Bacheloroppgave

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

Erlend Gran
Joakim Olerud Jensen
Morten Jentoftsen

# Trådløs datalogging for produkttest

Bacheloroppgave i ingeniørfag, Elektro
Veileder: Knut Wold

Mai 2023

NTNU
Kunnskap for en bedre verden

Tokvam®

Erlend Gran
Joakim Olerud Jensen
Morten Jentoftsen

# Trådløs datalogging for produkttest

Bacheloroppgave i ingeniørfag, Elektro
Veileder: Knut Wold
Mai 2023

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

NTNU
Kunnskap for en bedre verden

# SAMMENDRAG

| Oppgavens tittel: | Dato: | | 19.05.2023 | |
|---|---|---|---|---|
| Trådløs datalogging for produkttest | Antall sider: | | 63 | |
| | Antall Vedlegg: | | 7 | |
| | **Masteroppgave:** | | **Bacheloroppgave:** | X |
| **Navn:**<br>Erlend Gran, Joakim Olerud Jensen, Morten Jentoftsen | | | | |
| **Veileder:**<br>Knut Wold | | | | |
| **Kontaktperson:**<br>Jostein Håkonsen Kvikstad | | | | |


**Sammendrag**

Denne rapporten tar for seg utviklingen av et system, for trådløs datalogging til bruk under testing av produkter for snøbrøyting og kantklipping hos Tokvam AS. Målet med oppgaven var å kunne pålitelig samle inn, logge og fremvise sensorverdier med så lite kabling som mulig.

Det har blitt utviklet flere batteridrevne sensormoduler med mulighet for trådløs dataoverføring med bruk av nettverksprotokollen ESP-NOW. Systemet består i tillegg av en mottaker og en server som kjører programvare for logging og fremvisning av måleverdiene. Rapporten tar også for seg testing av begrensinger og muligheter ved bruk av ESP-NOW for trådløs dataoverføring.

Resultatet i rapporten er tre sensornoder for måling av temperatur, trykk og vibrasjon med mulighet for produksjon og videreutvikling av flere noder for oppdragsgiveren, i tillegg til en mottakernode. For å logge og fremvise sensorverdiene brukes en bærbar PC med programmene Node-Red, InfluxDB og Grafana installert. Sensornodene og mottakeren bruker mikrokontrolleren ESP-32 for å realisere ESP-NOW-kommunikasjon.

Systemet fungerer som tiltenkt, med pålitelig trådløs dataoverføring med bruk av ESP-NOW. Den trådløse kommunikasjonen fungerer bedre enn forventet, med lang rekkevidde. Batterilevetiden til sensornodene er bra, og fungerer fint til flere fulle arbeidsdager med produkttest. Systemet for logging og fremvisning av måleverdiene fungerer som tiltenkt, men kunne vært bedre ved bruk av annen maskinvare.



**Nøkkelord:** ESP32, ESP-NOW, trådløs kommunikasjon, sensorsystemer

# ABSTRACT

| Title: | Date: | | 19.05.2023 | |
|---|---|---|---|---|
| Wireless data collection for product testing | Number of pages: | | 63 | |
| | Number of attachments: | | 7 | |
| | **Masteroppgave:** | | **Bacheloroppgave:** | X |
| **Name:**<br>Erlend Gran, Joakim Olerud Jensen, Morten Jentoftsen | | | | |
| **Supervisor:**<br>Knut Wold | | | | |
| **Contact person:**<br>Jostein Håkonsen Kvikstad | | | | |

**Abstract**

This thesis consists of the development of a wireless data collection system for use during product testing of snow plowing and roadside clearing equipment. The assignment was provided by Tokvam AS. The goal of the thesis was to reliably collect, log, and display sensor values using as little wiring as possible.

Several battery-powered sensor modules capable of wireless data transmission using the ESP-NOW network protocol was developed. In addition, the system consists of a receiver and a server running software for logging and displaying the measured values. The project also explores the limitations and possibilities of using ESP-NOW for wireless data transmission.

The result of the thesis is three sensor nodes for measuring temperature, pressure, and vibration, including the possibility to produce additional sensor-nodes. A laptop with the programs Node-Red, InfluxDB, and Grafana installed is used to log and display the sensor values. The sensor nodes and the receiver use the ESP-32 microcontroller to utilize ESP-NOW communication.

The system functions as intended, with reliable wireless data transmission using ESP-NOW. The wireless communication performs better than expected, with a long range. The battery life of the sensor nodes is good and works well for multiple full workdays of product testing. The system for logging and displaying the measured values functions as intended but could be improved with the use of different hardware.

**Keywords:** ESP32, ESP-NOW, wireless communication, sensor systems

# FORORD

«Trådløs datalogging for produkttest» er utført av tre studenter ved studiet elektroingeniør på NTNU i Gjøvik. Oppgaven ble forespurt av oss, og i samarbeid med Tokvam fant vi en reell utfordring som vi kunne være med å løse. Oppgaven går ut på å kunne samle inn fysiske data under produkttest, loggføre og fremstille dataen for analysering. Tidligere har testing vært gjort manuelt uten mulighet for logging. Her kunne vi være med på å finne en løsning og bruke det vi har lært de siste tre årene.

Oppgaven har gitt oss uvurderlig erfaringen med prosjektarbeid og produktutvikling i tverrfaglige utfordringer som vi kan møte i arbeidslivet, samt hvordan dokumentere prosessen.

