teknologier og kunstig intelligens i industrien som man har observert de siste ti årene har fått en felles betegnelse «Den fjerde industrielle revolusjonen». Den første, andre og tredje industrielle revolusjonen har på hver sin måte endret retningen til menneskehistorie. Det faktum at dagens utvikling av sensorteknologi stilles på lik linje med de førstnevnte understreker betydningen til denne vendingen i verdensutvikling som tar sted foran våre øyne. Stadig flere industrigrener blir digitalisert og koblet opp mot «tingenes internett» og denne trenden eskalerer og overkommer alle hindringer som før i tiden virket ubegripelige. Begrenset tilgang til strømforsyning og internettilkobling, behov for vedlikehold, fare for mekaniske skader og lang avstand fra sensorene til datainnsamlingsenhetene får stadig mindre betydning, noe som muliggjør implementering av mer avanserte sensorløsninger. [1]

Det er blant annet i byggebransjen man observerer et voksende behov for smarte løsninger. I de fleste tilfeller er IoT-systemer relativt enkle å implementere da det finnes flere kommunikasjonsprotokoller som egner seg til slike bruksområder og utellelig med komponenter som støtter disse protokollene. [2]

Underjordiske anleggsområder derimot har begrensede muligheter for kommunikasjonsløsninger. De er meget utfordrende arbeidsmiljøer med mange faktorer som påvirker effektivitet og sikkerhet. Siden arbeidet pågår dypt under bakkenivå er kommunikasjonsalternativene svært begrensede. Tykke vegger mellom nærliggende rom absorberer alle bølgelengder fra det elektromagnetiske spekteret og trådløs kommunikasjon er derfor oppnåelig kun i åpent felt. [3] Dessuten strekker slike anleggsområder seg ofte over flere kilometer med areal og utvider seg nesten hver dag. Arbeid pågår som regel på flere ulike steder samtidig, og koordinasjon av hele anlegget er derfor essensiell for både sikkerhet og effektivitet. All infrastruktur som må til for både trådløs og kablet kommunikasjon kan bli ødelagt av kommende partikler under eksplosjoner. Forholdene i tunnel er svært krevende og preget av høyt støynivå, stor risiko for vann- og gasslekkasje, direkte sikkerhetstrussel ved sprengning av fjell og bruk av borerigg samt intensiv maskintrafikk. For at man skal kunne sikre trygge omgivelser for personellet er man nødt til å overvåke maskineriet og de ulike parametere. [4]

I forbindelse med forberedelse til prosjektet har prosjektgruppen fått invitasjon til en omvisning i et tunnelanlegg og fått forklaring på arbeidsprosessene der. Det ble blant annet nevnt at mye av datainnsamlingen foregår manuelt da sensorene må kalibreres og renses før de skal brukes. I noen tilfeller blir ikke registrerte data overført digitalt til sentralen, men blir istedenfor notert av en ansatt som er da nødt til å samle inn avleste verdier fra flere ulike punkter i anlegget for hånd. Noen sensorer registrerer data automatisk, men har ikke noe løsning for varsling ved dataoverflyt. Det vil si at de målte verdiene kan bli overskrevet av nye data før de sendes til sentralen. Selv om tunnelanlegg har WiFi-antenner installert så kan sensorene miste tilkoblingen til nettverket i ethvert øyeblikk uten å gi varsel om dette. Dermed er det behov for å kontrollere tilkobling med jevne mellomrom. Med alt dette tatt i betraktning, er det mye forbedringspotensial på kommunikasjonsfronten for tunnelanlegg i dag.

1.2. Formål

I oppgavebeskrivelsen uttrykket oppdragsgiveren at under tunneldriving er det utfordrende å opprettholde stabil datakommunikasjon frem på stuff og man er ofte avhengig av at personellet vedlikeholder måleinstrumentene manuelt. Det var derfor ønsket å implementere et IoT-system som skal kunne bidra til effektivisering av prosessene og en bedre kvalitet på samlede data. I tillegg ble det etterspurt å minimere behovet for at personell kjører rundt og sjekker status på anleggsdeler og utstyr.

1.3. Problemstilling

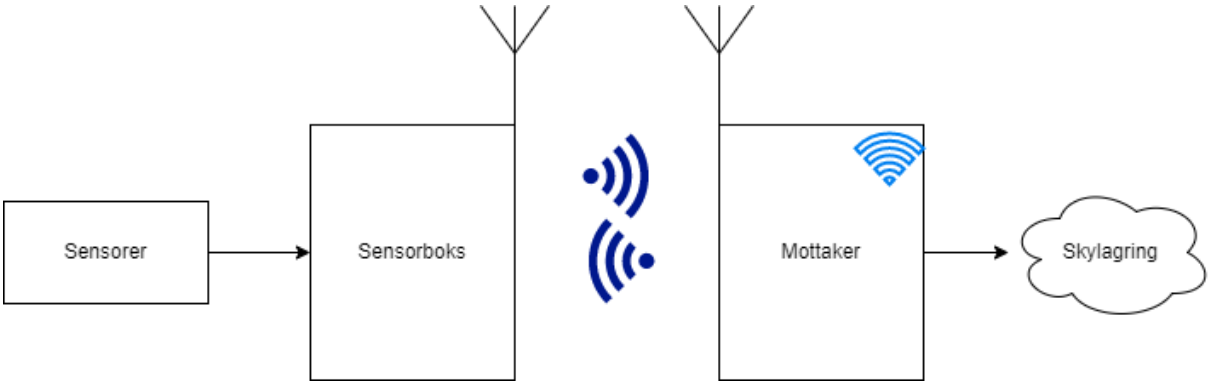
På bakgrunn av observasjonene gjort under besøket på et tunnelanlegg, utfordringene presentert av Oppdragsgiver og tilgjengelig kompetanse i prosjektgruppen ble det valgt å drøfte følgende problemstilling:

Parameterovervåkning i tunnelanlegg ved bruk av LoRaWAN-basert IoT-system

Ovennevnte problemstilling kan deles opp i følgende hovedmål:

- Selvkalibrerende sensorboks for overvåkning av støv-, oksygen-, gass- og karbondioksidnivå
- Automatisk dataoverføring ved kort avstand fra sensorboksen til flyttbar serverenhet ved hjelp av LoRa
- Dataopplasting til en skylagringstjeneste
- Varsling ved behov for vedlikehold av sensorboks

Systemets oppbygning er visualisert i blokkdiagrammet i Figur 1.



Figur 1 Blokkdiagrammet av systemet

1.3.1. Avgrensning av prosjektet

Prosjektets omfang ble fastsatt i samarbeid med Oppdragsgiveren. Prosjektet omfatter ikke implementering av løsninger for data- og kommunikasjonssikkerhet. Det ble heller ikke vektlagt å utvide brukergrensesnittet på skylagring med tilleggsfunksjoner, men i stedet benyttet de forhåndsdefinerte funksjonene som den valgte skytjenesten tilbyr. Systemet består kun av kostnadseffektive og lett tilgjengelige komponenter som har begrenset levetid og ikke har noe dokumentert ytelse i tunnelanlegg fra tidligere.

1.4. Rapportens oppbygning

Rapporten er formulert for å lett kunne tolkes av personer med noe kompetanse innenfor ingeniørfag og elektroteknikk. Det refereres til kunnskap fra grunnleggende programmering, elektronikk og trådløs kommunikasjon. Til tross for dette ble det sørget for at en leser uten en slik bakgrunn skal få muligheten til å forstå de grove trekkene i prosjektbeskrivelsen.

Rapporten begynner med definisjonen av problemstilling, samt forklaring av motivasjonen for valg av denne. Videre gir kapittelet 2. Teori en grundig gjennomgang av den relevante kunnskapen som man bør tilegne seg for å få en fullstendig forståelse av systemets virkemåte, både når det gjelder trådløs kommunikasjon, mikrokontrollere, programmering og 3D-modellering.

Deretter introduseres 3. Metode der det beskrives oppkobling av alle sensorer og mikrokontrollere og kode som er opplastet på disse. I tillegg omtaler kapittelet utført 3D-modellering, systemarkitekturen og sensorkalibrering. I 4. Resultater beskrives det utføring av testene på tunnelanlegget og i klasseromsmiljø. Det endelige produktet presenteres visuelt, både på maskinvare- og programvaresiden.

Kapittelet 5. Diskusjon inneholder gjennomgang av de presenterte resultatene, foreslår videre arbeid og påpeker forbedringspotensialet ved prosjektutføring. Til slutt oppsummerer kapittelet 6. Konklusjon de viktigste funnene og trekker en konklusjon for prosjektet.

1.5. Definisjoner og forkortelser

Ord, Begrep og Forkortelser	**Definisjon**
ADC	Analog to Digital Converter. Gjør analoge signaler om til digitale.
API	Application Programming Interface. Beskriver hvordan programvare kommuniserer med en annen programvare.
Borerigg	Maskin som brukes til å bore hull i fjell for å konstruere tunneler.
CSS	Chirp Spread Spectrum. Trådløs kommunikasjonsteknikk, bruker pulser med økende eller minkende frekvens for å sende data.
Eng.	Engelsk
ESP32	ESP32 er en avansert mikrokontrollerenhet med innebygd WiFi- og Bluetooth-funksjonalitet, som brukes i en rekke elektronikk- og IoT-applikasjoner.
G.code	G-kode er en standardisert språklig kode som brukes i 3D-utskrift for å kontrollere og instruere bevegelsen og handlingene til en 3D-skriver.
HTTP	Hypertext Transfer Protocol. Dataoverføringsprotokoll mellom klient og nettjener
I2C	Inter Integrated Circuit. Seriell kommunikasjon som tillater flere slaver for en master.
IoT	Internet of Things
KOLS	Kronisk Obstruktiv Lungesykdom
Lipo	Lipo er en forkortelse for litium-polymer-batteri, som er en type oppladbart batteri som benytter seg av litiumioner og polymerelektrolytt for å lagre og frigjøre elektrisk energi.
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
MQ-2	MQ-2 er en gassensor som brukes til å oppdage og måle konsentrasjonen av forskjellige brennbare gasser i luften.
Oppdragsgiver	Skanska Norge AS
PM	Particulate Matter. Partikler i lufta f.eks. støv.

RPi	Raspberry Pi
RSI-styrken	RSI-styrken (Relative Strength Index) er en teknisk indikator som brukes til å vurdere om et finansielt instrument er overkjøpt eller oversolgt basert på historisk prisutvikling.
Salve	Ordet "salve" refererer til en gruppe eller serie av eksplosjonsladninger som blir utløst samtidig.
SCD30	SCD30 er en sensor som brukes til å måle konsentrasjonen av karbondioksid (CO2) i luften.
Slicing	"Slicing" i 3D-utskrift refererer til prosessen med å dele en tredimensjonal modell i tynne horisontale lag for å instruere 3D-skriveren om hvordan objektet skal bygges lag for lag.
SPI	Serial Peripheral Interfact. Serial kommunikasjon
SSH	Secure Shell Protocol. Trygg måte å få tilgang til en datamaskin over et usikret nettverk.
Stuff	Område i tunnelfronten der det bores og sprenges fjell [4]
TBM	Tunnel Boring Machine
UART	Universal Asynchronous Receiver/Transmitter

2. Teori

2.1. Innhenting av informasjon og kunnskap

2.1.1. Helseeffekter av luftforurensning

God luftkvalitet er viktig for å bevare god helse, og luftforurensing er den mest skadelige miljøfaktoren for helsen. Det er beregnet over tusen tidlige dødsfall årlig som følge av luftforurensing i Norge. På verdensbasis ligger dette tallet på tre til fire millioner dødsfall årlig. Både kort og langvarig eksponering for luftforurensning, alt fra noen timer til flere år er skadelige for helsen. Personer med tidligere luftveissykdommer og hjerte- og karlidelser er spesielt utsatte for luftforurensning. Det er stoffene i uteluft som fører til sykdom og død av tidligere infeksjoner og sykdommer som astma, KOLS, hjerteinfarkt og slag. Ulike typer for svevestøv og gasser er eksempler på luftforurensningskomponenter som kan gi helseskadelige effekter, og kan fort oppstå under tunnelarbeid. [5]

2.1.2. Luftforurensning i tunnelanlegg

I tunnelanlegg er det oftest etter sprengning når luften blir forurenset av støvpartikler som spres rundt i tunnelen og gassene som dannes av eksplosjonene. Maskineri bidrar også til forurensningen i form av karbonforbindelser, som oftest karbondioksid (CO_2) og karbonmonoksid (CO). Forurensningskomponentene i tunnelen har høy konsentrasjon både på grunn av begrenset luftmengde, samt dårligere tilgang til fersk uteluft. Den høye konsentrasjonen av luftforurensningskomponenter er ofte farligere enn forurensingen som skjer i for eksempel byer og på motorveien. [6]

2.1.3. Lære om LoRaWAN

Gjennom vår innsamling av kunnskap om LoRaWAN-teknologien, ble det oppnådd en bedre forståelse av de viktige elementene som er involvert i implementeringen av et LoRaWAN-basert system. Det ble undersøkt funksjonene til de ulike komponentene som kreves, for eksempel LoRaWAN-noder, gateway-enheter og nettverksinfrastruktur. Dette ga oss innsikt i hvordan disse komponentene samhandler for å muliggjøre trådløs kommunikasjon over lange avstander med lavt strømforbruk.

Valg av riktig frekvensbånd for LoRaWAN-drift er også viktig, det er avhengig av regionale forskrifter og tilgjengelighet av frekvensressurser. Dette inkluderte å lære om forskjellen mellom de ulike frekvensbåndene, for eksempel EU 868 MHz og US 915 MHz, og hvordan man kan optimalisere ytelsen og rekkevidden til LoRaWAN-systemet basert på valgt frekvens. [7]

Videre har vår læring om LoRaWAN-teknologien gitt oss muligheten til å vurdere potensielle integrasjoner og utvidelsesmuligheter. Vi har utforsket hvordan LoRaWAN kan samhandle med andre IoT-teknologier og plattformer, for eksempel skybasert lagring og analyseverktøy. Dette åpner opp for nye muligheter for å utnytte dataene som samles inn fra sensorer, forbedre beslutningsprosesser og implementere avanserte analysemetoder.

En annen viktig del av vår læringsprosess var å utforske potensielle begrensninger og utfordringer knyttet til LoRaWAN-teknologien. Dette inkluderte å identifisere rekkeviddebegrensninger, påvirkning av interferens og signalforstyrrelser, samt muligheten for å skalere opp nettverket for å dekke store områder. Ved å være klar over disse utfordringene på forhånd, kan vi planlegge og implementere tiltak for å håndtere dem på en effektiv måte.

2.1.4. Lære om 3D-modellering og utskrift

Først må man ha en ide om hva man ønsker å lage. Dette kan man tegne opp eller modellere i en 3D-programvare som Tinkercad. Deretter eksporteres modellen som en STL-fil. Når STL-filen er eksportert, må man klargjøre filen for 3D-utskrift ved hjelp av en slicer-software som kan konvertere STL-filen til en GCODE-fil som 3D-skriveren kan forstå. Slicer-programmer som Cura kan brukes til dette formålet. I slicer-programmet må man sørge for at dimensjonene på modellen er riktige, at veggtykkelse og hullstørrelser er tilstrekkelige for utskriftsteknologien som skal brukes, og at modellen er optimalisert for styrke og stabilitet. [8] [9]

Når man har valgt teknologi og materiale, er det neste å sende modellen til utskrift. Dette kan gjøres ved å laste opp GCODE-filen til en lokal 3D-skriver. Under utskriftsprosessen bygger skriveren opp modellen lag på lag, basert på informasjonen i GCODE-filen. Det kan ta alt fra noen minutter til flere timer eller dager å fullføre utskriften, avhengig av størrelse og kompleksitet på modellen. [8]

Til slutt vil man sitte igjen med en fysisk modell av den opprinnelige ideen. Modellen kan nå brukes til en rekke formål, for eksempel prototyping, produksjon, visualisering eller utdanning. 3D-utskrift gir en fleksibilitet og kreativ frihet som gjør det mulig å lage objekter som ellers ville vært vanskelige eller umulige å produsere på andre måter. [8]

2.2. Teknisk teori:

2.2.1. Trådløs kommunikasjon

2.2.1.1. CSS

CSS (eng. *Chirp spread spectrum*) er en teknikk for signalmodulasjon spesielt egnet for dataoverføring over lange avstander. Den baserer seg på at bærebølgen får formen til en såkalt *chirp* som er en sinusbølge med en kontinuerlig synkende eller stigende frekvens. På grunn av dette kan informasjon sendes over et bredt frekvensbånd da det originale signalet blir rekonstruert ved hjelp av korrelasjon hos mottaker. Teknikken kjennetegnes av lav effekt i forhold til liknende teknologier. En annen fordel ved å bruke CSS er høy immunitet mot støy og interferens, nettopp fordi signalet demoduleres ved hjelp av korrelasjon, noe som minimerer risikoen for tap av enkelte symboler og dermed signalforvrengning. [10] [11]

2.2.1.2. LoRa

LoRa (eng. *Long Range*) er en trådløs kommunikasjonsteknologi som er spesielt egnet for IoT-systemer, da den muliggjør tosidig dataoverføring over lange distanser med minimalt effektforbruk. Teknologien tilhører det fysiske laget (eng. *physical layer*), mens kommunikasjonsprotokollen er definert av LoRaWAN som tilhører nettverkslaget og omtales i kapittel 2.2.1.3. LoRa ble oppfunnet av det franske selskapet *Cycleo* som senere ble overtatt av den amerikanske IT-leverandøren *Semtech*. I 2015 etablerte Semtech LoRa Alliansen som er den internasjonale foreningen for utvikling av LoRa. Organisasjonen jobber for teknologiens interoperabilitet og ekspansjon, samt definerer LoRaWAN-protokollen. [12]

LoRa-moduler opererer under lav spenning, ofte 3,3 V eller 5 V, og krever dermed ikke tilkobling til strømnettet, men kan heller bruke strømforsyning fra batterier. Hovedgrunnen til det lave effektforbruket er implementering av en moduleringsteknikk som ligner på CSS. De typiske bruksområdene legger til rette for ytterligere strømsparing da de overvåkede parametere i IoT-systemer blir som regel målt med visse mellomrom og utstyret behøver derfor ikke å være slått på kontinuerlig, men kan sendes i sovemodus når ikke i bruk. [13]

LoRa opererer i en frekvens som ikke krever en lisens for bruk. Denne frekvensen varierer avhengig av geografisk plassering. I USA er det 915 MHz, i Europa 868 MHz og i Asia 433 MHz. Modulasjonsteknikken er en spesialtilpasset versjon av CSS-teknologi. Båndbredden er på enten 125 kHz eller 500 kHz, avhengig av retningen for dataoverføring. Skillet mellom disse frekvenskanalene muliggjør effektiv toveiskommunikasjon. En viktig tilpasning av CSS

er optimaliseringen av spredningsfaktoren til de individuelle nodene i nettverket, basert på avstanden fra gateway. Spredningsfaktoren øker i takt med avstanden. [14]

Ifølge LoRa Alliance er den maksimale registrerte rekkevidden til LoRa-kommunikasjonen 766 km, men slike avstander kan kun oppnås ved klar sikt. Under normale forhold og med klar sikt til mottaker, har LoRa muligheten til å overføre data over femten kilometer. I urbane miljøer derimot er rekkevidden redusert og overstiger ikke fem kilometer. [15] [16]

LoRa støtter toveiskommunikasjon, noe som betyr at en avsender kan operere som en mottaker, og vice versa. Dette åpner for fjernt vedlikehold da det ikke er nødvendig med fysisk tilkobling til enheten for å oppdatere programvaren. I stedet kan oppdateringer sendes trådløst fra andre enheter.