Vi ønsker å takke Tokvam AS for oppgaven og erfaringen den har gitt oss. Takk til alle som har veiledet oss under alle stadier av oppgaven, og vi vil spesielt gi vår takk til:

- Knut Wold fra NTNU for veiledning
- Jostein Håkonsen Kvikstad fra Tokvam AS for samarbeid og god kommunikasjon under hele prosessen.
- IKM Laboratorium på Raufoss for bistand med kalibrering av sensor.

| | | |
|---|---|---|
| Erlend Gran | Joakim Olerud Jensen | Morten Jentoftsen |
| 19.05.2023 | 19.05.2023 | 19.05.2023 |

# INNHOLD

# FIGURER

# TABELLER

# ORDLISTE/FORKORTELSER

| | |
|---|---|
| ADC | Analog-to-Digital-Converter |
| CAD | Computer-Aided Design, er programvare for teknisk design og erstatter manuelle utkast. |
| CCMP | En krypteringsprotokol for trådløse LAN |
| GPIO | General Purpose Input/Output |
| Halv-dupleks | Toveis kommunikasjon som foregår i en retning av gangen |
| JSON | JavaScript Objective Notation |
| IoT | Internet of Things |
| LoRa | Er en trådløs kommunikasjonsprotokoll, designet for å sende små datamengder over lange avstander |
| IP grad | «International Protection» Hvor godt kapslingen beskytter mot støv og vann. |
| $I^2C$ | En seriell busskommunikasjonsprotokoll |
| MEMS | Micro-Electro-Mechanical-Systems, kombinerer elektriske og mekaniske bevegelige komponenter |
| MQTT | Er en lettvekts kommunikasjonsprotokoll, designet for lav båndbredde og ustabile nettverk. |
| PBM/PWM | Pulsbreddemodulasjon |
| RISC-V | Er en åpen kildekode instruksjonssettarkiterur basert på RISC prinsipper |
| SCA/SCL | Linje for klokkesignalet i $I^2C$ protokollen |
| SDA | Linje for dataoverføring i $I^2C$ protokollen |
| SoftAP | Software enabled Access Point, gjør det mulig å konfigurere Wifi applikasjoner uten display |
| STA | Station Mode. Enhet kan automatisk tilkobles tilgjengelige nettverk |

# 1 INNLEDNING

## 1.1 BAKGRUNN

Oppdrag gitt fra Tokvam er bakgrunn for rapporten. Tokvam er en norsk produksjonsbedrift på Reinsvoll i Vestre Toten kommune, som har produsert utstyr og løsninger for snøbrøyting siden 1958. De produserer snøfresere, ploger og strømaskiner, og har de siste årene begynt med produksjon av krattknusere. Selve oppdraget går ut på datalogging med ulike sensorer når disse redskapene skal testes.

## 1.2 PROBLEMSTILLING

**«Hvordan loggføre sensorverdier under produkttesting på en oversiktlig måte? Hvordan måle de analoge verdiene og omforme til digitale verdier, med bruk av minst mulig kabling?»**

Målet med oppgaven er å utvikle et system for trådløs innsamling av data fra ulike sensorer. Dette til én enkel enhet for oversikt av sensorverdier i sanntid og loggføring. Systemet skal realisere trådløs dataoverføring, knyttet til ulike sensorer for temperatur-, vibrasjon- og oljetrykkmåling. Til slutt er målet å konstruere tre sensornoder for bevis av konsept, og løsning på problemstillingen.

## 1.3 OMFANG OG BEGRENSNINGER

Oppgaven går ut på å utarbeide et system for datalogging av sensorverdier under produkttesting, ved å benytte flere sensorer og fremstille dataen på en god måte. Dette innebærer å utvikle et grensesnitt for trådløs kommunikasjon, for å loggføre og fremstille data. Systemet skal kunne brukes på forskjellige typer produkter, og bør derfor lages så universalt som mulig. Oppgaven innebærer også valg av forskjellige sensorer og målområder. Det trådløse grensesnittet skal gjøre om signaler fra de forskjellige sensorene til digitale signaler, og sende det trådløst til en lokal server. Sensornodene skal være batteridrevet, og mottakernoden har strøm tilgjengelig over USB fra en PC. Løsningen må kunne tåle miljøpåkjenningene som vann og kulde for å kunne brukes på Tokvam sine produkter under testing.

Prosjektet har som mål å implementere følgene sensorer:

- Temperaturmålinger for hydraulikkolje og luft
- Trykkmåling på hydraulikkolje
- Vibrasjonssensor

## 1.4 RAPPORTINNHOLD

Rapporten er inndelt i kapitler for å vise progresjon i planlegging og metodikk for valgene tatt i oppgaven. Rapporten skal gjøre det mulig å gjenskape løsningen og resultatene. En kort beskrivelse av kapitlene er gitt under:

**Kapittel 2 – Teori:** Forklarer teorien bak elementene i oppgaven, og det teoretiske grunnlaget for å kunne forstå innholdet i rapporten.

**Kapittel 3 – Metode:** Kapittelet beskriver fremgangsmåten for oppbygning av nodene med alle stegene inkludert. Først en oversikt over hele systemet, deretter beskrives koden, oppbygningen av nodene og til slutt forklares de forskjellige ulike nodene i detalj.

**Kapittel 4 – Resultater:** Kapittelet inneholder resultater for ulike tester tatt på nodene. I tillegg til en test av hele systemet på krattknuser i drift, med beskrivende metodikk og utførelse.

**Kapittel 5 – Diskusjon:** I dette kapittelet diskuteres utfordringer i prosessen og analyse av resultatene fra testene opp mot målene i oppgaven. Her beskrives også hva som kan forbedres ved eventuell videreutvikling av produktet.

**Kapittel 6 – Konklusjon:** Kapittelet gir en konklusjon på oppgaven med en oppsummering av resultatene opp mot målene.

# 2 TEORI

Teorikapittelet tar for seg nødvendig teori til hardware og software for å gi grunnleggende forståelse av innholdet i rapporten.

## 2.1 ESP32

ESP32 [7] er en kostnadseffektiv mikrokontroller fra Espressif Systems som er utviklet for å gi funksjonalitet til IoT-systemer og enheter som kombinerer hardware og software. De ulike ESP32 modellene har varierende prosessorer med ulike klokkehastigheter. De fleste modellene har både Wi-Fi og Bluetooth tilkoblingsmuligheter.

ESP32 mikrokontrolleren gir en god balanse mellom ytelse, tilkoblingsmuligheter og sikkerhet. Den gir støtte for et bredt spekter av sensorer og enheter. Det finnes også et stort utviklerfellesskap som oppdaterer tilleggsbiblioteker og verktøy for å gjøre det lettere å utvikle og programmere ESP32 baserte enheter. I tillegg har de flere innebygde funksjoner for ADC, DAC, SPI, $I^2C$, UART og PWM. ESP32 er designet for å kunne operere under temperaturer fra –40 °C til 125 °C og egner seg derfor godt til industriell bruk [8].