Til tross for at LoRa tilbyr viktige fordeler som lang rekkevidde og lavt effektforbruk, har teknologien ikke den samme høye overføringshastigheten som andre IoT-teknologier som Bluetooth og WiFi. Bitraten avhenger av båndbredden og varierer mellom 0,3 kbit/s til 27 kbit/s. [15] [17]

2.2.1.3. LoRaWAN

LoRaWAN står for Long Range Wide Area Network og er nettverksprotokollen som er utviklet for LoRa-teknologien. Den ble formulert av LoRa Alliance som også er ansvarlig for kontinuerlig revisjon av spesifikasjonene. Organisasjonen overvåker LoRa-kompatibiliteten til elektronisk utstyr på markedet ved å sertifisere produkter som oppfyller kravene fastsatt av protokollen. [18]

LoRaWAN-nettverk har enten stjerne- eller utvidet stjerne-topologi (eng. *extended star topology,* også kjent som *star-of-stars*). Denne egenskapen muliggjør sammenkobling av et stort antall individuelle LoRa-noder og gjør teknologien svært gunstig for bruk i smartbyer, store bygninger og komplekse IoT-systemer. Den høye kapasiteten kan oppnås uten betydelige investeringer da LoRaWAN-infrastrukturen består av få påkrevde komponenter som også er kostnadseffektive. Brukskostnadene reduseres ytterligere på grunn av at LoRaWAN-nettverksprogramvare som er åpen kildekode (eng. *open source*).

Ende-til-ende-kryptering i henhold til AES-128 standarden, implementert både i nettverks- og applikasjonslaget ivaretar sikkerheten til LoRaWAN-baserte nettverk. Enhver LoRa-sertifisert enhet tildeles et unikt identifikasjonsnummer som brukes til å aktivere og administrere enheten, samt identifisere den i nettverket. Gjenkjenning av nodens adresse finner sted hos

tjeneren. Nodene er ikke begrenset til en bestemt gateway, men kan kommunisere med ulike gateways. [7]

LoRaWAN anvender prinsippet av tidsforskjellen ved ankomst (eng. *The Time Difference on Arrival*, forkortet TDOA) til å estimere posisjonen til enhetene i LoRaWAN-nettverket. Det betyr at behovet for installering av en ekstern GPS-modul i LoRa-utstyret elimineres. [15]

2.2.1.4. WiFi

Wi-Fi er en trådløs kommunikasjonsteknologi som gjør det mulig for enheter å koble seg til internett eller utveksle data uten behov for fysiske kabler. Wi-Fi-teknologien er basert på standardene som er definert av Institute of Electrical and Electronics Engineers (IEEE) 802.11-familien. Denne familien inkluderer ulike standarder som IEEE 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac og 802.11ax, som alle tilbyr forskjellige hastigheter og funksjoner. Wi-Fi bruker radiosignaler til å overføre data mellom enheter og en trådløs ruter eller tilgangspunkt. Ruterne fungerer som broer mellom de trådløse enhetene og det kablede internettet. Wi-Fi-operasjonen er basert på bruk av ulike frekvensbånd, inkludert 2,4 GHz og 5 GHz, avhengig av den aktuelle standarden.

Et Wi-Fi-nettverk består av følgende komponenter: En trådløs ruter eller tilgangspunkt som styrer nettverket og gir tilkobling til internett; Wi-Fi-klientenheter som bærbare datamaskiner, smarttelefoner eller iot-nheter som er koblet til nettverket; og radiosignaler som overfører data mellom disse enhetene. Wi-Fi-teknologien benytter seg av forskjellige teknikker som Multiple Input Multiple Output (MIMO) for å forbedre ytelsen og rekkevidden. MIMO bruker flere antenner for å oppnå bedre overføringshastigheter og økt kapasitet ved å utnytte flerbaneteknikker.

Videre har Wi-Fi-nettverk sikkerhetsfunksjoner som Wi-Fi Protected Access (WPA) og WPA2 for å beskytte dataene som overføres i nettverket. Disse sikkerhetsprotokollene benytter seg av krypteringsmetoder for å sikre konfidensialiteten og integriteten til dataene. [19]

2.2.2. Periferienheter på mikrokontrollere

2.2.2.1. I2C

I2C, også kjent som *IIC, TWI* eller *Inter Integrated Circuit*, er en databuss for kablet, seriell, bidireksjonal kommunikasjon i mikrokontrollere. I denne protokollen definerer man en eller flere enheter som styrer kommunikasjonen (master) og deretter aksesserer de underordnede enhetene (slaver) gjennom deres individuelle adresser. Ved initiering av kommunikasjonen er det nødvendig å avklare retningen til dataoverføring, altså om master skal motta eller sende data. Etter at hver byte er overført, sender mottakeren en ACK-bit (eng. *acknowledgement*) for å bekrefte vellykket operasjon. Kommunikasjon er synkron, og man behøver derfor å implementere en klokkelinje (eng. SCL) mellom de tilkoblede enhetene. Den andre kommunikasjonslinjen er selve datalinjen (SDA) for dataoverføring. [20]

2.2.2.2. UART

UART står for *Universal Asynchronous Receiver Transmitter* og er en databuss for seriell kommunikasjon i mikrokontrollere. I UART definerer man de to kommunikasjonsdeltakere: TX (eng. *trasmitter*) og RX (eng. *receiver*). Kommunikasjonen foregår uten et klokkesignal, men det brukes en paritetsbit til å indikere hvilke data som skal ankomme mottaker. På grunn av dette er kommunikasjonen begrenset til én avsender og én mottaker, i motsetning til I2C som kan kobles til flere slaver om gangen. Databussen kan også implementeres synkront og blir da kalt *Universal Synchronous Receiver Transmitter*. [21]

2.2.2.3. ADC

Analog-til-digital omformer eller ADC (eng. *Analog-to-Digital Converter*) er en krets som gjør analoge signaler om til digitale pulstog ved å anvende sampling. Prosessen foregår ved å gjøre et bestemt antall målinger i diskrete tidsintervaller med en frekvens som oppfyller Nyquist samplingsteoremet. Kretsen spiller en essensiell rolle i elektroniske enheter som ikke er i stand til å behandle analoge signaler, men kun binære verdier. ADC genererer enten en 10-bit eller 12-bit resultat. [22]

2.2.2.4. SPI

Seriell Periferigrensesnitt (eng. *Serial Peripheral Interface*, SPI) er en høyeffektiv databuss med master-slave systemarkitektur. Den viktigste egenskapen til denne protokollen er muligheten til å overføre data i begge retninger samtidig (eng. full-duplex). Kommunikasjonen er synkron og koordineres av masterenheten gjennom en klokkelinje (eng. *Serial Clock*, SCK). SPI tillater konfigurering av datahastigheten da klokkefrekvensen kan

defineres av brukeren. I tillegg til klokkelinjen krever SPI en MOSI-linje (eng. *Master Out Slave In*), en MISO-linje (eng. *Master In Slave Out*) og en SS-linje (eng. *Slave Select*). Ved implementering av flere SS-linjer kan flere slaver kobles til samme master. [23]

2.2.3. Biblioteker og moduler

Biblioteker og moduler er samlinger av forhåndsdefinerte funksjoner som kan påkalles i hovedprogrammet. Implementering av biblioteker reduserer lengden og kompleksiteten til koden og forenkler gjenbruk av kodesnutter. Et program som består av funksjoner delt inn i biblioteker er enklere å tolke for andre programmerere og øvrige interesserte på grunn av en mer oversiktlig struktur. Det er dessuten tids- og ressursbesparende å ta i bruk biblioteker skrevet av andre programmerere. [24]

2.2.4. 3D-modellering

2.2.4.1. Tinkercad

Tinkercad er en web-basert 3D-modellerings programvare som lar brukerne lage 3D-modeller ved hjelp av enkle former og verktøy. Det er en brukervennlig plattform som er designet for både nybegynnere og mer erfarne brukere, og krever ikke mye tid eller erfaring for å lære å bruke. [25]

En av fordelene med Tinkercad er at det ikke krever noen nedlastinger eller installasjoner. Det er en nettbasert plattform som gir brukere muligheten til å logge inn og starte modellering umiddelbart, uten å måtte installere programvare på egen maskin. Dette gjør Tinkercad til en tilgjengelig og praktisk løsning for 3D-modellering og design.

Tinkercad har en rekke verktøy og funksjoner som lar brukerne lage en rekke komplekse former og detaljerte modeller. Det inkluderer verktøy for å lage solide og hule objekter, importere og eksportere modeller, justere størrelsen og posisjonen til objekter, og legge til tekst og figurer. Tinkercad har også en rekke maler og biblioteker som gjør det enklere å lage spesifikke objekter, for eksempel enkle bokser, hjul og andre geometriske figurer.

Det er imidlertid viktig å merke seg at Tinkercad har noen begrensninger når det gjelder kompleksiteten på modellene som kan opprettes. Den har for eksempel en begrenset kapasitet til å håndtere store filer, og kan derfor ikke håndtere komplekse modeller som krever store mengder data eller detaljer.

Tinkercad er også begrenset i sine avanserte verktøy og funksjoner. For mer avanserte 3D-modellering og design, kan det være nødvendig å bruke mer avanserte 3D-programvare som

Fusion 360 eller SolidWorks. Men for de fleste enkle og mellomstore 3D-modellering og designbehov, kan Tinkercad være en god løsning.

Totalt sett er Tinkercad en enkel og tilgjengelig web-basert 3D-modellerings programvare som er egnet for nybegynnere og brukere med enkle designbehov. Det gir en rekke verktøy og funksjoner som lar brukerne lage enkle og komplekse modeller med liten innsats og erfaring.

2.2.4.2. Cura

Cura er en gratis og åpen-kilde slicing programvare som brukes av mange 3D-utskriftsentusiaster og fagfolk. Det gir en enkel og brukervennlig plattform for å konvertere 3D-modeller til G-code instruksjoner som kan leses av 3D-skriveren. [8]

Cura har en rekke funksjoner og innstillinger som gir brukerne kontroll over utskriftsprosessen og lar dem tilpasse hver enkelt utskrift for å oppnå best mulig resultat. Det inkluderer muligheten til å justere lagtykkelse, infill-tetthet, hastighet, temperatur og mange andre parametere som påvirker utskriftskvaliteten.

En av de mest imponerende funksjonene i Cura er dens evne til å generere støttestrukturer, som kan være nødvendig for å støtte overhengende deler av en modell. Disse støttestrukturene kan justeres og tilpasses for å minimere mengden støtte som brukes og redusere tiden som kreves for fjerning av støtten etter utskrift. Cura støtter også en rekke forskjellige filformater for import og eksport av modeller, inkludert STL, OBJ, og 3MF. Det gir også brukere muligheten til å laste ned og importere maler og profilinnstillinger fra et stort utvalg av forskjellige 3D-skrivermodeller.

Når det gjelder spesifikasjoner, så har Cura et bredt utvalg av innstillinger som kan tilpasses for å oppnå den ønskede utskriftskvaliteten. Det inkluderer muligheten til å justere lagtykkelse fra 0,05 mm til 0,4 mm, avhengig av skriverens kapasitet. Innfyll-tettheten kan justeres fra 0% til 100%, avhengig av ønsket styrke og tetthet på modellen.

Cura kan også justere hastigheten på utskriftsprosessen fra 10 mm/s til 300 mm/s. Temperaturen på skriveren kan også justeres fra 20°C til 300°C. [8]

2.2.4.3. Ultimaker 2+

Ultimaker 2+ er en FDM (eng. *Fused Deposition Modeling*) 3D-skriver som er designet for både hobbyister og profesjonelle brukere. Den er produsert av det nederlandske selskapet Ultimaker og er kjent for å være en pålitelig og brukervennlig skriver med høy utskriftskvalitet.

Ultimaker 2+ har en byggevolum på 223 x 223 x 205 mm, noe som gir brukerne god plass til å produsere større og mer komplekse modeller. Skriveren har også en oppløsning på opptil 20 mikrometer, som gir mulighet for høy presisjon og detaljerte utskrifter. Ultimaker 2+ bruker en rekke forskjellige materialer, inkludert PLA, ABS, Nylon og PVA. Dette gir brukerne muligheten til å produsere modeller med ulike styrkeegenskaper, farger og utseende. Skriveren har også mulighet til å bruke tredje parts materialer, noe som gir brukerne enda flere muligheter. [8] [9]

En av de viktigste funksjonene til Ultimaker 2+ er dens evne til å justere og tilpasse innstillinger for å oppnå best mulig resultat. Dette inkluderer justering av hastighet, lagtykkelse og innfyll-tetthet, samt temperatur og materialeinnstillinger. Skriveren har også mulighet for automatisk nivellering av byggeplaten, noe som gir jevnere utskriftsresultater og mindre problemer med kvaliteten.

En annen funksjon som skiller Ultimaker 2+ fra mange andre 3D-skrivermodeller, er dens evne til å skrive ut med to materialer samtidig. Dette gjør det mulig å produsere modeller med forskjellige egenskaper og farger, samt støttestrukturer som kan fjernes etter utskrift.

Generelt sett er Ultimaker 2+ en pålitelig og kraftig 3D-skriver som gir brukerne en rekke muligheter for å produsere modeller med høy kvalitet og detaljer. Med en god byggevolum og høy presisjon, samt muligheten for å skrive ut med to materialer samtidig, gir Ultimaker 2+ brukerne fleksibilitet og kontroll over utskriftsprosessen. Men det er viktig å merke seg at skriveren har sine begrensninger når det gjelder oppløsning og støtte for overheng, så det er viktig å velge riktige innstillinger og designe modellene på en måte som kan tilpasses skriverens kapasitet. Alt i alt er Ultimaker 2+ en anbefalt 3D-skriver for både nybegynnere og mer erfarne brukere som ønsker en pålitelig og brukervennlig plattform for 3D-utskrift.

3. Metode

3.1. Utstyr

3.1.1. Mikrokontroller

Mikrokontrolleren som ble valgt i denne oppgaven er LilyGo TTGO ESP32 LoRa32 V2.1.6 produsert av LilyGo som er et kompromiss mellom lav pris og god ytelse. Kjernen i den er brikken ESP32 som støtter trådløs kommunikasjon, WiFi, Bluetooth i tillegg til en rekke kablede kommunikasjonsmuligheter, deriblant UART, SPI, I2C, samt en ADC-krets. Utviklingsbrettet kombinerer en OLED-skjerm og en integrert LoRa-modul SX1276 som kan fungere både som mottaker og som en sender. LilyGo TTGO-enheten kan bruke batterier som strømforsyning. Dette gjør brettet gunstig å bruke i IoT-systemer da det lett kan forflyttes og monteres på steder uten fast tilkobling til strømnettet. Frekvensene det kan operere i er 868 MHz eller 915 MHz med toleranse ±15kHz. Overføringshastigheten er mellom 0,18-37,5 kbit/s. Det ble brukt SMA antenner som var vedlagt mikrokontrollere. [26]

3.1.2. Sensorer

3.1.2.1. Karbondioksid-, temperatur- og fuktighetssensor

SCD30 er en integrert karbondioksid-, temperatur- og fuktighetssensor utviklet av Sensirion. Sensoren benytter seg av NDIR (eng. *non-dispersive infrared*) teknologi for å måle mengden CO_2 i luften. NDIR fungerer ved å sende infrarødt lys med spesifikke bølgelengder som kan absorberes av CO_2-molekyler. Ved å måle absorpsjonen av dette lyset kan sensoren bestemme CO_2-nivået i luften. I tillegg til CO_2-måling har SCD30 også en termistor, som er en temperatursensor. Termistoren endrer motstandsverdien basert på temperaturforandringer i omgivelsene. Denne funksjonen gir sensoren muligheten til å måle temperaturen i luften rundt seg. Sensoren inneholder også en fuktighetssensor, som kan måle fuktigheten i luften. Fuktighet uttrykkes vanligvis i prosent, og sensoren gir ut fuktighetsverdien som en prosentandel av fuktighet i luften. [27]

Figur 2 Karbondioksid- og fuktighetssensor SCD30 fra Sensirion [27]

Figur 3 Gassensor MQ-2 [28]

3.1.2.2. Gassensor

MQ-2 fra elektronikkprodusenten *Seeed* er en gassensor basert på en kjemisk motstand (eng. *chemiresistor*). Resistansen i den endrer seg ved kjemiske reaksjoner med forskjellige gasser. Sensoren har muligheten til å detektere molekyler av metan (CH_4), propan (C_3H_8), hydrogen (H_2) og karbonmonoksid (CO), røykpartikler og alkohol. Sensoren detekterer tilstedeværelse og mengden av en av ovennevnte gasser, men har derimot ikke mulighet til å identifisere hvilken gass som har blitt registrert.

Gasskonsentrasjonen i et gitt miljø blir estimert ved hjelp av spenningen sensoren sender ut. Ved lav konsentrasjon av de ovennevnte gassene vil sensoren gi ut en spenning på 0 V. For å få tak i gasskonsentrasjonen til sensoren må de analoge signalene bli gjort om til spennings verdier.

$$SensorVoltage = \frac{sensorValue}{1024} * 5 \rightarrow rs_{gas} = \frac{5.0 - SensorVoltage}{SensorVoltage} \rightarrow global\ gas = rs_{gas}/R0$$

Ved økning av gasskonsentrasjonen blir utspenningen høyere. [28]

3.1.2.3. Partikkelsensor

Sensirion SPS30 er en liten og budsjettvennlig partikkelsensor som brukes til å måle nivåene av små partikler i luften, også kjent som partikulær materie (PM). Den bruker en optisk laser til å måle partikkelkonsentrasjonen i luften og gir nøyaktige målinger av partikkelmasse og konsentrasjon i flere størrelsesområder, hvorav PM2.5 og PM10 er de viktigste. Tallene etter PM beskriver diameteren til partiklene målt i µm, og det tas målinger av både stoffmengden og massekonsentrasjonen per kubikkmeter. Partikler under 10 µm i diameter er små nok til å

kunne reise gjennom luftveissystemet og havne i lungene, mens partikler under 2.5 µm kan reise dypere inn i lungene og i blodet. [5]

Sensoren består av en strømforsyning, en laserdetektor og en mikrokontroller. Lasereffekten og detektoren gir et pulserende signal, og mikrokontrolleren utfører en kalkulasjon av dette signalet for å estimere partikkelstørrelsen og konsentrasjonen. SPS30 kommuniserer med andre enheter via en seriell UART eller I2C-grensesnitt, og den kan enkelt innpasses med mikrokontrollere og andre systemer. Den kan brukes i en rekke applikasjoner, inkludert luftkvalitetsmåling, overvåking av inneklima, styring av HVAC-systemer, luftrenseanlegg og mer.

SPS30 har en innebygd vifte som forsørger at det er konstant luftflyt igjennom sensoren for å hente fram data. Ved lengre perioder vil sensoren ta opp en del støv der noe av støvet forventes å sette seg fast i sensoren. Derfor er det inkludert en funksjon som renser enheter. Når funksjonen er fremkalt vil viften i partikkelsensoren kjøre med en høyere effekt for å blåse bort støvet som setter seg fast. [29]

Figur 4 Partikkelsensor SPS30 fra Sensirion 8 [27]

Figur 5 Oksygensensor GGC2330-O [28]

3.1.2.4. Oksygensensor

GGC2330-O Grove Oxygen Sensor er en sensor som brukes til å måle oksygennivået i luften. Den bruker en elektrokjemisk celle til å utføre denne målingen, og den er spesielt utviklet for å fungere sammen med Grove-sensorkort. Den innebygde elektrokjemiske cellen reagerer med oksygen i luften og produserer et elektrisk signal som korrelerer med oksygennivået. Sensoren kommuniserer med andre enheter via en analog utgang, som gir et spenningsnivå

som tilsvarer oksygennivået som måles. Ellers krever den en spenningsforsyning på 5 V, og den har et lavt strømforbruk som gjør den egnet for batteridrevne applikasjoner. [30]

3.1.3. Batteri

PIS-1129 - PiJuice LiPo Battery fra Pi Supply er et oppladbart batteri som er spesielt utviklet for Raspberry Pi og PiJuice-utvidelseskort. Batteriet har en kapasitet på 12000 mAh og er designet for å gi langvarig strøm til mikrokontrollere. Batteriet bruker Lithium Polymer (LiPo) teknologi, som gir høy effektivitet og kapasitet i forhold til størrelsen. LiPo-batterier gir også mer stabil spenning enn tradisjonelle batterier, noe som er viktig for å sikre pålitelig og stabilt strømtilførsel til Raspberry Pi-enheten. Batteriet gir ut 4.2 V når det er maksimalt oppladet, og over tid vil spenningen synke. Siden spenningen går ned er det viktig å forebygge at den blir for lav. Om spenningen når lavere enn 3.2 V vil batteriet bli skadet og miste noe av kapasiteten. Batteriet har også en innebygd beskyttelseskrets som hindrer overbelastning, overoppheting og kortslutning. I tillegg er det utstyrt med en innebygd strømstyringsenhet som kan brukes til å overvåke og kontrollere strømforbruket til Raspberry Pi-enheten. [31]

$$\frac{12000}{\frac{120 * 0.001}{5} + \frac{(90 + 75) * 0.001}{5} + \frac{80}{5} + \frac{(14 + 160 + 10) * 3.998}{5}} = 73$$

Figur 6 Utregning av teoretisk batteritid. Estimert på 73 timer

3.1.4. RaspberryPi 3 Modell B+

Raspberry Pi 3B+ er en enkeltkortdatamaskin (eng. single-board computer) denne oppgaven brukes det operativsystemet Raspbian som er en variant av Linux.

Datamaskinen kan enten styres trådløst fra en annen datamaskin tilkoblet det samme Wifi-nettverket gjennom SSH eller ved å koble til en skjerm og tastatur. [32] Datamaskinen ble forsøkt implementert som tjenerenheten, men etter mange mislykkede forsøk ble den erstattet med mikrokontrolleren LilyGo som omtales i 3.1.1 Mikrokontroller.

3.1.5. Adafruit RFM95W

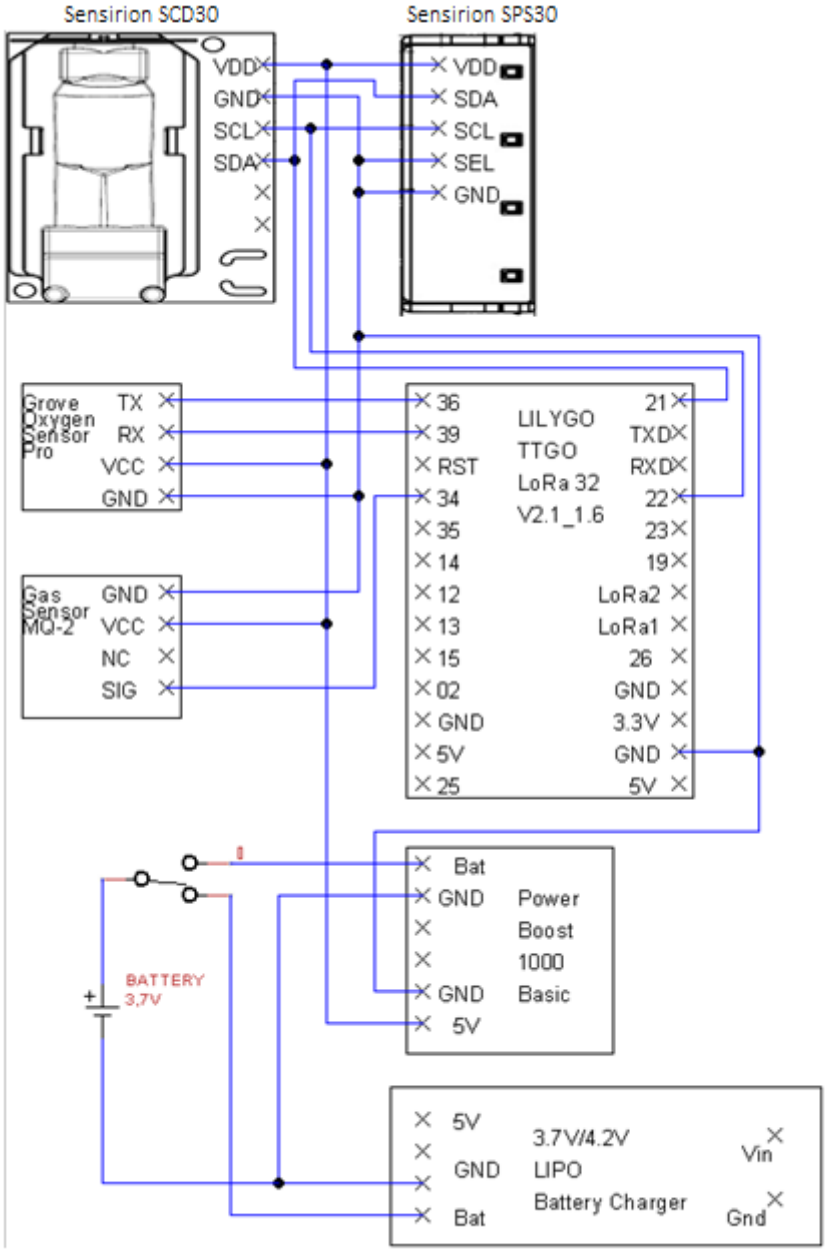
Adafruit RFM95W er en sender/mottaker-modul utviklet av Adafruit. Produktet er bygget rundt en SX1276-brikke som er en sertfisert LoRa-kompatibel enhet fra Semtech. Modulen kan brukes blant annet med Raspberry Pi og Arduino-baserte platformer. [33]

3.1.6. RGB LED

RGB LED brukes i varslingssystemet for å gi indikasjon på tilstanden til mikrokontrolleren. Ved å gi en riktig kombinasjon av basisfargene rød, grønn og blå kan man skape 16 777 216 unike farger. RGB-dioden kobles til pinnene 12, 13 og 14 gjennom 220Ω-motstander, i tillegg til å bruke jordpinnen GND.

3.2. Oppkobling

Oppkoblingen ble først testet på et koblingsbrett med ledninger (eng. *jumper wires*) og deretter gjenskapt på et prototypekort med alle sensorene loddet fast på, slik det er vist i Figur 15. I Figur 7 derimot ser man nøyaktig oppkobling av alle sensorene, batteriet og mikrokontrolleren i sensorboksen. Senere ble det også lagt til en RGB-lysdiode.



Figur 7 Kretsskjema

3.3. Programvare

3.3.1. Programmeringsmiljøer for utvikling av programvaren og grensesnittet

3.3.1.1.Arduino IDE

Arduino IDE (eng. *Arduino Integrated Development Environment*) et utviklingsmiljø med åpen kildekode som brukes til å programmere brett fra Arduino kretskort- familien og andre kompatible brett, blant annet ESP32. Programmeringsspråket som brukes er hovedsakelig Arduino C som er en mer brukervennlig utgave av C/C++. I tillegg til innebygde moduler for

de mest grunnleggende funksjonene i mikrokontrollere slik som WiFi, SPI, I2C osv. kan man importere eksterne eller brukerdefinerte biblioteker for bruk av tredjepartskomponenter slik som sensorer, LoRa-moduler og liknende. En viktig egenskap til Arduino IDE er Seriell Monitor som muliggjør overvåkning av seriell datastrøm. [34]

3.3.1.2.Geany

Geany er et tekstredigeringsverktøy og integrert programmeringsmiljø for bruk i de fleste operativsystemer og med mulighet til koding i ulike programmeringsspråk. Programmet er kompatibel med Raspbian og ble derfor brukt til utvikling av Python-kode for bruk av LoRa-modulen Adafruit RFM95W. [35]

3.3.1.3.Arduino IoT Cloud

Arduino IoT Cloud er en skybasert IoT-platform som kan brukes til trådløs styring og overvåkning av sensorsystemer. Platformen er spesialtilpasset for bruk med Arduino-baserte brett. For å få tilgang til platformen velger man antall enheter som skal tilkobles og en betalingsplan. Platformen kan åpnes både i nettleser, som et datamaskinprogram og som en applikasjon på smarttelefoner. Brukergrensesnittet består av dashboardet med tilhørende variabler som er knyttet til de respektive sensorene gjennom. [36]

*3.3.1.4.*Ubidots

Ubidots er også en skybasert IoT-plattform som brukes til å samle inn, visualisere og analysere data fra internett-tilkoblende enheter i sanntid. Plattformen har et brukervennlig grensesnitt og et sett med verktøy som gjør det mulig for brukere å opprette tilpassede applikasjoner og dashbord for å overvåke og styre IoT-enheter.

For å bruke Ubidots som programvare, begynner vi med å opprette en konto på plattformen og konfigurere tilkoblingen til våre IoT-enheter. Ubidots støtter et bredt spekter av protokoller og enheter, slik at man kan koble til og samle data fra ulike sensorer og enheter i nettverket. Når tilkoblingen er etablert, kan man begynne å definere variabler og opprette hovedside for å visualisere dataene. Ubidots tilbyr en rekke visuelle komponenter og diagrammer som gjør det mulig for oss å presentere dataene på en oversiktlig måte. Der kan man også tilpasse grensesnittet ved å legge til grafer, tabeller, kart og andre interaktive elementer som passer til våre spesifikke behov.

En annen viktig funksjon i Ubidots er evnen til å konfigurere alarmer og varsler. Ved å definere terskelverdier for bestemte variabler, kan vi motta varsler via e-post, SMS eller andre

kommunikasjonskanaler når dataene overstiger eller faller under gitte grenser. Dette gjør det mulig for oss å reagere raskt på avvik eller kritiske hendelser.

Prosjektgruppen har valgt å bruke Ubidots som programvareplattform på grunn av dens brukervennlighet, fleksibilitet og funksjonalitet. Ved å utnytte Ubidots' evner til datainnsamling, visualisering og analyse, har vi kunne effektivt overvåke og styre våre IoT-enheter og bruke dataene som resultater.

3.3.2. Kode

3.3.2.1.Biblioteker

`<SPI.h>` er et innebygd bibliotek utviklet av Arduino for bruk av SPI-databussen på Arduino-kompatible brett. Biblioteket inneholder funksjoner for konfigurering av modulen og dataoverføring. [37]

`<LoRa.h>` er et eksternt bibliotek utviklet av sandeepmistry, brukt til å konfigurere LoRa-radio, definere parametere som frekvens, signalstyrke, spredningsfaktor og båndbredde. I tillegg inkluderer moduler funksjoner for sending og mottak av data, nodeadressering, energisparing og feilsøking. [38]

`<sps30.h>` fra Sensirion samler funksjonene som er nødvendige for å kommunisere med partikkelsensoren Sensirion SPS30. Biblioteket initierer kommunikasjonen, tar seg av partikkeldeteksjon og beregning av konsentrasjonen, samt tilbyr støtte med feilsøking. I tillegg definerer det en funksjon for automatisk selvrensing av sensoren. [39]

`<Adafruit_SCD30.h>` fra Adafruit definerer funksjonene som påkalles for å initiere, styre og lese data fra karbondioksid- og fuktighetssensoren Adafruit SCD30. [40]

`<WiFiManager.h>` er et bibliotek utviklet av tzapu, som blir brukt for å koble ESP32 til internett fjernt ved hjelp av aksesspunktet det lager. [41]

3.3.2.2.Kodeflyt

Programmet for sensorboksen og den for mottakerenheten består i prinsippet av de samme elementene. I begynnelsen importerer man de nødvendige bibliotekene. Deretter definerer man såkalte konstanter, dvs. verdier som ikke skal forandres. Eksempler på dette er pinnenumrene, fargedefinisjonene i RGB-systemet og påloggingsinformasjon til WiFi-nettverket. Etter dette kommer `void setup()` der man initierer de nødvendige periferienhetene, Seriell Monitor og LoRa-kommunikasjonen. Som en del av dette er man nødt til å tilordne riktige pinner til de aktiverte databussene.

Den neste seksjonen i kode er `void loop()` som repeteres kontinuerlig så lenge mikrokontrolleren er slått på. I denne inngår det den viktige innebygde funksjonen `millis()` som brukes til å tidsdefinere de ulike hendelsene i koden. Funksjonen teller antall millisekunder som har passert siden mikrokontrolleren ble slått på. Etter omtrent 49 dager oppnår registeret overflyt, og tellingen begynner på nytt. Ved å oppgi antall millisekunder siden forrige hendelse kan man definere intervallene mellom sensoravlesninger og forespørsler om dataoverføring. I de neste underkapitlene beskrives de spesifikke programmene for sensorboksen og mottakerenheten.

3.3.2.2.1. Sensorboks

I `void loop()` er det en `lora_transmit()` – funksjon som henter og overfører data til mottakeren. Funksjonen består av fem underfunksjoner som henter data fra hver av sensorene og leser av batterispenningen. Det blir så opprettet en streng som inneholder alle verdiene fra hver av sensorene separert med en ¨|¨ (vertikal strek). Denne strengen vil så overføre data til igjennom LoRa ved hjelp av `LoRa.print(sensor_values)`.

```
LoRa.print(sensor_values);
void lora_transmit() {
  o2_read();
  gas_read();
  scd30_read();
  sps30_read();
```

```
  battery_read();

  //Lager string av sensorverdier med separatoren "|".
  String separator = "|";
  String sensor_values = global_gas + separator + global_o2 + separator +
global_particle_size + separator + global_dust_mc2_5 + separator +
global_dust_nc2_5 + separator + global_dust_mc10 + separator +
global_dust_nc10 + separator + global_temp + separator + global_co2 +
separator + global_humidity + separator + battery_voltage;

  Serial.print("Sending packet: ");
  Serial.println(counter);
  counter++;

  //Sender sensordata
  LoRa.beginPacket();
  LoRa.print(sensor_values);
  LoRa.endPacket();
  Serial.println(sensor_values);
```

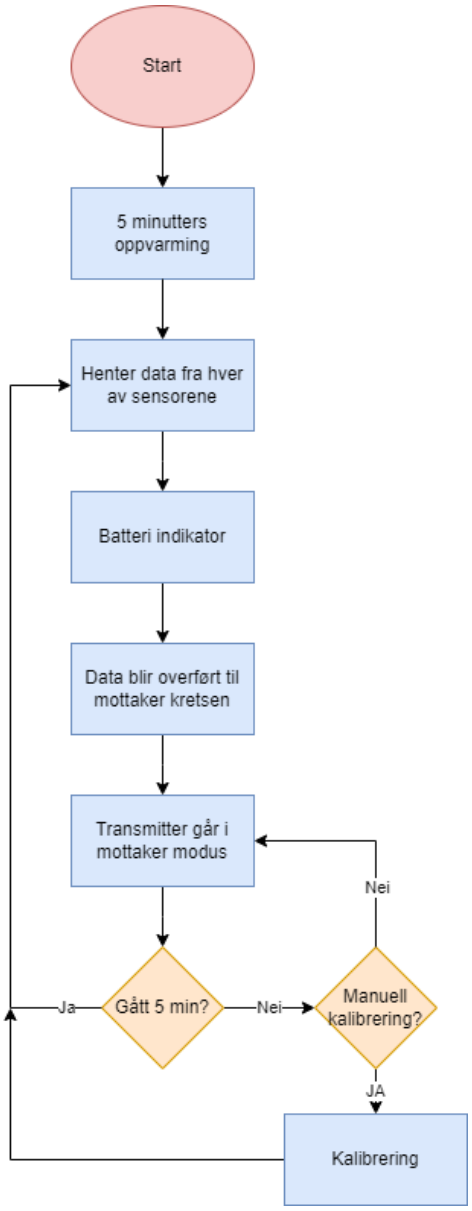
Hver av sensorfunksjoner er bygd opp på lik måte. Alle inneholder en fremkallingskommando som kommer fra hver sitt dedikerte bibliotek. I dette tilfellet blir karbondioksidsensoren SCD30 brukt som et eksempel.

```
void scd30_read() {
  if (scd30.dataReady()) {
    Serial.println("Data Hentet fra!");

    if (!scd30.read()) {
      Serial.println("Error reading sensor data");
      return;
    }
    global_temp = scd30.temperature;
    global_humidity = scd30.relative_humidity;
    global_co2 = scd30.CO2;
  }
```

```
}
```