En sammenligning av noen ulike ESP32 modeller: ESP32-S2, ESP32-C3 og ESP32-S3 og ulikhetene mellom disse er samlet i tabell 1.

*Tabell 1:* Forskjeller mellom ESP32-S2, ESP32-C3 og ESP32-S3 [9].

| | **ESP32-S2** | **ESP32-C3** | **ESP32-S3** |
|---|---|---|---|
| Utgivelsesår | 2020 | 2020 | 2020 |
| Hovedprosessor | Tensilica Xtensa enkelkjerne 32-bit LX7 240MHz | RISC-V 32-bit 160MHz | Tensilica Xtensa dobbelkjerne 32-bit LX7 240MHz |
| SRAM | 320KB | 400KB | 512KB |
| ROM | 128KB | 384KB | 384KB |
| Bluetooth | X | Bluetooth 5.0 | Bluetooth 5.0 |
| $I^2C$ | 2 | 1 | 2 |
| GPIO | 43 | 22 | 45 |
| USB OTG | USB OTG | X | USB OTG |

Den største forskjellen mellom ESP32-S2, ESP32-C3 og ESP32-S3 er at S2 ikke har Bluetooth tilkoblingsmulighet. ESP32-S2 og ESP32-S3 støtter USB spesifikasjonen "USB On-The-Go" (OTG). Denne teknologien gjør det mulig for to USB-enheter å kommunisere med hverandre uten styring fra en datamaskin.

Prismessig er det ikke stor forskjell på de to ulike modellene. Både ESP32-C3 og ESP32-S2 er derfor kostnadseffektive mikrokontrollere. ESP32-S3 har en litt høyere pris enn de andre.

## 2.2 ESP-NOW

ESP-NOW er en trådløs «connectionless» kommunikasjonsprotokoll utviklet av Espressif. Protokollen gjør det mulig for flere enheter å kommunisere sammen, uten å være koblet til et Wi-Fi LAN nettverk eller internett [10]. ESP-NOW meldinger blir innkapslet i eget ramme-format før det sendes. For kryptering brukes CTR (counter), med CBC-MAC protokollen, forkortet til CCMP, som er en trådløs LAN protokoll. For kommunikasjon mellom de ulike ESP-32 nodene, må MAC-adressen til mottakeren være kjent. Etablerte tilkoblinger er varige og hvis en av nodene mister tilkoblingen eller starter på nytt, vil den automatisk koble seg til igjen. [11]

ESP-NOW kan brukes i to ulike konfigurasjoner:

- Enveis kommunikasjon: En master og flere slave noder, eller én slave og flere master noder
- Toveis kommunikasjon: Direkte kommunikasjon mellom nodene

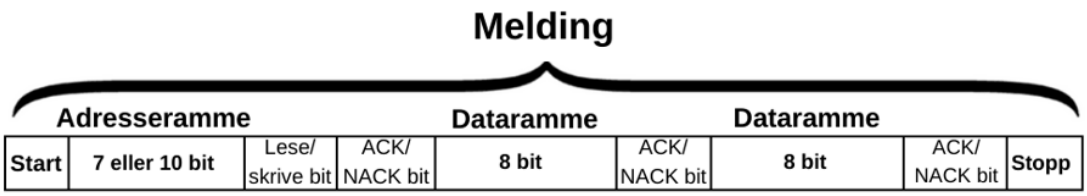
ESP-NOW har en god rekkevidde som vist i rapport [11], hvor resultatet var en stabil tilkobling ved 190 m avstand mellom nodene.

## 2.3 $I^2C$

$I^2C$ er et grensesnitt for seriellkommunikasjon imellom flere enheter via en buss. Bussen består av to ledere i tillegg til en jordtilkobling. De to lederne er seriellklokke (SCL) og serielldata (SDA), som støtter toveis kommunikasjon og er halv-duplekse. Hver $I^2C$ enhet har en unik adresse, som er nødvendig for en fungerende busskommunikasjon. I tillegg er det enkelt å legge til og fjerne enheter ved behov. [12]

All data sendes over SDA linjen, og SCL fungerer kun som klokkesignal til synkronisering. Til å begynne med sendes en «START» melding til alle enhetene på bussen. Denne meldingen signaliserer til alle tilkoblede enheter at dataoverføring vil skje. Tilsvarende «STOP» melding sendes ut til alle enhetene når overføring er fullført og at linjen er ledig for overføring av data.

Oppbygningen av en datapakke i I²C kommunikasjon vises i figur 2.1. Adressen er en 7 eller 10 bit-sekvens. Lese/skrive-bitet spesifiserer om master enheten ønsker å sende data (bitverdi 0) eller motta data (bitverdi 1). Etter hver melding følger en bekreftelsesbit ACK/NACK, som sendes hvis meldingen ble mottatt. [13]




*Figur 2.1: Oppbygning av datapakke i I²C kommunikasjon, designet i Canva.*

## 2.4 Node-RED

Node-RED er en visuell programmeringsplattform basert på Node.js og JavaScript. Plattformen gir et enkelt og brukervennlig grensesnitt der man kan koble sammen noder med forskjellige funksjoner og hardware i nettleser. Eksempler på noder er MQTT tilkoblinger, http funksjoner og datahåndteringsfunksjoner. Node-RED kan konfigureres til å kjøre på forskjellige plattformer som Android, Raspberry pi eller Windows. [14]

## 2.5 InfluxDB

InfluxDB er et databasesystem med åpen kildekode lagd for å håndtere tidsavhenging data som sensorverdier og metrikkdata. InfluxDB organiserer data i tidsserier, som er en serie datapunkter indeksert i tidsrekkefølge. [15]

## 2.6 Grafana

Grafana er et analyse- og datavisualiseringsprogram. Programmet finnes både med åpen kildekode eller som en betalingstjeneste. Grafana er et verktøy for å lage dashbord til visualisering av data fra forskjellige databaser. Programmet støtter mange forskjellige datakilder og kan integreres med andre verktøy. [16]

## 2.7 ARDUINO IDE

Arduino Integrated Development Environment (IDE) er software som brukes til å programmere og opplasting av kode til mikrokontrollere. Arduino IDE er en uavhengig plattform som kan kjøres på Windows, MacOS og Linux. Den har åpen kildekode som betyr at den er tilgjengelig for alle å se, modifisere og bidra til.