Det fremkalles riktige farger på RGB-lysdioden ved å kalle på funksjonen `setColour(colourCode)`. `ColourCode`-verdiene er RGB-matriser som defineres på forhånd som konstanter, f.eks. `uint8_t orangeColour[3] = {255, 165, 0};`.

```
void setColour(uint8_t colourCode[3])
{
  R = colourCode[0];
  G = colourCode[1];
  B = colourCode[2];
  ledcWrite(1, R); // send verdien på rød til R-dioden, osv.
  ledcWrite(2, G);
  ledcWrite(3, B);
}
```



Figur 8 Flytdiagram for sensorboksen

3.3.2.2.2. Mottakerenhet

LoRa mottakerenheten består av den samme mikrokontrolleren som avsenderen. Mottaker behandler dataen avsender overfører gjennom LoRa. Strengen med dataen som blir overført til mottakeren blir behandlet ved å separere hver av verdiene mellom ¨|¨ og gi dem et eget felt i en liste (eng. *array*).

```
int i = 0;
    int i2 = 0;
    char message[int(LoRa.Hentet fra())];

    while (LoRa.Hentet fra()) {
      message[i] = LoRa.read();
      i++;
    }

    pch = strtok(message, delim);

    while (pch != NULL) {
      strings[i2] = pch;
      i2++;

      if (i2 > 11) {
        global_gas           = atof(strings[0]);
        global_o2            = atof(strings[1]);
        global_particle_size = atof(strings[2]);
        global_dust_mc2_5    = atof(strings[3]);
        global_dust_nc2_5    = atof(strings[4]);
        global_dust_mc10     = atof(strings[5]);
        global_dust_nc10     = atof(strings[6]);
        global_temp          = atof(strings[7]);
        global_co2           = atof(strings[8]);
```

```
      global_humidity       = atof(strings[9]);
      battery_voltage       = atof(strings[10]);
      signal_strength       = atof(strings[11]);
    }
    pch = strtok(NULL, delim);
  }
```

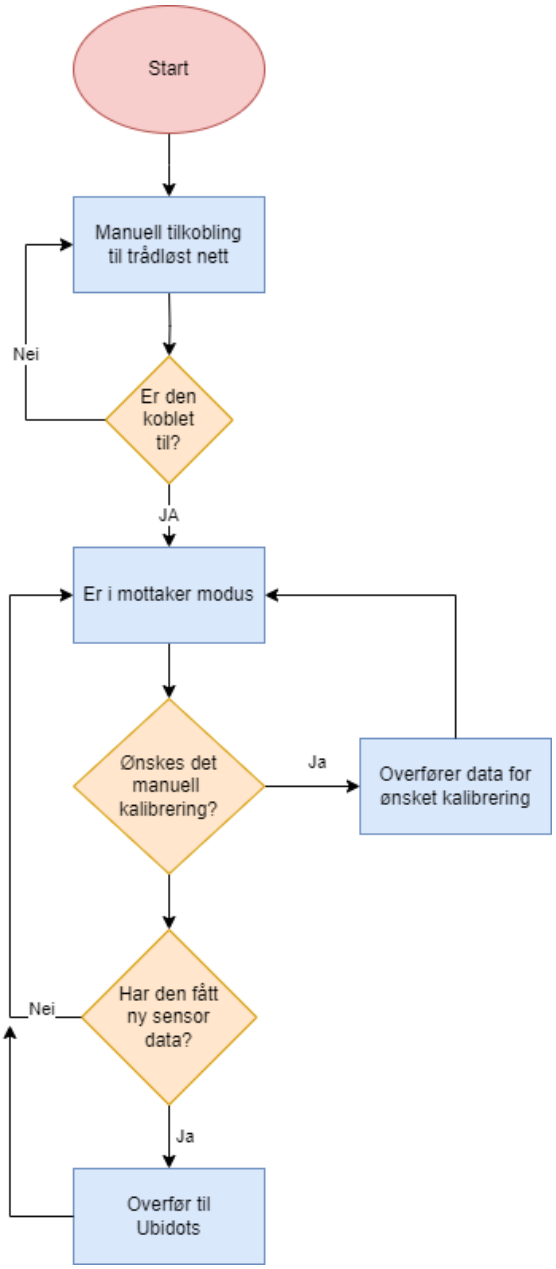
Dette vil resultere i opprettelsen av en liste som gir muligheten til å hente frem hver av målingene. Mottakeren vil deretter overføre verdiene til de angitte variablene på Ubidots ved hjelp av «`ubidots.upload`» funksjonen. Funksjonen utfører kodesnutter som består av `ubidots.add(enhets_navn, verdi);` som setter opp hvor verdiene skal bli sent hen. Og deretter blir `ubidots.publish(enhets_navn)` fremkalt som laster opp dataen til Ubidots.

```
    ubidots.add(dustnc25, global_dust_nc2_5);
    ubidots.publish(DEVICE_LABEL);
```

For å kunne laste opp dataen til Ubidots er det nødvendig at enheten er koblet til det trådløse nettet. For å forenkle oppkoblingen til WiFi blir det brukt WiFi Manager og deres bibliotek. Biblioteket gjør ESP32 til et aksesspunkt. Når man er tilkoblet aksessspunktet er det mulig å konfigurere hvilket trådløst nett enheten skal være tilkoblet. Deretter vil den også være tilkoblet til Ubidots.

```
WiFi.mode(WIFI_STA);

  WiFiManager wm;

  bool res;
  res = wm.autoConnect("LoRaGateway", "E2317Bachelor");

  if (!res) {
    Serial.println("Failed to connect");
  }

  else if (res) {
    //Ubidots#####################################
    ubidots.setCallback(callback);
```

```
  ubidots.setup();
  ubidots.reconnect();
}
```



Figur 9 Flytskjema for mottakerenhet

3.3.3. Skylagringstjeneste

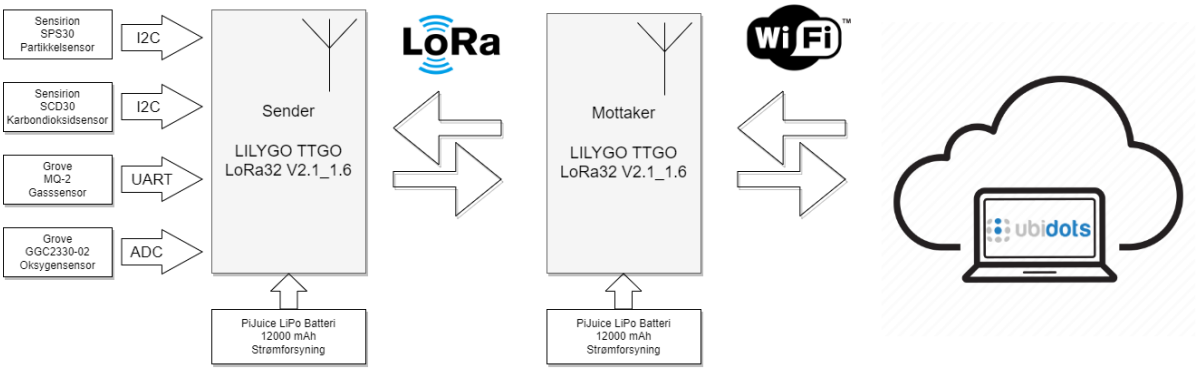
For å implementere Ubidots som en skylagringstjeneste for Lora32 og IoT-sensorer, ble Arduino C-programmeringsspråket brukt. Implementeringen involverte flere trinn for å opprette en effektiv kommunikasjonskanal mellom sensorene og Ubidots-plattformen.

Først ble de nødvendige bibliotekene inkludert i Arduino C-koden til mottakerenheten for å sikre tilkobling og kommunikasjon mellom Lora32-enheten og Ubidots API. Dette omfattet

bruk av LoraWAN-biblioteket for Lora32-funksjonaliteten og biblioteket <UbidotsEsp32Mqtt.h> for Ubidots-integrasjon.
Videre ble LoRa32-enheten initiert og konfigurert med riktige nettverksinnstillinger for å etablere en tilkobling til LoraWAN-nettverket. Dette inkluderte å tilpasse frekvensbånd, datarate og tilkoblingsparametere basert på de spesifikke nettverkskravene.

For å muliggjøre kommunikasjon mellom LoRa32 og Ubidots-plattformen, ble det opprettet en HTTP-tilkobling ved hjelp av en unik API-nøkkel tildelt av Ubidots. Dette etablerte en sikker og autentisert kommunikasjonskanal for dataoverføring. Sensordataene ble deretter lest fra tilkoblede sensorer og lastet opp til skyen via den sikre koblingen.

3.3. Systemarkitektur



Figur 10 Systemarkitektur

3.3.4. Overordnet struktur

Systemet består i grove trekk av tre hoveddeler: en sensorboks, en mottakerenhet og skylagringstjeneste. Sensorboksen rommer fire sensorer og en mikrokontroller med tilkoblet LoRa-antenne. En antenne av samme type er også montert på mottakerenheten. Både sensorboksen og mottakerenheten er tildelt hver sitt batteri. Skylagringstjenesten kan aksesseres fra hvilken som helst datamaskin, nettbrett eller smarttelefon med tilkobling til nettverket.

De individuelle delene av systemet kommuniserer med hverandre på forskjellige måter. Sensorene bruker databusser og periferienheter til å motta kommandoer og sende data til mikrokontroller mens sender- og mottakerenheten kommuniserer med hverandre gjennom

LoRa. Til slutt lastes data opp fra mottakerenheten til Arduino IoT Cloud eller Ubidots gjennom WiFi.

3.3.5. Sensorboksen

Partikkelsensoren SPS30 og karbondioksidsensor SCD30 benytter seriekommunikasjonen I2C. Masterenheten på mikrokontrolleren aksesserer de respektive slavene gjennom deres heksadesimale adresser. Begge sensorene inneholder en intern motstand og kobles til SDA og SCL portene på mikrokontrolleren. Grove Oxygen Sensor Pro bruker UART for å kommunisere med mikrokontrolleren mens målinger fra gassensoren MQ-2 leses av ADC-kretsen.

3.3.6. LoRa-kommunikasjonen

For å oppnå trådløs kommunikasjon mellom avsender og mottaker, brukes det to LoRa-moduler som er integrert i hver av mikrokontrollerne. Modulene kommuniserer med mikroprosessoren ved å bruke SPI-pinnene, hvor hver pinne må defineres i koden. Når alt er konfigurert, kan modulene kommunisere med mikroprosessoren ved hjelp av LoRa.h-biblioteket. Det brukes funksjoner for å trådløst overføre data begge veier.

3.3.7. WiFi-kommunikasjonen

Mottakeren kobler seg til et WiFi-nettverk ved å forsøke å koble til siste tilkoblede nettverk, hvis forsøket mislykkes blir mikrokontrolleren til et aksesspunkt. Når man kobler seg til aksesspunktet med mobiltelefonen eller datamaskinen, kan man konfigurere nettverksinstillingene med enheten. Etter at mottakeren kobler seg til WiFi vil dette koble seg til Ubidots og starte dataopplasting.

3.3.8. Skylagring

Via Ubidots blir dataen også overført trådløst gjennom internett. For å vise dataen på dashbordet må det opprettes variabler for hver av verdiene som skal hentes fra mottakeren. Disse variablene kan deretter visualiseres og presenteres på dashbordet. Visualiseringene inkluderer grafer som viser endringene i innhentede data over tid, og ulike målere for batterispenning, temperatur, signalstyrke osv.

3.4. Kalibrering av sensor

3.4.1. Karbondioksid- og fuktighetssensor SCD30

Karbodioksid- og fuktighetssensor SCD30 har en selvkalibreringsfunksjon definert i biblioteket `<Adafruit_SCD30.h>` som kan brukes hvis ønskelig. Imidlertid er det ikke

optimalt å bruke den i dette tilfelle. Selvkalibreringen krever at sensoren er plassert utendørs i frisk luft og gjør syv målinger som er separert med atten timer. Dette anses som unødvendig overkomplisering av bruksmåte i dette tilfellet og vil derfor utelates. Derfor er det bestemt at det skal benyttes av manuell kalibrering på sensoren. I biblioteket er det definert en funksjon som tillater en tvungenkalibrering som gir sensoren en bestemt ppm-verdi (eng. parts per million). Sammenligningen av CO_2-nivåene skal bli sammenlignet med mengden som er i frisk luft, sensoren vil derfor bli kalibrert til å sammenligne med 400 ppm CO_2.

3.4.2. MQ-2 gassensor

Kalibreringen av MQ-2 sensoren skjer automatisk. Inkludert i programmeringen er det satt opp en oppvarmingstidsrom under oppstart. Oppvarmingen varer i en periode som ligger på fem minutter. Etter oppvarming er sensoren klar for å måle data i omgivelsene som kan bli lesbart. Dersom det er en periode der målingene fra sensoren er unormalt høye, vil det være sensoren som sender ut for høy spenning, og det kan være nødvendig å justere potensiometeret manuelt. Potensiometeret brukes til å justere sensorens følsomhet og kan hjelpe med å korrigere for unormalt høye målinger.

3.5. Innkapsling

Dette kapittelet omtaler metoden brukt for å komme fram til innkapslingen. Det ble planlagt to modeller, en tilpasset LoRa32 og sensorene, vist i Figur 11 og en annen boks laget for Raspberry Pi 3, presentert i Figur 12. Siden RPI til slutt ikke ble tatt i bruk, ble justeringer gjort for at den kan romme en LoRa32.

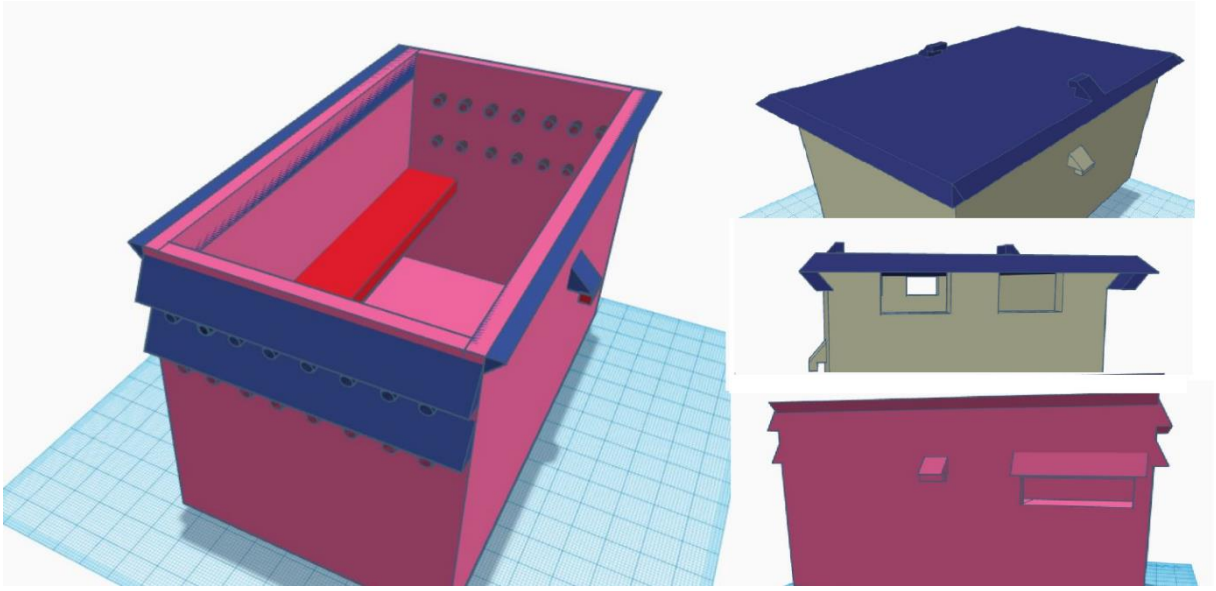
3.5.1. Design

For å romme de ulike sensorene og LoRaWAN mikrokontroller, ble det utviklet en spesialdesignet 3D-boks. Designprosessen startet med å ta mål av den største delen av boksen, som var batteriet. Disse målene ble brukt til å definere dimensjonene til boksens indre flate.

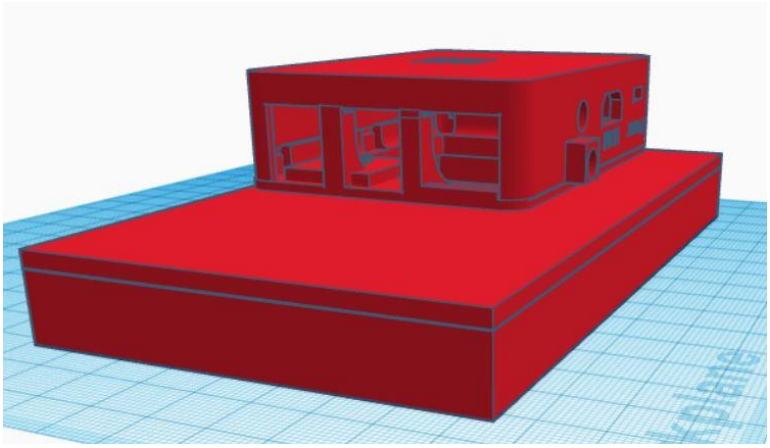
Ved å bruke Tinkercad sin 3D-designplattform kunne ulike former og dimensjoner enkelt legges til for å oppnå ønsket design. Boksen ble konstruert ved hjelp av fem firkanter, som ble brukt som gulv og fire vegger. Den største av disse flatene representerer batteriet, og det ble lagt til en ekstra 3 mm klaring på flere kanter for å sikre tilstrekkelig plass. Dette var viktig for å unngå kontakt mellom batteriet og andre komponenter og for å muliggjøre enkel montering og demontering.

Boksen ble konstruert for å være solid og beskyttende, samtidig som den tillot tilstrekkelig ventilasjon for komponentene inne i boksen. For batteriet ble det gjort en spesiell justering i designet. Batteriet ble plassert litt løftet fra grunnflaten i boksen. Dette tillater bedre luftsirkulasjon rundt batteriet og bidrar til å avkjøle det under drift. Ved å sørge for tilstrekkelig luftstrøm bidrar denne tilpasningen til å opprettholde en gunstig driftstemperatur for batteriet, og dermed forbedre den generelle ytelsen og levetiden.

I tillegg til de nevnte funksjonene har boksen også blitt utstyrt med fire bein. Disse beina er designet for å løfte boksen litt opp fra bakken, noe som gir en ekstra beskyttelse mot fuktighet, vann og gjørme som kan forekomme inne i en tunnel. Ved å heve boksen bidrar beina til å redusere risikoen for at vann eller skitt trenger inn i boksen og påvirker komponentene. Dette ekstra laget av beskyttelse sikrer at sensorene og mikrokontrolleren forblir trygge og fungerer optimalt, selv under utfordrende forhold.



Figur 11 Tinkecad modellering av sensorboks



Figur 12 Lora mottaker designet for Raspberry Pi 3

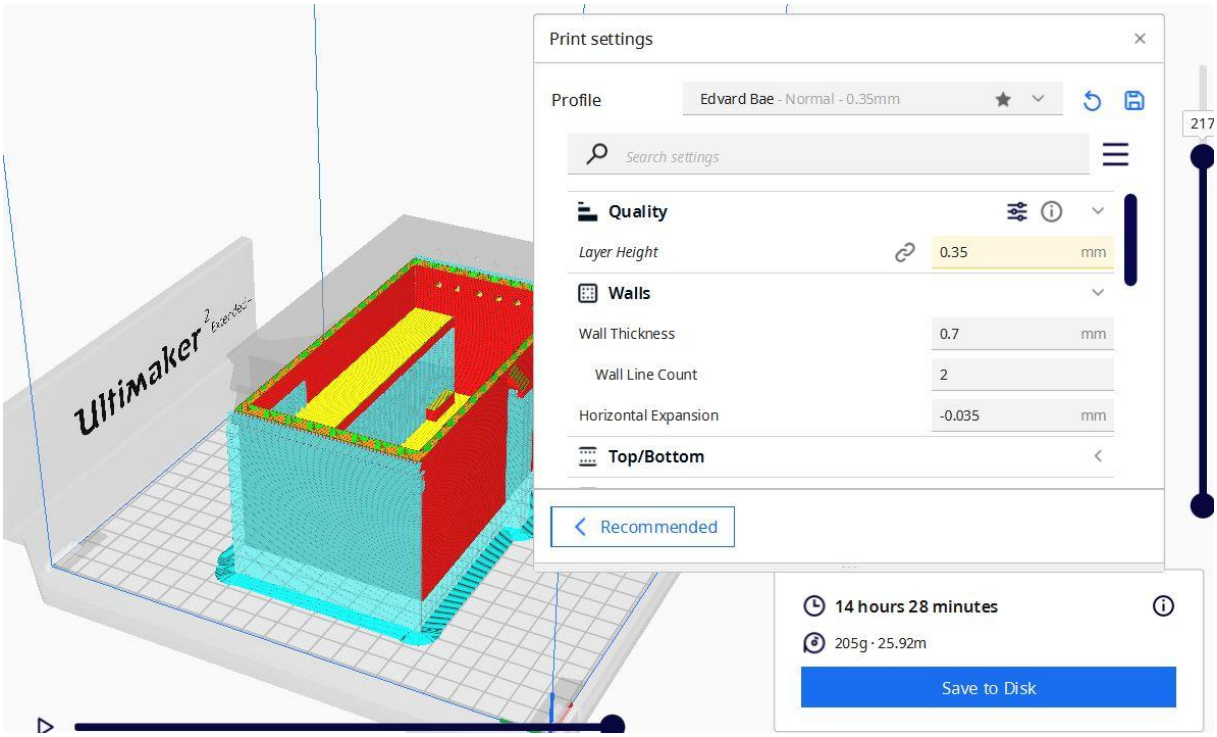
3.5.2. Innstillinger

For å oppnå en rask utskrift av den komplekse boksen i Tinkercad, ble spesifikke innstillinger brukt i Cura-slicerprogramvaren. Vi tok i bruk MakeNtnu sitt verksted som låner ut printermodellen Ultimaker 2+. Figur 14 Et av kravene deres er at en utskrift ikke skal overskride 16 timer. Med bruk av standard innstillinger ble printetid estimert på omtrent 20 timer. Derfor ble det fokus på å øke utskriftshastigheten ved hjelp av følgende innstillinger: Figur 13

Generelt sett kan lagtykkelser variere fra 0,05 mm til 0,3 mm for vanlige desktop FDM (Fused Deposition Modeling)-skrivere. Mindre lagtykkelser, som 0,1 mm eller 0,15 mm, kan gi høyere detaljnivå og bedre overflatekvalitet, men det vil ta lengre tid å fullføre utskriften. Større lagtykkelser, for eksempel 0,2 mm eller 0,3 mm, kan gi raskere utskrifter, men med mindre detaljer og mer synlige laglinjer. En lagtykkelse på 0,35 ble valgt for å øke hastigheten på utskriften.

For å akselerere utskriftsprosessen ytterligere ble en relativt tykk skalltykkelse valgt. Dette resulterte i færre lag som måtte printes for å fullføre boksens vegger. En skalltykkelse på 0,7 mm ble brukt.

Et raskt og enkelt fyllmønster som grid ble valgt for å spare tid samtidig som tilstrekkelig styrke ble opprettholdt. Utskriftshastigheten ble justert til 60 mm/s for å oppnå en raskere utskrift.

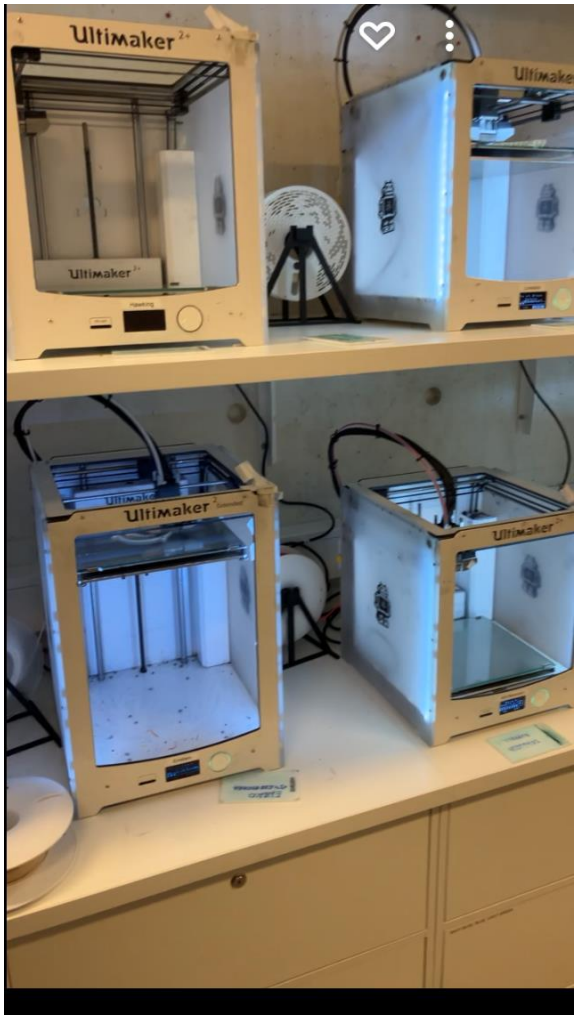


Figur 13 Innstillinger brukt i Cura

3.5.3. Utskrift

Prosessen av selve utskriften gikk ut på å overføre en lokal g.code fil en tilgjengelig 3D-printer. Overføringen skjer med et sd-kort (minnekort) som printeren kan lese. Når minnekortet er satt inn, blir filen tilgjengelig på en liten meny. Når riktig fil er valgt starter en oppvarmingsprosess. Deretter lager printeren en grunnflate som den byggere videre på, lag for lag. Når grunnflaten er laget uten uhell kan man ganske sikkert si at resten av printen blir gjennomført. Akkurat den printeren som ble brukt hadde kamera overvåkning slik at en kan se

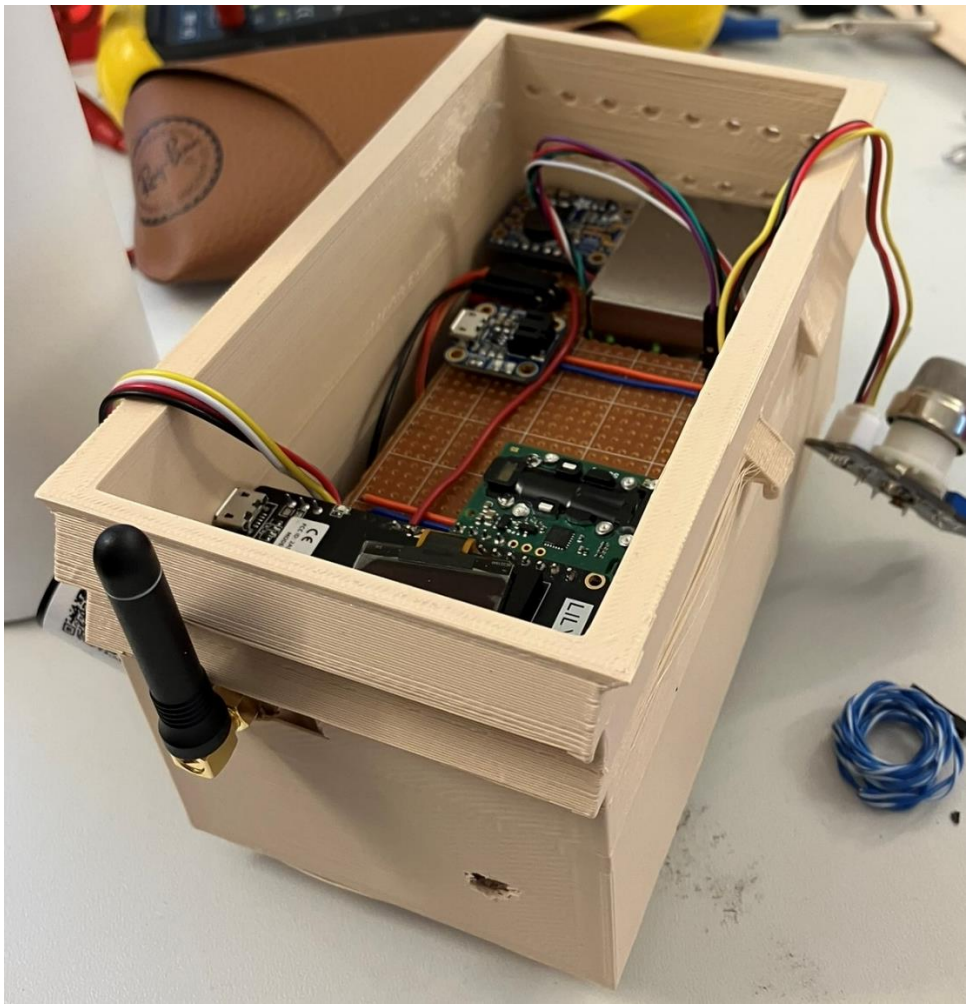
tilstanden til objektet uten å være der. Med tanke på at utskriften var såpass omfattende, ble to utskrifts prosesser startet likt for å øke sjansen for et brukbart sluttprodukt.



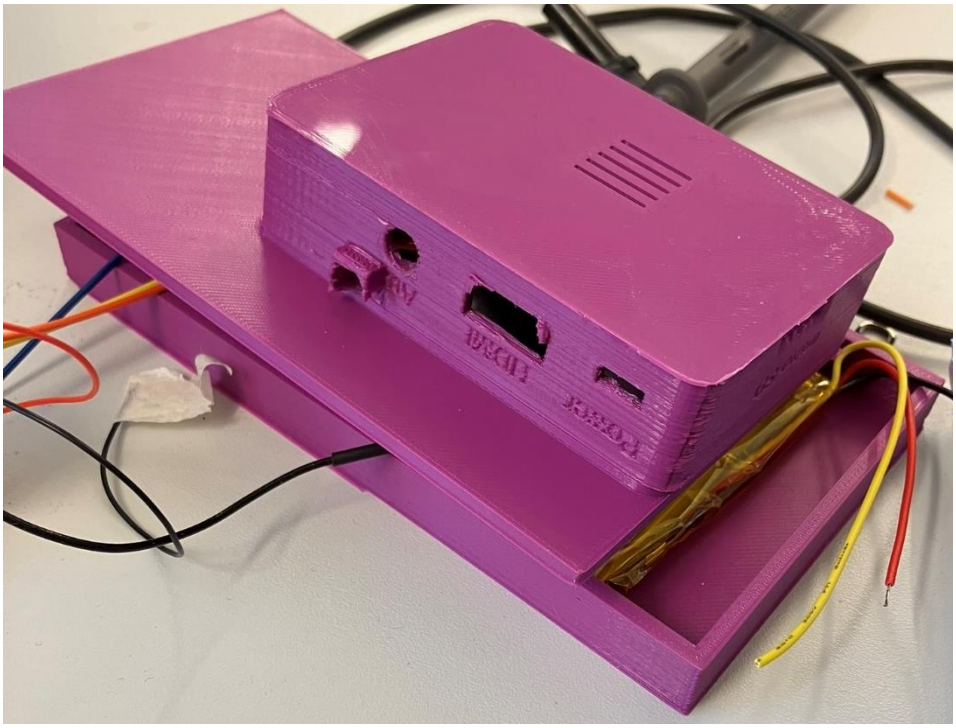
Figur 14 3D-printer på MakeNTNU sitt lokale

3.5.4. Montering

Før sensor, batteri og mikrokontroller kunne bli montert på boksen måtte støttestrukturen fjernes. Noen små justeringer av boksen ble gjort for å tilpasse lokket til boksen. En fil ble brukt til å slipe ned noen kanter slik at lokket skulle skli lettere på.



Figur 15 3D-printet sensorboks



Figur 16 LoRa mottaker under montering, boksen var opprinnelig laget for å romme Raspberry Pi 3 Modell B+

3.6. Testing av systemet

3.6.1. Testing av sensorfunksjonalitet

1) Støvsensor SPS30

Partikkelsensoren SPS30 ble testet i ulike tilstander. Sensoren ble testet ute for å hente fram data fra mer normale omgivelser for å få data på hvordan forholdene generelt burde være. Dataene som blir tatt opp vil så bli sammenlignet med annen data som er tatt under mer normale omstendigheter.

2) SCD30:

SCD30 blir testet på to ulike områder, på kontoret og langs en travel gate. Etter som at det forventes at bilene som forbikjører skal øke CO2 nivåene i nærområdet. Dataen som blir tatt opp fra kontoret blir sett på som dataen som kommer fra mer normale omgivelser for å få data som kan sammenlignes med det som blir målt langs veien.

3) Kommunikasjon:

For å teste den trådløse kommunikasjonen mellom sensorboksen og mottakeren ble det utført en rekke forskjellige tester. Der hver at testene var basert på signalstyrke og distanse.

Den første testen var basert på å finne den lengste distansen sensorboksen har muligheten til å sende data. Testen ble derfor gjort i et område med klart og direkte syn mot mottakeren. Der vi mottok RSI styrken og målte distansen mellom enhetene. Målingene ble gjort helt til dataen som ankommer mottakeren ble korrupt.

Det ble også utført andre tester der hoved fokuset var å finne ut av hvordan LoRa oppfører seg når signalene må gå igjennom hindringer. Hindringene som ble satt i fokus er vegger og bygg. Derfor ble det gjort tester i områder der det er bygg med mange tykke vegger.

3.6.2. Testing av systemet i et tunnelanlegg

Systemet ble testet i et tunnelanlegg på Sollihøgda ved Oslo der oppdragsgiveren Skanska Norge AS bygger motorveien E16 mellom Bjørum og Skaret i Viken-fylket. [42] Før besøket ble det formulert følgende testplan:

1) Før ankomst i selve tunnelen må utstyret tas ut fra den beskyttende innpakningen. Alle sensorene kobles på nytt til mikrokontrolleren, antennen monteres på. Det utføres tester for å se om alt er riktig oppkoblet og om LoRa-mottakeren detekterer signalet fra avsenderen. LoRa-mottaker kobles opp mot WiFi på anlegget med brukernavnet og passordet oppgitt av personellet på anlegget. Brukergrensesnittet kontrolleres for å se om data vises korrekt. Etter ovennevnte testene kan prosjektgruppen kjøre inn i tunnelen sammen med kontaktpersonen fra personellet på anlegget.
2) Tester av selve sensorboksen:

- Sensorboksen plasseres på steder der den kan utsettes for mekaniske skader under salver/eksplosjoner og massetransport.
- Det forsøkes å plassere sensorboksen på forskjellige steder i anlegget for å finne den optimale beliggenheten for parameterovervåkning og sikkerhet.
- Det benyttes manuelle måleapparater for å kunne bruke de avleste tallene som referanseverdier for sensorsystemet og på denne måten kontrollere at data som hentes inn av sensorene er troverdige.
- Det utføres en kvalitativ kontroll av sensorenes målinger ved å plassere dem på steder med ulike forventningsverdier (f.eks. et sted med mye støv og et annet uten støv).

3) Tester av LoRa-kommunikasjon:

- Avsender og mottaker plasseres i betydelig avstand fra hverandre. Avstanden økes gradvis for å måle den maksimale rekkevidden for uforstyrret dataoverføring.

- Kommunikasjonen kontrolleres for ytelse med hindringer i sikt (rundt hjørner, bak maskiner osv.) .
- Målt data leses av direkte fra sensorer på en datamaskin i Seriell Monitor på Arduino IDE parallelt med avlesning av data i skyen for å kontrollere om alt data blir overført eller kun deler av den.