Med Arduino IDE kan brukere skrive programmer i Arduino sitt språk, som ligner på C/C++. Arduino IDE har en rekke innebygde funksjoner og biblioteker som gjør det enkelt å programmere forskjellige enheter og sensorer. Det kan overvåke seriell kommunikasjon til tilkoblede enheter, og feilsøke problemer med kode. Arduino IDE er en populær plattform for både hobby designere og profesjonelle som ønsker å bygge og utvikle elektroniske produkter. [17]

### 2.7.1 Onewire

Biblioteket en samling av funksjoner som gjør det enkelt å kommunisere med enheter som bruker OneWire-protokollen. OneWire-protokollen er en seriell kommunikasjonsprotokoll som gjør det mulig å koble flere enheter til en enkelt datapinne og overfører data mellom ved hjelp av kun én enkelt ledning. [18]

Biblioteket inkluderer funksjoner for å søke etter enheter på OneWire-bussen, lese og skrive data, i tillegg til å konfigurere avbruddshåndtering for å gi bedre ytelse og pålitelighet i kommunikasjonen. OneWire-biblioteket er inkludert i Arduino-programvaren, og er et godt alternativ for $I^2C$ når tilgjengelig, for å minimere antall ledninger.

### 2.7.2 WiFi

Wi-Fi-biblioteket [19] gir mulighet for trådløs kommunikasjon mellom enhetene ved bruk av trådløse protokoller som Bluetooth eller Wi-Fi. Biblioteket inneholder funksjoner som lar mikrokontrolleren koble til og kommunisere med andre enheter på nettverket [20].

Biblioteket gir mulighet for å administrere nettverksforbindelser og autentisering, ved å oppgi passord og sikkerhetsnøkler for å koble til et Wi-Fi-nettverk. Det gir også Arduino-enheten funksjoner for å hente og sende data via nettverket. Biblioteket er en viktig del av mange IoT-prosjekter som krever trådløs kommunikasjon og tilkobling til internett.

### 2.7.3 ArduinoJson

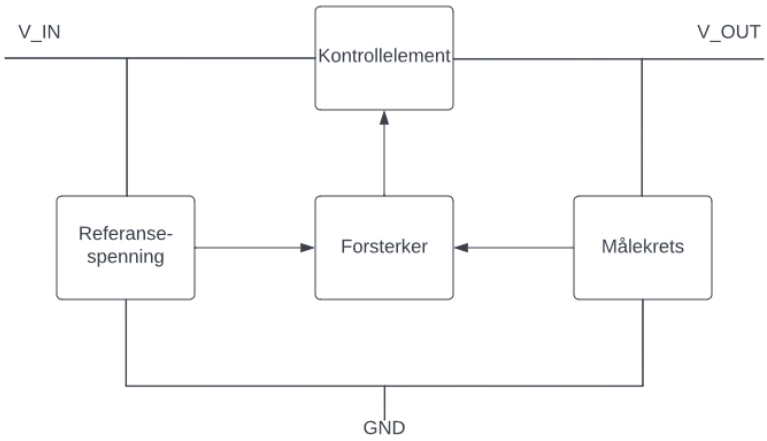
ArduinoJson biblioteket gir en enkel og effektiv måte for å analysere og generere JSON-data. JSON er et format som er utbredt i web- og IoT-applikasjoner, og er utviklet for å være ressurseffektiv og enkel å bruke på mindre enheter som mikrokontollere. Ved å bruke minneallokering på en effektiv måte blir behovet for minne og båndbredde mindre krevende. Biblioteket tilbyr også en rekke funksjoner for å konvertere data mellom JSON-format og andre dataformater som C++ strenger og numeriske typer. [21]

## 2.8 LINEÆR LDO SPENNINGSREGULATOR

En lineær spenningsregulator regulerer utgangsspenningen til en konstant verdi uavhengig av variasjoner i inngangsspenningen og lasten. Det finnes to typer varianter av lineære spenningsregulatorer:

- Faste spenningsregulatorer
- Justerbare spenningsregulatorer

Funksjonen til en lineær spenningsregulator illustrert i figur 2.2 er som følger: Målekretsen består typisk av en spenningsdeling mellom to motstander og er koblet videre til kontrollelementet. Kontrollelementet har er en forsterker som sammenligner spenningen fra referansen og målekretsen. Dette styrer igjen operasjonen til kontrollelementet som kan være en FET transistor. Når utgangspenningen begynner å synke vil forsterkeren sørge for at FET transistorer leder mer strøm og når utgangspenningen øker leder FET transistoren mindre strøm. [22]




*Figur 2.2: Blokkskjema for spenningsregulator, designet i Lucidchart*

En variant av lineære spenningsregulatorer er LDO (Low-dropout) spenningsregulatorer. De kan regulere utgangspenningen selv om inngangsspenningen er veldig nær utgangspenningen. Spenningsfallet over spenningsregulatoren kalles «Dropout» spenning og er oppgitt i databladet. LDO spenningsregulatorer har et spenningsfall over regulatoren som er lavere enn andre spenningsregulatorer. Generell for at LDO spenningsregulatoren skal fungere er å sørge for at:

$$V_{IN} \geq V_{OUT} + V_{Dropout} \tag{2.1}$$

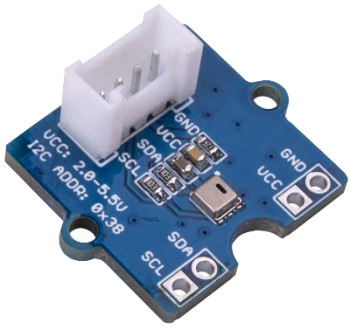
## 2.9 PTE7300 Trykksensor

En trykksensorer er en enhet for måling av trykk, på enten gasser eller væsker. Trykk kan defineres som kraften for å hindre en gass eller væske fra å ekspandere. Det er flere måter for å kunne måle trykk. PTE7300 trykksensoren er en type «gauge pressure» sensor kombinert med en «sealed gauge pressure sensor». En «gauge pressure» sensor måler trykket relativt til det atmosfæriske trykket og en «sealed gauge pressure sensor» måler trykket med en konstant verdi som referansen. Sensoren har innebygd elektronikk for konvertering av de analoge trykkverdiene til digitale $I^2C$ verdier. [23]