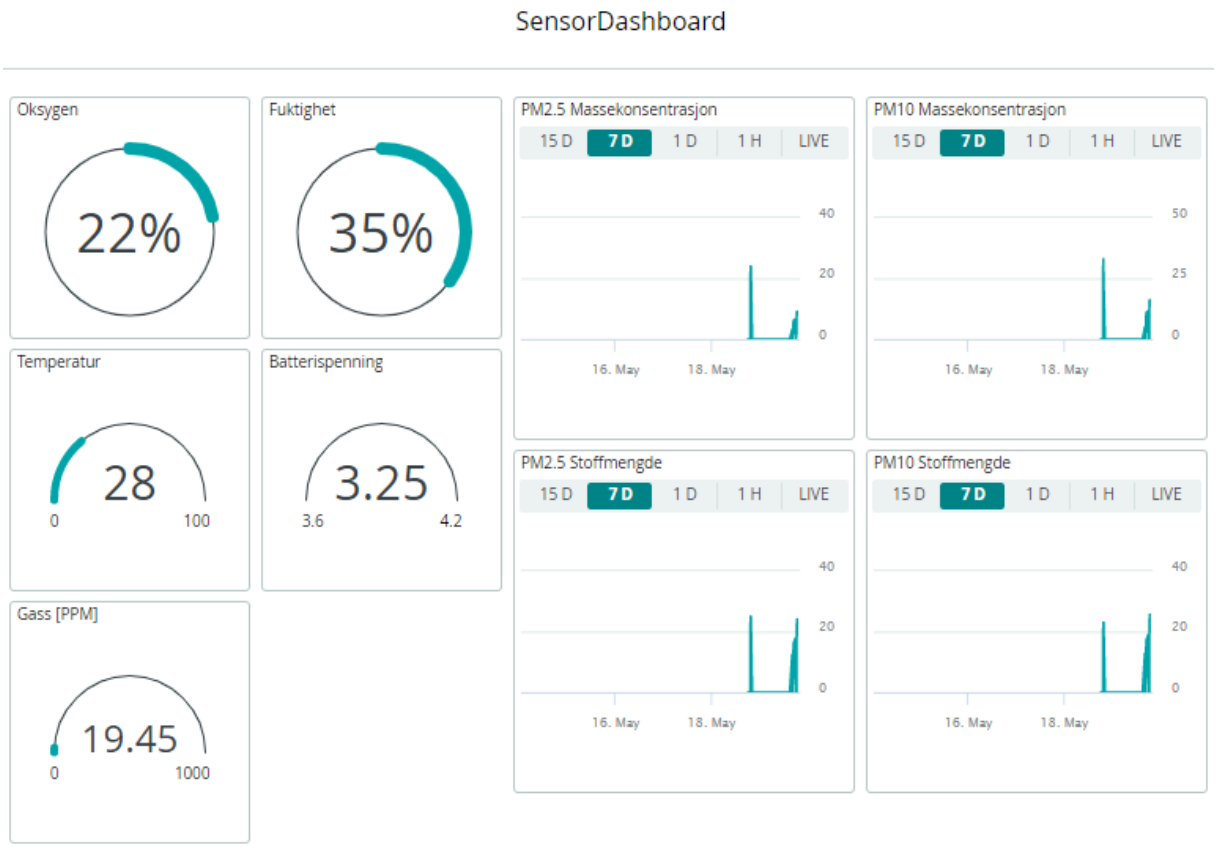
4) Dataopplasting til skylagringstjenesten:

- Det undersøkes systemets kapasitet ved å forkorte samplingintervaler og forårsake overbelastning av data.
- LoRa-mottakeren plasseres i lang avstand fra WiFi-antennen for å sjekke ytelsen til dataopplasting ved svak nettilkobling.

5) Varslingssystem ved feil:

- Det utløses ulike hendelser for å sjekke hvor lang tid det tar før varslingssystemet aktiveres.
- Det kontrolleres om det blir aktivert riktig respons på de ulike hendelsene.
- Det undersøkes og diskuteres følgende problemstilling: Er visuell varsling tilstrekkelig? - Varsel blir bl.a. videresendt til skylagringen, men hva skjer når sensorboksen ikke har tilkobling til nettet?

4. Resultater



Figur 17 Skjermbilde fra Arduino IoT Cloud

4.1. Funksjonalitetstest av sensorer

4.1.1. Introduksjon:

I denne delen presenteres resultatene fra funksjonalitetstesten av sensorene i et tunnelanlegg, som ble gjennomført over to testdager. Formålet med testen var å evaluere ytelsen og påliteligheten til sensorboksen med fokus på parameterovervåkning i henhold til beskrivelsen gitt i 3.6.2 Testing av systemet i et tunnelanlegg. Testingen ble gjennomført i samarbeid med Skanska Norge AS, som er ansvarlig for byggingen av motorveien E16 mellom Bjørum og Skaret i Viken-fylket.

På den første testdagen oppstod det utfordringer med en av mikrokontrollerne i systemet, trolig som følge av skader under transporten. Som en del av skadekontrollen ble batteriet skilt ut fra systemet, og hver enkelt sensor ble undersøkt for å vurdere funksjonaliteten. Resultatet

av denne gjennomgangen var at sensorene fremdeles fungerte, men en av mikrokontrollerne ble identifisert som defekt. På grunn av dette ble ingen av testene gjennomført denne dagen.

På dag to ble LoRa-delen av systemet utelatt da den defekte mikrokontrolleren ikke kunne erstattes på så kort tid. Kun sensor-enheten ble brukt med direkte WiFi-kobling. Sensorboksen ble plassert 600 meter fra stuff der den sprengning ville skje. Sensorboksen forble i tunnelen både under sprengningen og i 30 minutter etter sprengningen. Resultatene under er fra partikkelsensoren som var den eneste sensoren som registrerte brukbar data fra tunellbesøket.

Figur 18 Resultat fra støvsensor NC2.5, det er en type sensor som brukes til å måle konsentrasjonen av partikler i luften, spesielt partikler med en diameter på 2,5 mikrometer eller mindre

Figur 19 Resultat fra støvsensor NC10, det er en type sensor som brukes til å måle konsentrasjonen av partikler i luften, spesielt partikler med en diameter på 10 mikrometer eller mindre

4.1.2. Gjennomførte tester i relevant miljø:

I den første testen ble Ubidots-forbindelsen evaluert utenfor tunnelen, og denne testen ble gjennomført uten problemer. Det ble oppnådd en stabil og pålitelig forbindelse mellom enheten og Ubidots-plattformen utenfor tunnelen.

Videre ble det gjennomført en test av partikkelsensoren under en sprengning, og denne testen var også vellykket. Sensoren var i stand til å pålitelig registrere støvnivåene under sprengningen. Direkte målingsdata fra partikkelsensoren vises i Figur 18 og Figur 19. Disse dataene gir en tydelig demonstrasjon av sensorens evne til å registrere støvnivåer i sanntid.

Det er verdt å merke seg at både sensordataene og disse dataene ble lastet opp til skyen (Ubidots-plattformen) i sanntid under testen. Salven ved stuff gikk av klokken 14:30 og ut ifra Figur 18 Figur 19 øker konsentrasjonen til sin høyeste verdi klokken 14:48 for deretter å synke til tilsvarende verdi som var registrert før sprenging.

4.1.3. Ufullførte tester og begrensninger:

Flere av de planlagte testene kunne ikke gjennomføres som planlagt i forkant av tunnelbesøket. Spesifikt var følgende sensorer ikke operative under testen i tunellmiljø: gass-sensoren, oksygensensoren, CO_2-sensoren, temperatursensoren og fuktighetssensoren. På et senere tidspunkt ble det imidlertid etablert forbindelse med disse sensorene og utført vellykkede tester i et annet miljø, utendørs under åpen himmel.

I tillegg ble varslingssystemet ved feil ikke testet under selve tunellbesøket som planlagt. De tiltenkte testene skulle involvere utløsning av ulike hendelser for å vurdere aktiveringstiden for varslingssystemet og kontrollere om riktig respons ble aktivert for hver hendelse. Imidlertid ble varslingssystemet testet på et senere tidspunkt under de samme forholdene som nevnt tidligere, og testen var vellykket.

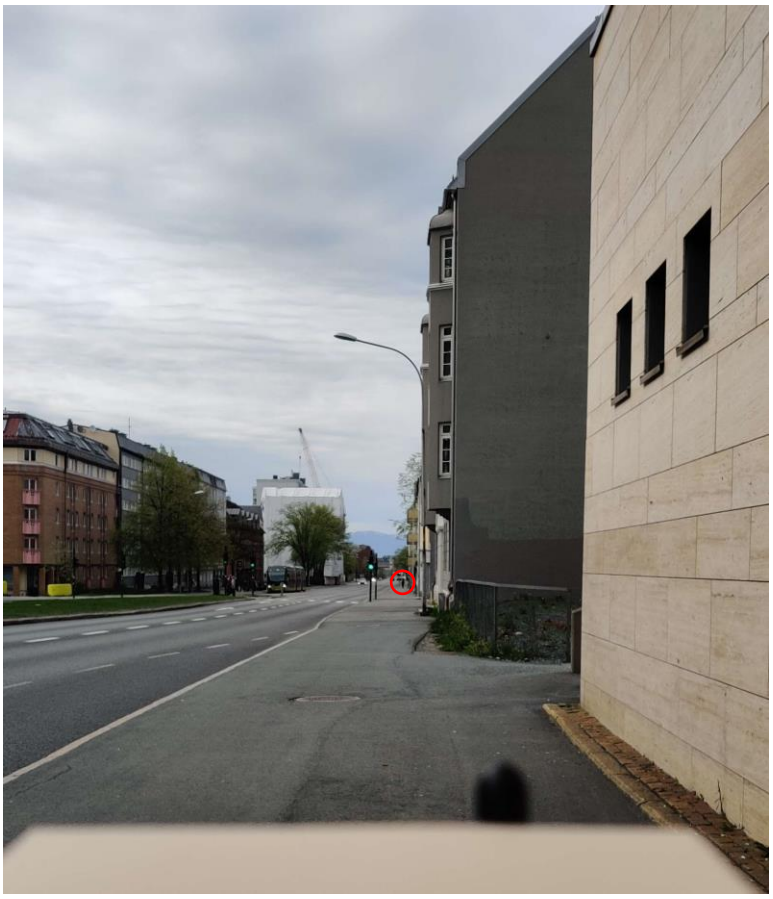
4.2. Kommunikasjon mellom LoRa-enhetene

4.2.1. Introduksjon:

For å etterligne testforholdene i tunnelanlegget der LoRa rekkevidden skulle opprinnelig testes, ble det valgt en langstrakt vei med høye bygninger på hver side som testmiljø. Både fotgjengere, biler og trær fungerte som obstruksjonene som er ofte til stede i tunnelanlegg. Den valgte veien strekker seg to kilometer i rett linje, og er godt egnet for å teste den potensielle rekkevidden.

4.2.2. Gjennomførte tester:

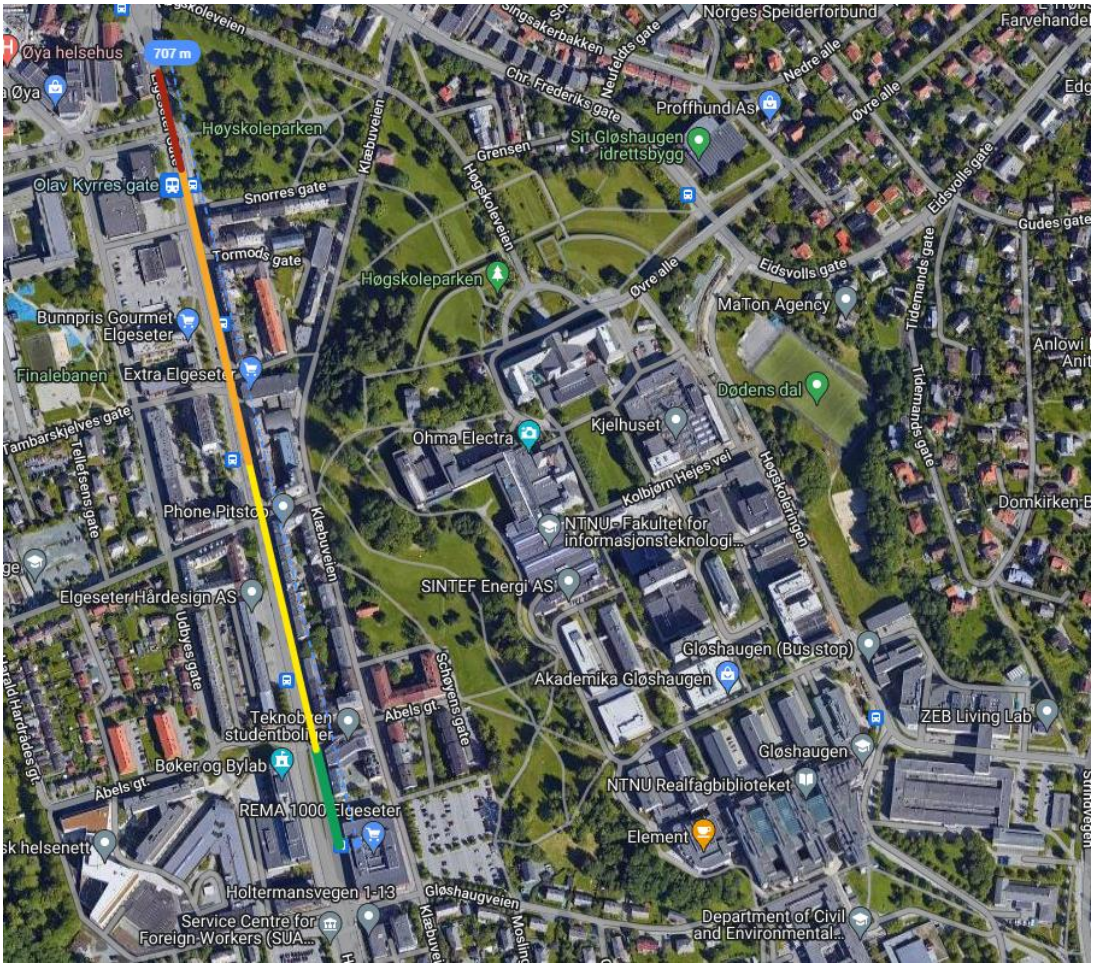
Det ble gjort to typer tester av LoRa-kommunikasjonen, hvor den første testen undersøkte den maksimale avstanden sensorboksen kunne kommunisere med mottakeren. Sensorboksen ble plassert én meter over bakken og sto stasjonært ved en bussholdeplass, mens mottakeren ble båret bort langs veien til signalstyrken ble betydelig svekket og opplastningen av sensordata stanset. Antennene ble også vendt mot hverandre for å hindre signaltap årsaket av kroppen og sensorboksen. Oppdatering av sensordata stoppet etter 700 m avstand mellom avsender og mottaker.



Figur 20. Viser sensorboksen med antenna rettet mot mottakeren (rød sirkel) ca. 200 m unna.

På veien tilbake til avsenderen ble avstanden notert hver gang signalstyrken økte med 10. Ved den maksimale rekkevidden lå signalstyrken på -105 dBm, noe som er 15 dBm unna LoRas minimum RSSI verdi på -120 dBm. Dette droppet ned til -90 dBm ved 620 m, altså 90 m nærmere med relativt stabil datakommunikasjon. Ved 350 m lå signalstyrken på -80 dBm og

regnes ennå som svakt signal, men det ble ikke opplevd datakorrupsjon. -70 dBm regnes som tilfredsstillende sterkt signal, og oppsto 90 m unna avsenderen.



Figur 21. Strekningen for rekkeviddetesten.

Grønn: <-70dBm, 90m. Gul: <-80dBm, 260m. Oransje: <-90dBm, 270m. Rød: <-105dBm, 90m

4.2.3. Ufullførte tester og begrensninger:

Ideelt sett hadde rekkeviddetesten blitt gjennomført i et tunnelanlegg, men i stedet ble det valgt en rett strekning med høye bygninger på hver side som emulerte tunnelveggene. Hindringer som fører til signaltap var til stede i form av fotgjengere, biler og trær. Dette etterligner obstruksjonene i tunnelen som ofte er i form av kjørende lastebiler, stillestående steiner og ansatte. Det er derimot umulig å vite prestasjonsforskjellen av systemet mellom tunnelanlegg og resultatene som ble oppnådd under det improviserte testmiljøet, uten å teste systemet i tunnel.

4.3. 3D-utskrift

Resultatene av 3D-utskriften viser at boksen ble generelt vellykket, men det ble observert noen ujevnheter i lagstrukturen på den ene veggen og undersiden.

5. Diskusjon

5.1. Sensorboksen

Selv om de ufullførte testene og begrensningene i testmiljøet har påvirket studiens omfang, er det viktig å anerkjenne de oppnådde resultatene og deres implikasjoner. Gjennomføringen av vellykkede tester og demonstrasjonen av funksjonalitet og pålitelighet i visse aspekter av systemet gir en indikasjon på potensialet for videreutvikling og forbedring av sensorboksen.

I fremtidig arbeid kan det være hensiktsmessig å fokusere på fullføring av de ufullførte testene, spesielt med tanke på de ikke-operasjonelle sensorene og testing av varslingssystemet ved feil. Dette vil bidra til en mer omfattende evaluering av sensorenes funksjonalitet og pålitelighet samt effektiviteten av varslingssystemet.

Videre undersøkelser kan også adressere andre relevante faktorer, for eksempel sammenligning av sensorenes nøyaktighet og pålitelighet i ulike miljøer og evaluering av forskjellige alternativer for varslingssystemet ved feil. Dette vil bidra til en bedre forståelse av sensorboksens ytelse og potensiale for fremtidig implementering i reelle miljøer.

5.2. LoRa-kommunikasjonen

Det er bemerkelsesverdig at LoRa-teknologien demonstrerte en betydelig rekkevidde, som spesielt kan være gunstig i bruksområder som tunnelanlegg og andre steder med behov for langdistansekommunikasjon. Ved å implementere mer retningsbestemte antenner og nøye planlegge deres posisjonering, ville signalkvaliteten ha blitt forbedret, og rekkevidden ville dermed ha blitt ytterligere utvidet. Dette ville åpnet for bruk av systemet i lengre avstander.

For å ytterligere evaluere og optimalisere ytelsen til LoRa-kommunikasjonssystemet, er det hensiktsmessig å gjennomføre flere tester og analyser under ulike miljøforhold og interferensscenarier. Dette vil gi økt forståelse av systemets grenser og identifisere potensielle forbedringsområder for å oppnå pålitelig kommunikasjon over enda lengre avstander. Ved å utforske disse aspektene vil man kunne identifisere optimale løsninger og teknikker for å oppnå økt rekkevidde og pålitelighet i ulike scenarier.

5.3. 3D-utskrift

Disse ujevnhetene nevnt i 4.3. 3D-utskrift kan tilskrives økningen i utskriftshastigheten, samt endringene i veggtykkelse og lagtykkelse. Økningen i utskriftshastigheten kan ha påvirket kvaliteten ved å redusere utskriftstiden, men det kan også ha ført til ujevnheter da lagene ikke fikk tilstrekkelig tid til å herde skikkelig. Endringene i veggtykkelse og lagtykkelse kan også ha spilt en rolle i ujevnhetene. Det er viktig å finne en balanse mellom utskriftshastighet, veggtykkelse og lagtykkelse for å oppnå ønsket kvalitet. For fremtidige utskrifter anbefales det å eksperimentere med ulike kombinasjoner og justere parameterne gradvis for å redusere ujevnheter. Til tross for ujevnhetene, oppfylte boksen formålet og var funksjonell, men det anbefales å vurdere justeringer av utskriftsparameterne for å forbedre kvaliteten.

5.4. Forbedringspotensiale og videre arbeid

Dette kapittelet beskriver viktig lærdom som tas med videre og forhåpentligvis kan hjelpe fremtidige bachelorkandidater og andre lesere å unngå lignende feil.

Det anbefales å bruke en annen strategi for komponentinnkjøp enn den som ble valgt av prosjektgruppen i denne oppgaven. Det er viktig å merke seg at komponentene kan bli utsatt for utellelige typer skader, spesielt når man jobber med lodding eller når man frakter komponentene til og fra arbeidsstedet. Det oppfordres på det sterkeste å skaffe seg flere sett med komponenter slik at man lett kan erstatte den mangelfulle komponenten med en annen utgave av den. Dessuten anbefales det å anskaffe komponenter fra ulike leverandører for å ha en alternativ handlingsplan i tilfelle en av bestillingene blir betydelig forsinket.

Videre er det ikke alltid åpenbart hvilke sensorer man skal bruke basert kun på databladet. Det fornuftige valget er da å kjøpe inn flere ulike typer sensorer for så å teste hver enkel av dem og avgjøre hvilken som egner seg best til det konkrete formålet. Det er flere egenskaper som kan spille en rolle i dette valget og man bør formulere disse kravene tidlig i prosjektarbeidet. I denne oppgaven har de fleste implementerte sensorer vist en tilfredsstillende ytelse i klasserommiljø, unntatt gassensor som ikke oppfyller kravene stilt til den da den ikke skiller mellom de detekterte gassene. Siden testene i tunnelmiljø ikke ble tilfredsstillende, er det uklart om sensorene presterer like godt i utfordrende forhold. Det antas at sensorene bør vurderes utskiftet med komponenter som er laget for industrielle formål.

På grunn av dette må resultatet av prosjektet betraktes som et såkalt konseptbevis (eng. *proof of concept*) og ikke et fullstendig produkt som kan implementeres umiddelbart.

Det burde ha blitt implementert en fysisk knapp for selvkalibrering som man kan trykke på for å aktivere rensing og selvkalibrering. I tillegg er batteriløsningen mangelfull og utnytter ikke potensiale for strømsparing som LoRa tilbyr. Batteritilstanden bør kunne overvåkes av systemets brukere. I dette IoT-systemet er det ikke behov for at sensorboksen og mottakerenheten skal være slått på konstant. Det bør vurderes å definere funksjoner i koden som sender mikrokontrollere i sovemodus når de ikke er i bruk.

6. Konklusjon

Målet for bachelorprosjektet var å implementere parameterovervåkning i tunnelanlegg ved bruk av LoRaWAN-basert IoT-system, og spesifikt bygge en selvkalibrerende sensorboks for overvåkning av støv-, oksygen-, gass- og karbondioksidnivå, samt et integrert varslingssystem, oppnå automatisk dataoverføring ved kort avstand fra sensorboksen til flyttbar serverenhet ved hjelp av LoRa og sikre dataopplasting til en skylagringstjeneste. De viktigste funnene fra prosjektet er som følger:

Funksjonalitetstesten av sensorene avslørte begrensninger og utfordringer, inkludert feil på mikrokontrollere og operative begrensninger. Partikkelsensoren viste pålitelig ytelse i deteksjon av støvnivåer under sprengning. Imidlertid kunne flere planlagte tester med gassensor, oksygensensor, CO_2-sensor, temperatursensor og fuktighetssensor ikke gjennomføres i tunnelmiljøet. Videre testing og forbedring av sensorene under mer realistiske forhold anbefales for å evaluere deres funksjonalitet og pålitelighet.

Testene på LoRa-kommunikasjon viste en betydelig rekkevidde på opptil 710 meter under testforholdene. Det er imidlertid viktig å merke seg at ulike faktorer kan påvirke signalstyrken og rekkevidden. Videre testing og analyse under ulike miljøbetingelser og interferensscenarier anbefales for å forstå systemets begrensninger og optimalisere ytelsen.

Resultatene av 3D-utskrift viste generelt sett en vellykket sensorboks, selv om det ble observert ujevnheter i lagstrukturen på en vegg og undersiden. Justering av utskriftsparametere anbefales for å forbedre kvaliteten og redusere uregelmessigheter.

Konklusjonen er at resultatene fra prosjektet gir en god start for videreutvikling og forbedring av sensorboksen. Det er viktig å adressere de ufullførte testene og begrensningene i prosjektet for å evaluere sensorenes funksjonalitet og pålitelighet i mer realistiske miljøer. Videre optimalisering av LoRa-kommunikasjonen og justering av utskriftsparametere kan forbedre kvaliteten på sensorboksen. Prosjektet gir verdifull innsikt og lærdom for fremtidige prosjekter innen parameterovervåkning og IoT-applikasjoner.

7. Bibliografi

[1] M. O. Francis Griffiths, «The fourth industrial revolution - Industry 4.0 and IoT [Trends in Future I&M,» *IEEE Instrumentation & Measurement Magazine*, vol. 21, nr. 6, pp. 29-43, 2018.

[2] D. J. E. M. R. H. Arka Ghosh, «Patterns and Trends in Internet of Things (IoT Research: future applications in the construction industry,» *Engineering, Construction and Architectural Management*, vol. 28, nr. 2, pp. 457-481, 2021.

[3] S. Gerard, «How Can LoRaWAN Benefit the Tunnel Monitoring During Construction,» Moko LoRa, 20 oktober 2022. [Internett]. Hentet fra: https://www.mokolora.com/lorawan-tunnel-monitoring/. [Funnet 4 mai 2023].

[4] J. K. R. Ankush Rai, «Internet of things-based multisensor non-invasive technology for robust monitoring of tunneling infrastructure.,» *Asian Journal of Pharmaceutical and Clinical Research*, vol. 10, p. 312, 2017.

[5] P. Nafstad, «Helseeffekter av utendørs luftforurensning,» *Tidsskriftet Den Norske Legeforening*, p. 4, 18 November 2004.

[6] J. Sørlie, «Tromsøysundet tunnel. Vurdering av luftforurensning.,» NILU, Tromsø, 1990.

[7] LoRa Alliance, «About LoRaWAN,» LoRa Alliance, 2022. [Internett]. Hentet fra: https://lora-alliance.org/about-lorawan/. [Funnet 15 Mai 2023].

[8] Ultimaker, «Ultimaker,» Ultimaker B.V., 9 Mai 2023. [Internett]. Hentet fra: https://ultimaker.com/. [Funnet 12 Mai 2023].

[9] 3D Printing Industry, «3D Printing Industry,» 3D Printing Industry, 10 Mai 2023. [Internett]. Hentet fra: https://3dprintingindustry.com/. [Funnet 10 Mai 2023].

[10] Nanotron Technologies GmbH, «Chirp Spread Spectrum (CSS),» Nanotron Technology, [Internett]. Hentet fra: https://nanotron.com/EN/CO_techn-css-php/. [Funnet 2023 mai 7].

[11] B. Dunlop, H. H. Nguyen, R. Barton og J. Henry, «Interference Analysis for LoRa Chirp Spread Spectrum Signals,» IEEE Xplore, 11 oktober 2019. [Internett]. Hentet fra: https://ieeexplore.ieee.org/abstract/document/8861956. [Funnet 10 mai 2023].

[12] L. Slats, «A Brief History of LoRa®: Three Inventors Share Their Personal Story at The Things Conference,» Blog Semtech, 8 januar 2020. [Internett]. Hentet fra:

https://blog.semtech.com/a-brief-history-of-lora-three-inventors-share-their-personal-story-at-the-things-conference. [Funnet 15 mai 2023].

[13] U. Noreen, A. Bounceur og L. Clavier, «A study of LoRa low power and wide area network technology,» IEEE Xplore, 23 oktober 2017. [Internett]. Hentet fra: https://ieeexplore.ieee.org/abstract/document/8075570. [Funnet 6 mai 2023].

[14] A. Zourmand, A. L. K. Hing, C. W. Hung og M. A. Rehman, «Internet of Things (IoT) using LoRa technology,» IEEE Xplore, 5 september 2019. [Internett]. Hentet fra: https://ieeexplore.ieee.org/abstract/document/8825008. [Funnet 9 mai 2023].

[15] LoRa Developer Portal, «What are LoRa® and LoRaWAN®?,» LoRa Alliance, 2023. [Internett]. Hentet fra: https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/.

[16] LoRa Alliance, «LoRaWAN® distance world record broken, twice. 766 km (476 miles) using 25mW transmission power,» LoRa Alliance, august 2017. [Internett]. Hentet fra: https://lora-alliance.org/lorawan-news/lorawanr-distance-world-record-broken-twice-766-km-476-miles-using-25mw-transmission/. [Funnet 18 mai 2023].

[17] RF Wireless World, «LoRa Link Budget Formula,» [Internett]. Hentet fra: https://www.rfwireless-world.com/calculators/LoRa-Link-Budget-Calculator.html. [Funnet 10 april 2023].

[18] LoRa Alliance, «About LoRa Alliance®,» LoRa Alliance, [Internett]. Hentet fra: https://lora-alliance.org/about-lora-alliance/. [Funnet 5 mai 2023].

[19] H. Øverby, «SNL,» 22 12 2022. [Internett]. Hentet fra: https://snl.no/Wi-Fi.

[20] J. B. Jonathan Valdez, «Understanding the I2C Bus,» Texas Instruments, Juni 2015. [Internett]. Hentet fra: https://www.ti.com/lit/an/slva704/slva704.pdf. [Funnet 2023].

[21] Texas Instruments, «Universal Asynchronous Receiver/Transmitter (UART),» November 2010. [Internett]. Hentet fra: https://www.ti.com/lit/ug/sprugp1/sprugp1.pdf?ts=1684618451032. [Funnet 15 Mai 2023].

[22] Texas Instruments, «Analog-to-digital converters,» Mars 2021. [Internett]. Hentet fra: https://www.ti.com/video/series/analog-to-digital-converters-adcs.html. [Funnet 15 Mai 2023].

[23] Texas Instruments, «Serial Peripheral Interface,» Mars 2012. [Internett]. Hentet fra: https://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf?ts=1684634964188&ref_url=https%253A%252F%252Fwww.google.com%252F. [Funnet 15 Mai 2023].

[24] Tiny, «Modular Programming Principle,» Tiny, 10 Mai 2023. [Internett]. Hentet fra: https://www.tiny.cloud/blog/modular-programming-principle/. [Funnet 10 Mai 2023].

[25] Tinkercad, «Tinkercad,» Autodesk, inc, 10 Mai 2023. [Internett]. Hentet fra: Tinkercad.com. [Funnet 10 Mai 2023].

[26] LilyGo, «LILYGO® TTGO ESP32-Paxcounter LoRa32 V2.1 1.6 Version 433/868/915MHZ LoRa ESP-32 OLED,» [Internett]. Hentet fra:

http://www.lilygo.cn/claprod_view.aspx?TypeId=62&Id=1130&FId=t28:62:28. [Funnet 15 mai 2023].

[27] Sensirion, «Datasheet Sensirion SCD30 Sensor Module,» Mouser, mai 2020. [Internett]. Hentet fra: https://no.mouser.com/datasheet/2/682/Sensirion_CO2_Sensors_SCD30_Datasheet-1901872.pdf. [Funnet 15 mai 2023].

[28] Hanwei Eletronics co.,ltd, «Technical Data MQ-2 Gas Sensor,» Mouser, [Internett]. Hentet fra: https://www.mouser.com/datasheet/2/321/605-00008-MQ-2-Datasheet-370464.pdf. [Funnet 15 mai 2023].

[29] Sensirion, «Datasheet SPS30,» Sparkfun, [Internett]. Hentet fra: https://www.sparkfun.com/products/15103 . [Funnet 15 mai 2023].

[30] Seeed Studio, «Grove - Oxygen Sensor Pro(GGC2330-O2),» Seeed Studio, [Internett]. Hentet fra: https://wiki.seeedstudio.com/Grove-Oxygen-Sensor-Pro/. [Funnet 15 mai 2023].

[31] Pi Supply, «PIS-1129 - PiJuice LiPo Battery, 12000mAh, Pi Supply,» Elfa Distrelec, [Internett]. Hentet fra: https://www.elfadistrelec.no/en/pijuice-lipo-battery-12000mah-pi-supply-pis-1129/p/30163381. [Funnet 15 mai 2023].

[32] Raspberry Pi, «Raspberry Pi 3 Model B+,» Raspberry Pi, [Internett]. Hentet fra: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf. [Funnet 15 mai 2023].

[33] Adafruit, «Adafruit RFM69HCW and RFM9X LoRa Packet Radio Breakouts,» Elfa Distrelec, 22 august 2018. [Internett]. Hentet fra: https://www.elfadistrelec.no/Web/Downloads/_t/ds/Adafruit_RFM95_eng_tds.pdf. [Funnet 13 mai 2023].

[34] Arduino Docs, «Arduino Integrated Development Environment (IDE),» Arduino, 16 mai 2023. [Internett]. Hentet fra: https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics. [Funnet 20 mai 2023].

[35] Geany, «About Geany,» Geany, 2023. [Internett]. Hentet fra: https://www.geany.org/about/geany/. [Funnet 10 mai 2023].

[36] Arduino Docs, «Arduino IoT Cloud,» 2023. [Internett]. Hentet fra: https://docs.arduino.cc/arduino-cloud/. [Funnet 13 mai 2023].

[37] Arduino, «SPI.h,» GitHub, 4 januar 2012. [Internett]. Hentet fra: https://github.com/arduino/ArduinoCore-avr/blob/master/libraries/SPI/src/SPI.h. [Funnet 25 april 2023].

[38] sandeepmistry, «LoRa.h,» GitHub, 20 august 2016. [Internett]. Hentet fra: https://github.com/sandeepmistry/arduino-LoRa/blob/master/src/LoRa.h. [Funnet 25 april 2023].

[39] Sensirion, «sps30.h,» GitHub, 12 mars 2019. [Internett]. Hentet fra: https://github.com/Sensirion/arduino-sps/commits/master/sps30.h. [Funnet 28 april 2023].

[40] Adafruit, «Adafruit_SCD30.h,» GitHub, 19 desember 2020. [Internett]. Hentet fra: https://github.com/adafruit/Adafruit_SCD30. [Funnet 30 april 2023].

[41] zhouhan0126, «Wifimanager-ESP32.h,» Github, 18 august 2017. [Internett]. Hentet fra: https://github.com/zhouhan0126/WIFIMANAGER-ESP32. [Funnet 22 mai 2023].

[42] Skanska , «E16 Bjørum - Skaret,» Skanska, 2020. [Internett]. Hentet fra: https://www.skanska.no/hva-vi-gjor/prosjekter/249448/E16-Bjorum-Skaret. [Funnet 13 mai 2023].

[44] LastMinuteEngineers, «LastMinuteEngineers,» LastMinuteEngineers, 1 Mai 2023. [Internett]. Hentet fra: https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/. [Funnet 8 Mai 2023].

[45] K. Rembor, E. Herrada og C. Nelson, «Adafruit,» Adafruit Industries, 28 Juli 2022. [Internett]. Hentet fra: https://learn.adafruit.com/adafruit-scd30/field-calibration. [Funnet 10 Mai 2023].

[46] Seeed Studio, «Seeed Studio,» 23 Juli 2022. [Internett]. Hentet fra: https://wiki.seeedstudio.com/Grove-Oxygen-Sensor-Pro/. [Funnet 1 Mai 2023].

[47] Store norske leksikon, «Stuff,» Store norske leksikon, 7 Juni 2019. [Internett]. Hentet fra: https://snl.no/stuff. [Funnet 1 Februar 2023].

[48] A. Johansen, J. Alexander Langlo, N. Olsson og A. Rolstadås, Praktisk prosjektledelse - fra idé til gevinst, Trondheim: Fagbokforlaget, 2020.

[49] L. Halbo, «Kvalitetssikring,» Store Norske Liksikon, 6 Januar 2020. [Internett]. Hentet fra: https://snl.no/kvalitetssikring. [Funnet 20 Mars 2023].