## 2.10 Temperatursensorer

### 2.10.1 AHT20

AHT 20 fra Seeed Studio er en temperatur og fuktighetssensor på et ferdig kretskort, med $I^2C$ grensesnitt. Kan måle temperaturer fra -40 °C til 85 °C, med en oppgitt nøyaktighet på ±0,3 °C. Fuktighetsmålingene er fra 0 til 100% relativ fuktighet, med en oppgitt nøyaktighet på ±2 %. [24]



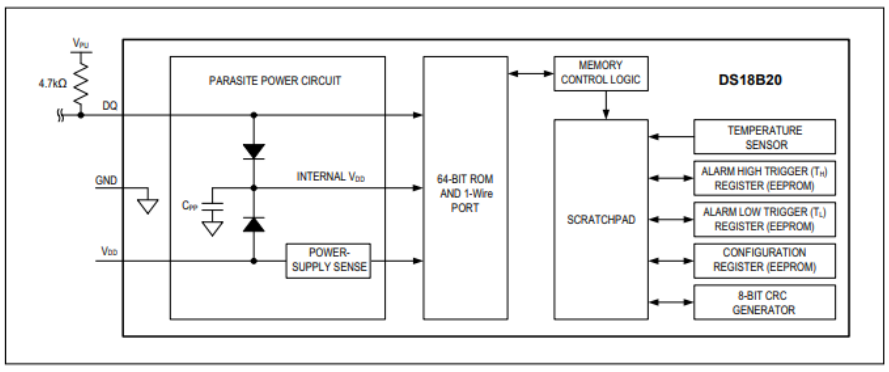

*Figur 2.3: Bilde av temperatursensor AHT20, hentet fra Seeed Studio[5]*

### 2.10.2 DS18B20

DS18B20 er en temperatursensor som er innkapslet i en metallsylindret probe, slik at den er vanntett. Sensoren kan måle fra -55 °C til 125 °C. Dette er en sensor med innebygd ADC. Dette minimerer utfordringer med støy på målingene. Temperatursensorer sender temperaturverdier med 12-bit konfigurerbar oppløsning, hvor alt sendes over signalledning. Kodebiblioteket OneWire må benyttes for å kunne motta data fra temperatursensoren. [3]

*Figur 2.4: Bilde av sensormodulen DS18B20 hentet fra elektroimportøren[4]*

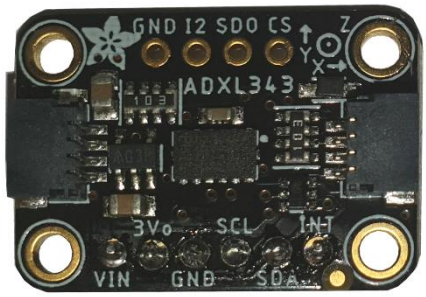
Blokkdiagrammet til temperatursensoren vises i figur 2.5, som viser oppbygningen for konverteringen. Oppgitt nøyaktighet er ±0,5°C. 12-bit gir en oppløsning 0,0625°C som gir lav kilde til feilmåling. Hver sensor har unik adresse som gjør det mulig å koble flere sensorer på samme ledning. En 4,7 kΩ motstand kan kobles imellom $V_{PU}$ og DQ som et alternativ til å koble til $V_{DD}$ ledningen til en strømforsyning. [3]



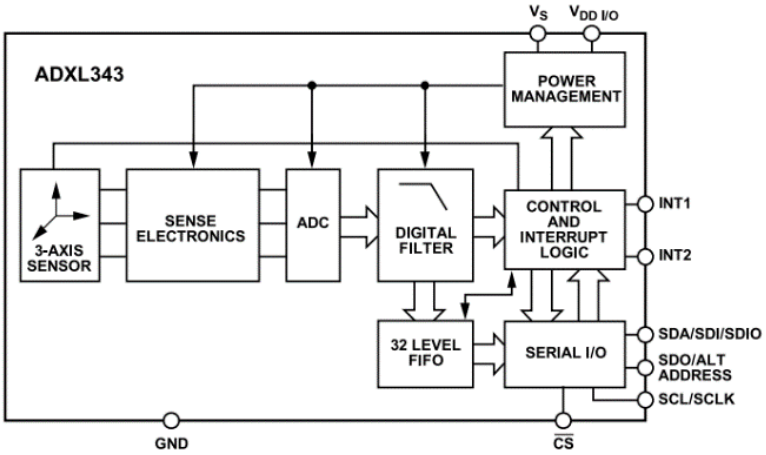

*Figur 2.5: Blokkdiagram for sensor DS18B20, hentet fra datablad [3]*

## 2.11 AKSELEROMETER ADXL343

Adafruit ADXL343 er et kretskort med akselerometer som måler akselerasjon i tre akser. Den er designet for å oppdage både statisk og dynamisk akselerasjon, og egner seg derfor godt for applikasjoner som bevegelses- og tilt-sensorer. Akselerometeret er kompakt, har lavt strømforbruk og høy oppløsning (4096 bit). ADXL343 bruker MEMS-teknologi [25] til å måle akselerasjonen, og har et målområde på +/- 16g. Den kommuniserer over $I^2C$ eller SPI-protokoller og har en innebygd temperatursensor for temperaturkompensasjon. ADXL343 kan benyttes for måling av bevegelse i klokker, stabilisering av kameraer og måling av vibrasjoner i maskiner og konstruksjoner. Eget kodebibliotek for Adafruit ADXL343 må benyttes for å kunne bruke sensoren. [26]




*Figur 2.6: Bilde av ADXL343 vibrasjonssensoren*



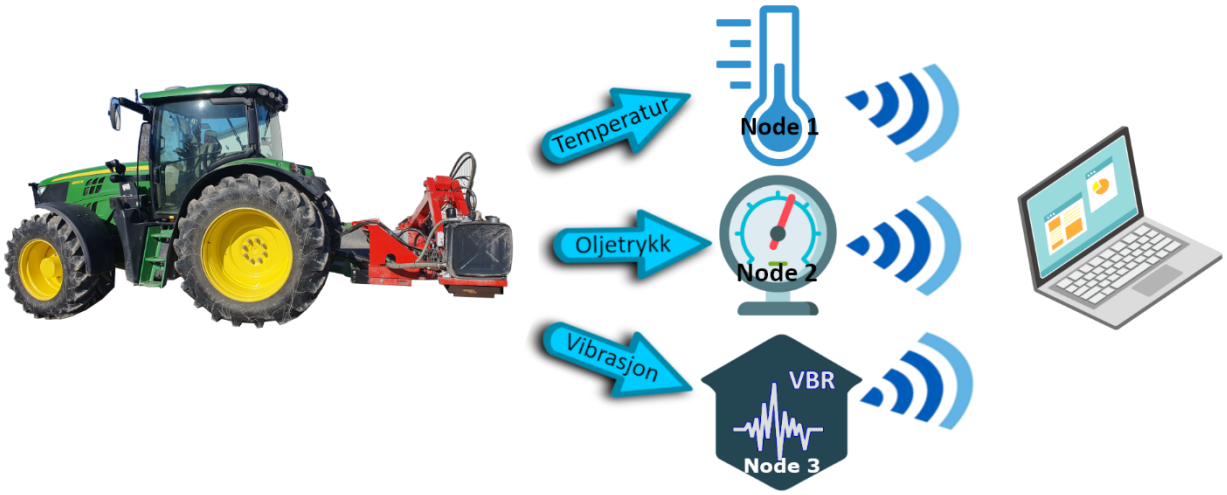

*Figur 2.7: Blokkdiagram for ADXL343, hentet fra datablad [1]*

## 2.12 IP-GRAD

IP-graden til et produkt angir tetthet mot berøring og støv i tillegg til beskyttelse mot vanninntrengning. Den består av to tall hvor det første tallet beskriver graden av beskyttelse mot inntrengning av solide gjenstander/partikler fra 0 til 6, og det andre tallet beskyttelse mot inntrengning av vann fra 0 til 8. Eksempelvis vil en IP-grad på 67 tilsi at produktet er støvtett og tåler kortvarig neddykking i vann. [27]

# 3 METODE

Dette kapittelet omhandler valg og fremgangsmåte som ble brukt for å løse problemstillingen. Kapittelet fremstilles kronologisk med nødvendig tilleggsteori for de forskjellige delene i underkapitellene. Figur 3.1 viser et forenklet oversiktsbilde av systemet.



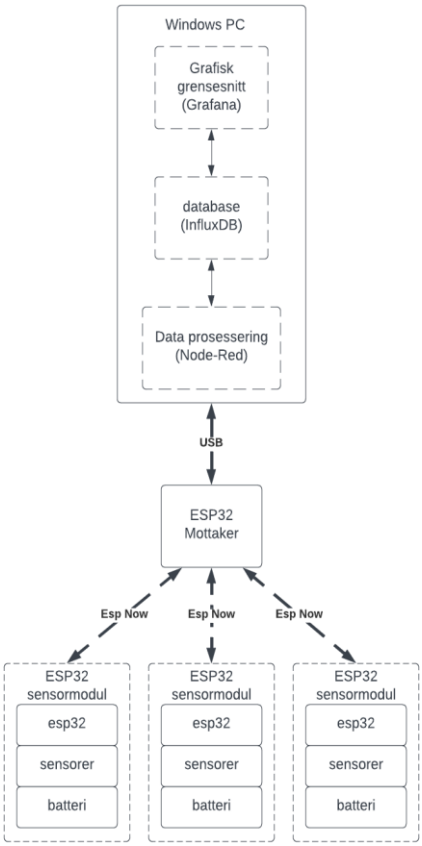

*Figur 3.1: Oversiktsbilde over systemet. Bilde er lagd i GIMP*

## 3.1 OPPBYGNING AV KOMMUNIKASJONSSYSTEMET

### 3.1.1 Valg av kommunikasjonsprotokoll

ESP-NOW brukes som den trådløse kommunikasjonsprotokollen. Protokollen ble valgt ettersom det er en innebygd funksjon i ESP-32 mikrokontrollere. Andre kommunikasjonsprotokoller som MQTT via Wi-Fi og LoRa kunne vært benyttet. Utfordringen med MQTT er at det er avhengig av å være koblet til et Wi-Fi nettverk. LoRa er avhengig av dyre mottakere, i tillegg til at det har en lav båndbredde, men lang rekkevidde. Systemet skal samle inn data fortløpende med flere målinger per sekund over en kort rekkevidde, LoRa er derfor ikke det beste alternativet.

ESP-NOW fungerer kun mellom ESP-32 enheter, derfor er systemet avhengig av å ha en mottaker-node i tillegg til sensor-noder av denne typen. Kommunikasjonen mellom mottakeren noden og serveren foregår over USB. Figur 3.2 viser oppbygningen av systemet for innsamling av sensor-data.




*Figur 3.2: Blokkdiagram for kommunikasjon mellom enheter, designet i Lucidcharts*

## 3.2 HÅNDTERING OG FREMVISNING AV SENSORVERDIER

Systemet for å logge og vise frem sensorverdier består i hovedsak av tre programmer; Node Red, influxDB og Grafana. Disse programmene kjøres på en laptop som Windows-tjenester i bakgrunnen og er satt opp for å starte automatisk ved oppstart.