8. Vedlegg

VEDLEGG 1. KODE FOR SENSORBOKSEN

```
//LoRa Libraries
#include <SPI.h>
#include <LoRa.h>

//Sensor Libraries
#include <sps30.h>
#include <Adafruit_SCD30.h>

Adafruit_SCD30 scd30;

//Definerer LoRa pins
#define SCK 5
#define MISO 19
#define MOSI 27
#define CS 18
#define RST 23
#define DIO0 26
//Definerer RGB pins
uint8_t ledR = 12;
uint8_t ledG = 13;
uint8_t ledB = 14;

//Definerer farger

uint8_t redColour[3] = {255, 0, 0};
uint8_t greenColour[3] = {0, 255, 0};
uint8_t blueColour[3] = {0, 0, 255s};
uint8_t yellowColour[3] = {150, 0, 150};
uint8_t orangeColour[3] = {255, 165, 0};
uint8_t cyanColour[3] = {0, 255, 255};
uint8_t noColour[3] = {0, 0, 0};

uint32_t R, G, B;
uint32_t frequency = 12000;
uint8_t resolution = 8;

//Databehandling på mottakersiden
char delim[] = "|";
char *data_tokens[4];
char *pch = NULL;

//Globale sensorverdier
//Grove MQ2 gassensor
float global_gas = 0;
```

```
float R0 = 2.2;

//Seeed-Grove GGC2330-O2 oksygensensor
float global_o2 = 0;

//Sensirion SPS30 støvsensor
int auto_clean_days = 4;
float global_particle_size = 0;
//Massekonsentrasjon og stoffmengde for små og farlige støvpartikler
float global_dust_mc2_5 = 0;
float global_dust_nc2_5 = 0;

//Massekonsentrasjon og stoffmengde for større støvpartikler
float global_dust_mc10 = 0;
float global_dust_nc10 = 0;

//Adafruit SCD30 co2-, temperatur- og fuktighetssensor
int global_temp     = 0;
int global_co2      = 0;
int global_humidity = 0;

//Batteristatus
float battery_voltage = 0;

//Kommandovariabler
int command = 0;
int interval = 5000;

//Millis som delay
unsigned long previous_millis;
unsigned long current_millis;

int counter = 0;

void setup() {
  Serial.begin(9600);
  //Omkobler UART RX og TX til GPIO 36 og 39, fordi default-pins brukes til
skjerm-reset og skaper feil ved opplasting.
  Serial2.begin(9600, SERIAL_8N1, 36, 39);
  //Initialisering av støvsensor og co2-sensor
  scd30_init();
```

```
  sps30_init();
  //Initialisering av LoRa-modulen
  lora_init();
   //Initialisering av RGB led
  ledcAttachPin(ledR, 1); // tildeler RGB led pins til riktige kanaler
  ledcAttachPin(ledG, 2);
  ledcAttachPin(ledB, 3);

  ledcSetup(1, frequency, resolution); // 12 kHz PWM, 8-bit resolution
  ledcSetup(2, frequency, resolution);
  ledcSetup(3, frequency, resolution);

}

void loop() {
  current_millis = millis();

  if (current_millis - previous_millis >= interval) {
    lora_transmit();
    previous_millis = millis();
  }

  else if (command == 1) {
    recalibrate();
  }

  else {
    lora_receive();
  }
}

//Initialisering av LoRa-pinner og frekvensområde.
void lora_init() {
  setColour(greenColour);
  SPI.begin(SCK, MISO, MOSI, SS);
  LoRa.setPins(SS, RST, DIO0);
  setColour(noColour);

  if (!LoRa.begin(868E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}
```

```
//Senderdelen oppdaterer sensordata og sender disse som 1 streng med
separatorer.
void lora_transmit() {
  setColour(blueColour);
  o2_read();
  gas_read();
  scd30_read();
  sps30_read();
  battery_read();
  setColour(noColour);



  //Lager string av sensorverdier med separatoren "|".
  String separator = "|";
  String sensor_values = global_gas + separator + global_o2 + separator +
global_particle_size + separator + global_dust_mc2_5 + separator +
global_dust_nc2_5 + separator + global_dust_mc10 + separator +
global_dust_nc10 + separator + global_temp + separator + global_co2 +
separator + global_humidity + separator + battery_voltage;

  Serial.print("Sending packet: ");
  Serial.println(counter);
  counter++;

  //Sender sensordata
  LoRa.beginPacket();
  LoRa.print(sensor_values);
  LoRa.endPacket();
  Serial.println(sensor_values);

  //Nødvendig delay for skiftet fra transmitter- til receivermodus
  delay(50);
}

//Mottakerdelen leser kommandoer som kommer fra sentral mikrokontroller når
den ikke sender data.
void lora_receive() {
  int packetSize = LoRa.parsePacket();
  //Sjekker om LoRa modulen har mottatt data
```

```
  if (packetSize) {
    //Mottatt packet
    setColour(greenColour);
    Serial.print("Received packet '");

    //Setter opp indekser for arrays som skal lagre mottatt LoRa-data.
    int i  = 0;
    int i2 = 0;
    char lora_message[LoRa.Hentet fra()] = {};

    //Leser av en byte om gangen og setter dette inn i arrayet
lora_message[].
    while (LoRa.Hentet fra()) {
      lora_message[i] = LoRa.read();
      i++;
    }

    pch = strtok(lora_message, delim);

    while (pch != NULL) {
      data_tokens[i2] = pch;
      i2++;
      pch = strtok(NULL, delim);
    }

    command  = atoi(data_tokens[0]);
    interval = atoi(data_tokens[1]);

    Serial.println(command);
  }
}

void o2_read() {
  unsigned long o2_warmup_millis = 0;
  current_millis = millis();

  if (current_millis - o2_warmup_millis >= 300000) {
    while (Serial2.Hentet fra() > 0){
      uint8_t start_byte = Serial2.read();
      delay(10);
      uint8_t command_byte = Serial2.read();
```

```
      delay(10);
      uint8_t concentration_high = Serial2.read();
      delay(10);
      uint8_t concentration_low = Serial2.read();
      delay(10);
      uint8_t check_byte_1 = Serial2.read();
      delay(10);
      uint8_t check_byte_2 = Serial2.read();
      delay(10);
      uint8_t check_byte_3 = Serial2.read();
      delay(10);
      uint8_t check_byte_4 = Serial2.read();
      delay(10);
      uint8_t checksum_byte = Serial2.read();
      delay(10);

      if(start_byte == 255 && command_byte == 134){

      float o2_val = ((concentration_high * 256) + concentration_low) *
0.1;
      global_o2 = o2_val;
      }
    }
    while(Serial2.read()>=0);    //clear buffer
  }

  else {
    Serial.println("O2 sensor warming up");
  }
}

void gas_read() {
  float sensor_volt;
  float rs_gas;
  float gas_air_ratio;
  float sensorValue = analogRead(34);

  unsigned long gas_warmup_millis = 0;
  current_millis = millis();

  if (current_millis - gas_warmup_millis >= 300000) {
```

```
    sensor_volt = sensorValue/1024*5.0;
    rs_gas = (5.0-sensor_volt)/sensor_volt;

    gas_air_ratio = rs_gas/R0;
    global_gas = gas_air_ratio;
  }

  else {
    Serial.println("Gas sensor warming up");
  }
}

void gas_calibrate() {
  //Legg gassensor i ren luft før kalibrasjon
  float sensor_volt;
  float rs_air;
  float sensor_val;

  for(int x = 0; x < 100; x++) {
    sensor_val = sensor_val + analogRead(34);
  }
  sensor_val = sensor_val / 100.0;

  sensor_volt = sensor_val / 1024 * 5.0;
  rs_air = (5.0 - sensor_volt) / sensor_volt;
  R0 = rs_air / 9.8;

  delay(1000);
}

void sps30_init() {
  uint16_t ret;
  uint32_t auto_clean;

  sensirion_i2c_init();

  while (sps30_probe() != 0) {
    Serial.print("SPS sensor probing failed\n");
    delay(500);
  }
```

```
  #ifndef PLOTTER_FORMAT
    Serial.print("SPS sensor probing successful\n");
  #endif

  ret = sps30_set_fan_auto_cleaning_interval_days(auto_clean_days);
  if (ret) {
    Serial.print("error setting the auto-clean interval: ");
    Serial.println(ret);
  }
}

void sps30_read() {
  struct sps30_measurement m;
  char serial[SPS30_MAX_SERIAL_LEN];
  uint16_t data_ready;
  int16_t ret;

  ret = sps30_start_measurement();
  if (ret < 0) {
    Serial.print("error starting measurement\n");
  }

  do {
    ret = sps30_read_data_ready(&data_ready);
    if (ret < 0) {
      Serial.print("error reading data-ready flag: ");
      Serial.println(ret);

    } else if (!data_ready)
      Serial.print("data not ready, no new measurement Hentet fra\n");
    else
      break;
    delay(100); /* retry in 100ms */
  } while (1);

  ret = sps30_read_measurement(&m);
  if (ret < 0) {
    Serial.print("error reading measurement\n");
  } else {

    global_dust_mc2_5 = m.mc_2p5;
```

```
    global_dust_mc10 = m.mc_10p0;

    global_dust_nc2_5 = m.nc_2p5;
    global_dust_nc10 = m.nc_10p0;

    global_particle_size = m.typical_particle_size;

    sps30_stop_measurement();
  }
}

//Manuell rensing av støvsensor, må settes i målemodus først.
void sps30_calibrate() {
  sps30_start_measurement();
  sps30_start_manual_fan_cleaning();
  sps30_reset();
  delay(100);
  sps30_stop_measurement();
}

void scd30_init() {
  if (!scd30.begin()) {
    Serial.println("Failed to find SCD30 chip");
    while (1) { delay(10); }
  }
  Serial.println("SCD30 Found!");
}

void scd30_read() {
  if (scd30.dataReady()) {
    Serial.println("Data Hentet fra!");

    if (!scd30.read()) {
      Serial.println("Error reading sensor data");
      return;
    }

    global_temp = scd30.temperature;
    global_humidity = scd30.relative_humidity;
    global_co2 = scd30.CO2;
  }
```

```
}

void scd30_calibrate() {
  unsigned long co2_calibration_millis = millis();
  current_millis = millis();

  if (current_millis - co2_calibration_millis >= 120000) {
    if (!scd30.forceRecalibrationWithReference(400)) {
      Serial.println("Failed to force recalibration with reference");
      while(1) { delay(10); }
    }
  }
}

//Enkel kalibrasjon hvis ønsket
void recalibrate() {
  sps30_calibrate();
  gas_calibrate();
  scd30_calibrate();

  command = 0;
}

void battery_read() {
  float adc_read_35 = analogRead(35);
  battery_voltage = adc_read_35 / 4095 * 2 * 3.3 * 1.1;
}
void setColour(uint8_t colourCode[3])
{
  R = colourCode[0];
  G = colourCode[1];
  B = colourCode[2];
  ledcWrite(1, R);
  ledcWrite(2, G);
  ledcWrite(3, B);
}
```

VEDLEGG 2. KODE FOR MOTTAKERENHETEN

```
#include <SPI.h>
#include <LoRa.h>
#include <UbidotsEsp32Mqtt.h>
#include <WiFiManager.h>

//Define LoRa pins
#define SCK 5
#define MISO 19
#define MOSI 27
#define CS 18
#define RST 23
#define DIO0 26

//Ubidots tokens og sånt
//////////////////////////////////////////////////////////////////////

const char *UBIDOTS_TOKEN = "BBFF-5yeBcBxGcURyMBJKH9kCUKtWIKg4WB";  // Put
here your Ubidots TOKEN

const char *DEVICE_LABEL = "lora-gateway";   // Put here your Device label
to which data  will be published

//Variabel navn  for ubidots
/////////////////////////////////////////////////
const char *gas = "Gass [PPM]";
const char *o2 = "Oksygen Vol[%]";
const char *particlesize = "Partikkelstørrelse [um]";
const char *dustmc25 = "PM2.5 Massekonsentrasjon [ug/m3]";
const char *dustnc25 = "PM2.5 Stoffmengde [#/m3]";
const char *dustmc10 = "PM10 Massekonsentrasjon [ug/m3]";
const char *dustnc10 = "PM10 Stoffmengde [#/m3]";
const char *temperature = "Temperatur [C]";
const char *co2 = "CO2 [PPM]";
const char *humidity = "Fuktighet [%]";
const char *batteryvoltage = "Batterispenning [V]";
const char *signalstrength = "Signalstyrke [rssi]";

// Variabel navn for koden
float global_gas = 0;
float global_o2 = 0;
```

```
float global_particle_size = 0;
float global_dust_mc2_5 = 0;
float global_dust_nc2_5 = 0;
float global_dust_mc10 = 0;
float global_dust_nc10 = 0;
float global_temp = 0;
float global_co2 = 0;
float global_humidity = 0;
float battery_voltage = 0;
float signal_strength = 0;

//#################################################

char delim[] = "|";
char *strings[13];
char *pch = NULL;

Ubidots ubidots(UBIDOTS_TOKEN);

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

  //LoRa#########################################
  SPI.begin(SCK, MISO, MOSI, SS);
  LoRa.setPins(SS, RST, DIO0);

  while (!Serial);

  Serial.println("LoRa Receiver");

  if (!LoRa.begin(868E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }

  //WiFi#########################################
  WiFi.mode(WIFI_STA);

  WiFiManager wm;

  bool res;
```

```
  res = wm.autoConnect("LoRaGateway", "E2317Bachelor");

  if (!res) {
    Serial.println("Failed to connect");
  }

  else if (res) {
    //Ubidots#####################################
    ubidots.setCallback(callback);
    ubidots.setup();
    ubidots.reconnect();
  }
}

void loop() {
  // try to parse packet

  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // read packet

    int i = 0;
    int i2 = 0;
    char message[int(LoRa.Hentet fra())];

    while (LoRa.Hentet fra()) {
      message[i] = LoRa.read();
      i++;
    }
    pch = strtok(message, delim);

    while (pch != NULL) {
      strings[i2] = pch;
      i2++;

      if (i2 > 11) {
        global_gas           = atof(strings[0]);
        global_o2            = atof(strings[1]);
        global_particle_size = atof(strings[2]);
        global_dust_mc2_5    = atof(strings[3]);
        global_dust_nc2_5    = atof(strings[4]);
```

```
        global_dust_mc10      = atof(strings[5]);
        global_dust_nc10      = atof(strings[6]);
        global_temp           = atof(strings[7]);
        global_co2            = atof(strings[8]);
        global_humidity       = atof(strings[9]);
        battery_voltage       = atof(strings[10]);
        signal_strength       = atof(strings[11]);
      }
      pch = strtok(NULL, delim);
    }

    ubidots_upload();
  }
}
void callback(char *topic, byte *payload, unsigned int length)
{
  Serial.print("Message arrived");
}
void ubidots_upload() {
  Serial.println("Uploading...");
  ubidots.add(gas, global_gas);
  ubidots.add(o2, global_o2);
  ubidots.add(particlesize, global_particle_size);
  ubidots.add(dustmc25, global_dust_mc2_5);
  ubidots.add(dustnc25, global_dust_nc2_5);
  ubidots.publish(DEVICE_LABEL);

  delay(1000);

  ubidots.add(dustmc10, global_dust_mc10);
  ubidots.add(dustnc10, global_dust_nc10);
  ubidots.add(temperature, global_temp);
  ubidots.add(co2, global_co2);
  ubidots.add(humidity, global_humidity);
  ubidots.publish(DEVICE_LABEL);

  delay(1000);

  ubidots.add(batteryvoltage, battery_voltage);
  ubidots.add(signalstrength, signal_strength);
  ubidots.publish(DEVICE_LABEL);}
```

VEDLEGG 3. PLAKAT

E 2 3 1 7

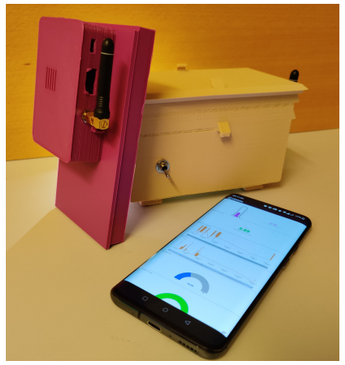
LORAWAN PARAMETEROVERVÅKNING I TUNNELANLEGG

ABSTRAKT

Denne bacheloroppgaven ble utført i samarbeid med Skanska Norge AS med hensikt å opprette god datakommunikasjon, effektivisere og automatisere parameterovervåkning i tunnelanlegg. Løsningen er en Internet-of-Things (IoT) system bygget rundt en sensorboks som måler luftkvaliteten i tunnel og sender avlesningene til en tjenerenhet trådløst. Den trådløse kommunikasjonen oppnås med en relativt ny teknologi kalt for LoRa (fra "Long Range"). Fordelene med LoRa-teknologien er lang rekkevidde, god energieffektivitet, kompatibilitet med IoT prosjekter. Avlest sensordata lastes opp til Ubidots av LoRa-tjeneren, og vises med ulike grafer og grafikk. Systemet har ikke blitt testet grundig i tunnelanlegg grunnet tekniske feil på testdagen, og et urbant område i byen ble derfor brukt som et alternativt testmiljø. Under testen ble det oppnådd en maksimal rekkevidde på 650m med moderat mengde. Det finnes mange forbedringer som kan gjøres på systemet, som å bytte til mer industrielle sensorer, bygge en mer robust innkapsling, lage egnet kretskort og forbedre varslingssystemet.

TEORETISK BAKGRUNN

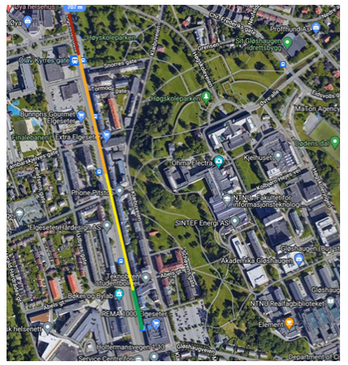
Luftforurensning er den mest skadelige miljøfaktoren for helsen og er årsaken til flere enn tusen tidlige dødsfall i Norge, og over tre millioner tidlige dødsfall i verden. I tunnelanlegg forekommer det ofte høy konsentrasjon av forurensningskomponenter på grunn av begrenset luftmengde og tilgang til fersk uteluft. Derfor er det viktig å overvåke luftkvaliteten i tunnelen for å bevare god helse hos anleggsarbeidere. Fjern overnåkning oppnås ved å bruke den trådløse kommunikasjonsprotokollen LoRa, dersom WiFi ikke dekker de siste 500 meterne frem mot stuff. LoRa tilbyr viktige fordeler som lang rekkevidde og lav energiforbruk.



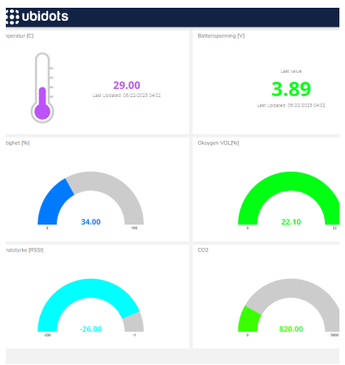
METODE OG REALISERING

Avsenderen og mottakeren i systemet bruker utviklerkortet TTGO LoRa32 av LilyGo, som har ESP32 i kjernen og en SX1276 LoRa-modul. Mikrokontrolleren i sensorboksen er batteridrevet av en 3.7V 12000mAh LiPo, med en beregnet levetid på over 70 timer. Sensorboksen inneholder en gassensor, oksygensensor, partikkelsensor og en sensor som måler temperatur, karbondioksid og fuktighet. Det blir brukt I2C, UART og ADC for å hente data fra sensorene, og blir sendt som én datapakke med LoRa hvert femte sekund. Mottakeren kobles til Ubidots via WiFi, og oppdaterer dashbordverdiene når den mottar og behandler LoRa-data fra avsenderen.

RESULTATER



Under testbesøket i et av Skanskas tunnelanlegg på E16 i Viken-fylket, ble det oppdaget at mikrokontrolleren i sensorboksen ble ødelagt under transport. Hensikten med anleggsbesøket var å teste LoRa rekkevidden i tunnel, men denne planen endret seg til å måle med kun partikkelsensoren før og etter sprengning. Data vi hentet fra testen viste en kraftig økning i antall partikler i lufta ti minutter etter sprengning, og minker når ventilasjonssystemet pumper luft i tunnelen. Rekkevidden ble istedet testet etter besøket, i en langstrakt vei ved Elgeseter gate mot Studentersamfundet. Da ble det oppnådd en maksimal rekkevidde på ca. 700 meter, før dataopplasting stoppet.



DISKUSJON

De utvalgte sensorene er mer egnet til bruk for utvikling, og kan erstattes med mer pålitelige og robuste sensorer ment for industriell bruk. Gassensoren spesielt burde erstattes med sensorer som gir ut individuelle målinger for forskjellige gasstyper. Resultatet av LoRa-testen kan også avvike fra hvordan systemet presterer i et tunnelanlegg, og bør derfor testes videre i riktige forhold. Obstruksjonene i testmiljøet er også veldig annerledes og uforutsigbare i forhold til de ofte stasjonære obstruksjonene i tunnelene. Anvendelsen av en mer direktiv antenne kan også være gunstig for å øke rekkevidden i rettere tunneler, men vil tape distanse i mer komplekse tunneler med varierende høyde og svinger.

EDVARD LIE LOURENZO RAY LAPIZ JULIA CZARNOCKA ERIK KLANWILAI NIELSEN

NTNU × SKANSKA