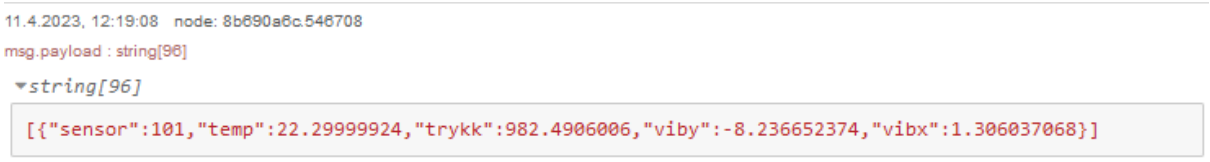
Grensesnittet til programmene er tilgjengelig via nettleser med adressen localhost og nettverksportene til de ulike programmene. Eksempelvis brukes localhost:3000 for å koble til Grafana. Portnummer for Node-red er 1880, og for influxDB er det 8086.

### 3.2.1 Valg av enhet for behandling av data

Det ble undersøkt muligheten for å bruke en Raspberry pi til logging og fremvisning av sensorverdiene, ettersom det er en kraftig liten ettkorts-datamaskin som er relativt enkel å bruke. Programvaren for logging skulle være enkel å bruke og fremvisning av data ble valgt på grunn av enkel bruk og installering med bruk av skriptet IoT-Stack[28] på Raspberry pi. Dette er også gratis programmer som er «open source», og krever ikke abonnement for å kunne brukes, som var et ønske fra oppdragsgiver. På grunn av stor etterspørsel på Raspberry pi ble det vanskelig å få tak i enhet til oppgaven, og det var kun mulig å låne Raspberry pi i en begrenset periode. Løsningen ble å bruke en Windows-PC. Dette gjør også at oppdragsgiver enkelt kan erstatte datamaskinen dersom det skulle være nødvendig. Grafana, Node RED og influxDB er støttede programmer også på Windows PC-er.

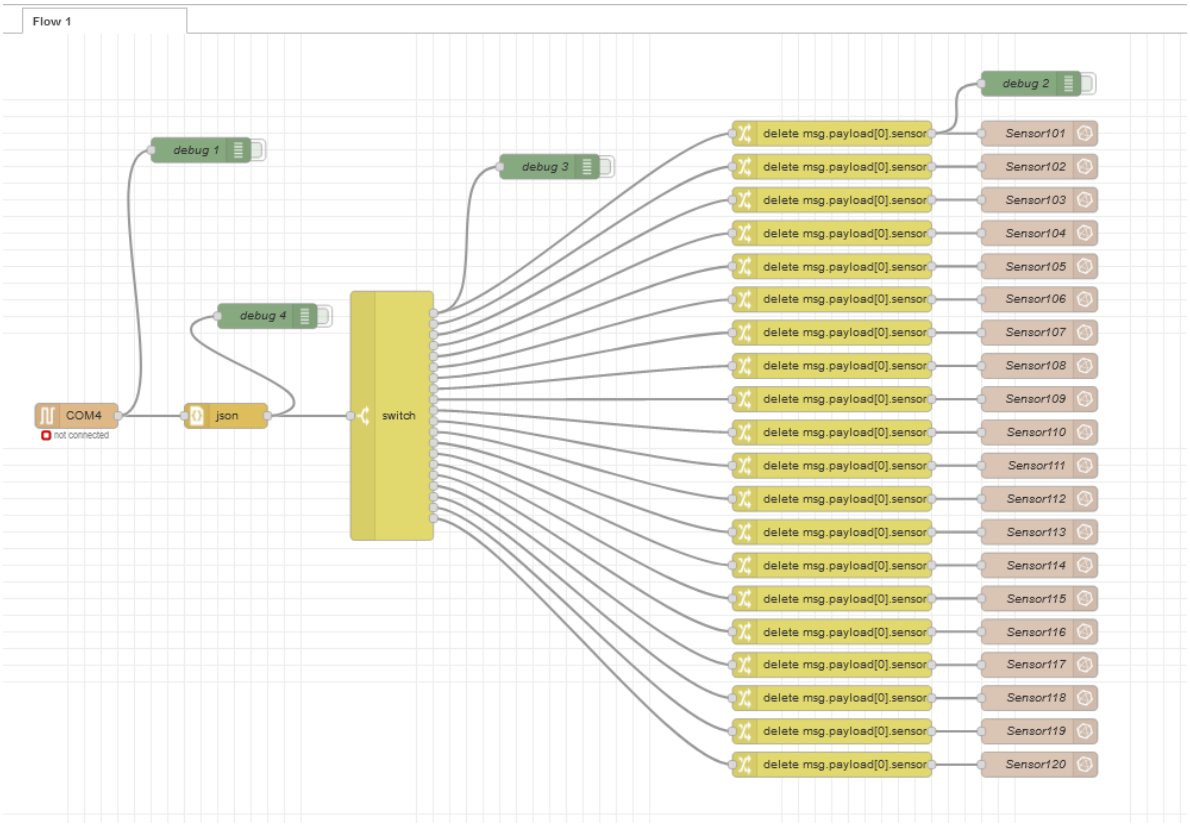
### 3.2.2 Node RED

Node Red blir brukt til å lese sensorverdiene fra mottaker-noden via USB, sortere dem og sende verdiene til riktig lagringsplass i InfluxDB. Sensorverdiene mottas som en Json-string via USB og blir i første steg omgjort til et Json-objekt. Deretter sorteres meldingene etter navn på sensornodene og sendes til riktig lagringsplass i influxDB. Systemet har blitt satt opp slik at det kan ta imot data fra opptil 20 enheter. Figur 3.3 viser et eksempel på en innkommende Json-String, og figur 3.4 viser et Json-objekt etter omgjøring.




*Figur 3.3: Innkommende Json-String*

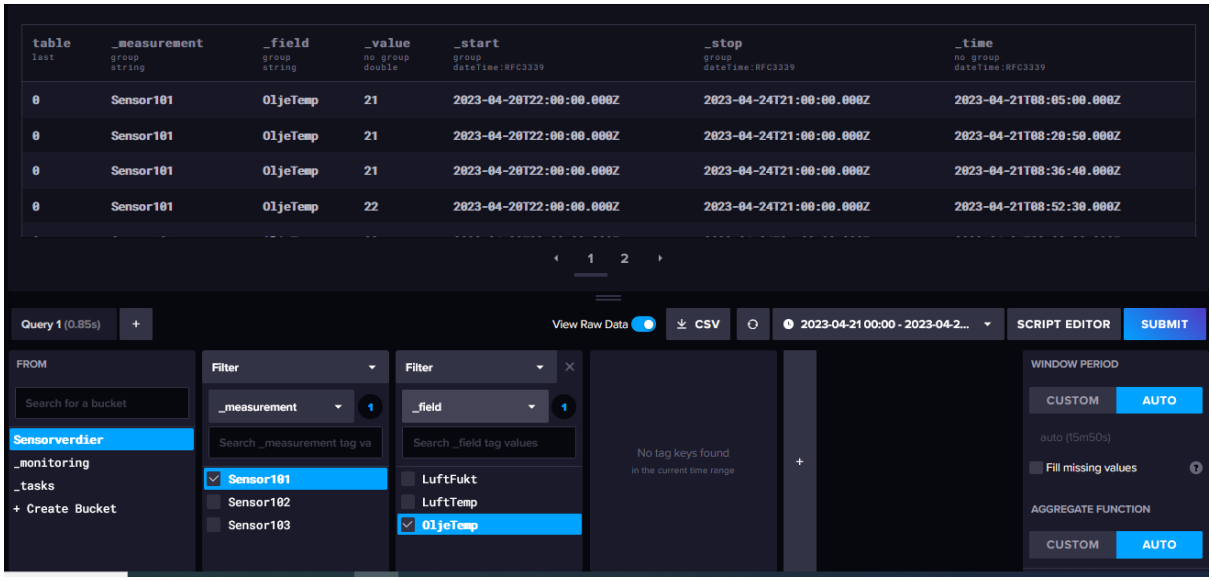

*Figur 3.4: Json-objekt etter formatering*




*Figur 3.5: Skjermbilde av oppsett i Node-Red, med rom for utvidelse av sensornoder*

### 3.2.3 InfluxDB

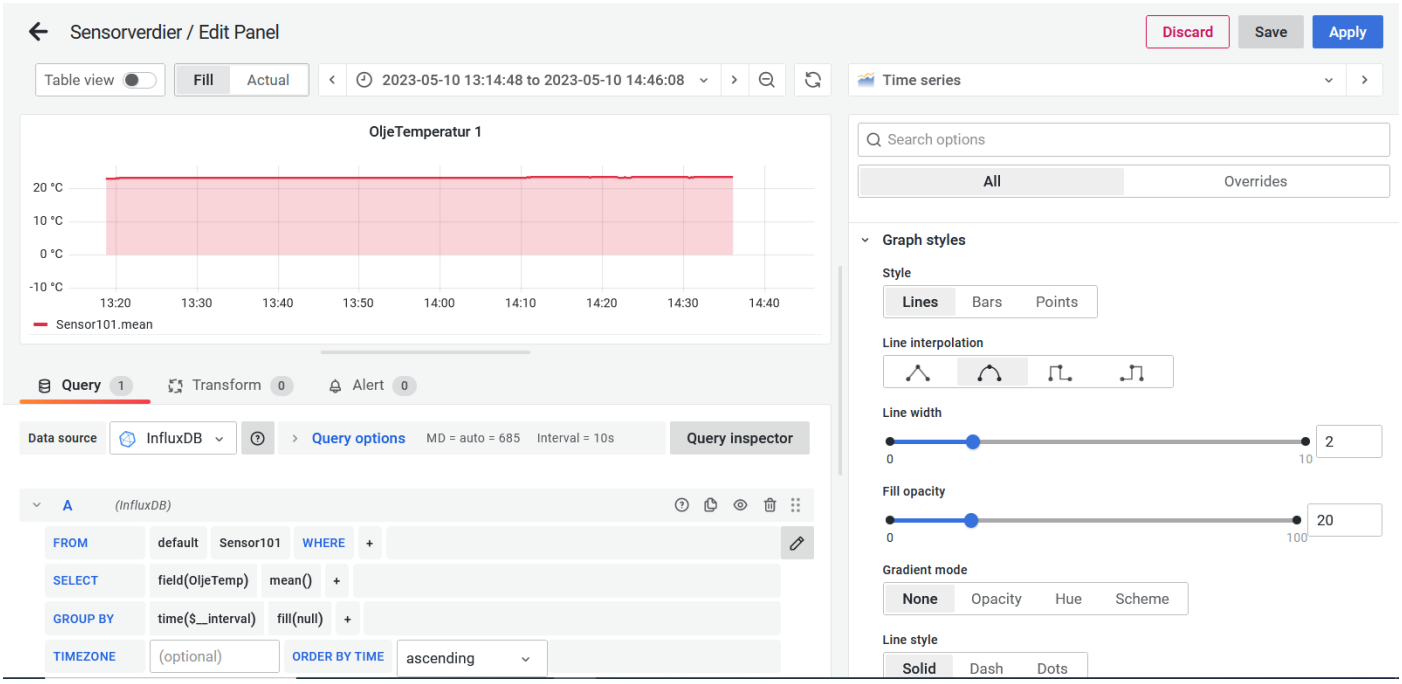
Databasen i InfluxDB er delt opp i «Buckets», «measurements» og «fields». Alle måleverdiene som kommer fra Node-RED blir lastet inn i en «bucket» som heter Sensorverdier. Hver sensornode får sin egen «measurement» og hver enkelt måling fra hver node blir lagret i hver sin «field» sammen med et tidsstempel. I hver «bucket» velger man hvor lenge databasen skal holde på mottatt data. Data er satt til å lagres for alltid, men dette kan endres hvis lagrede data tar for mye plass. Figur 3.6 viser et eksempel på hvordan dataen ser ut i InfluxDB.



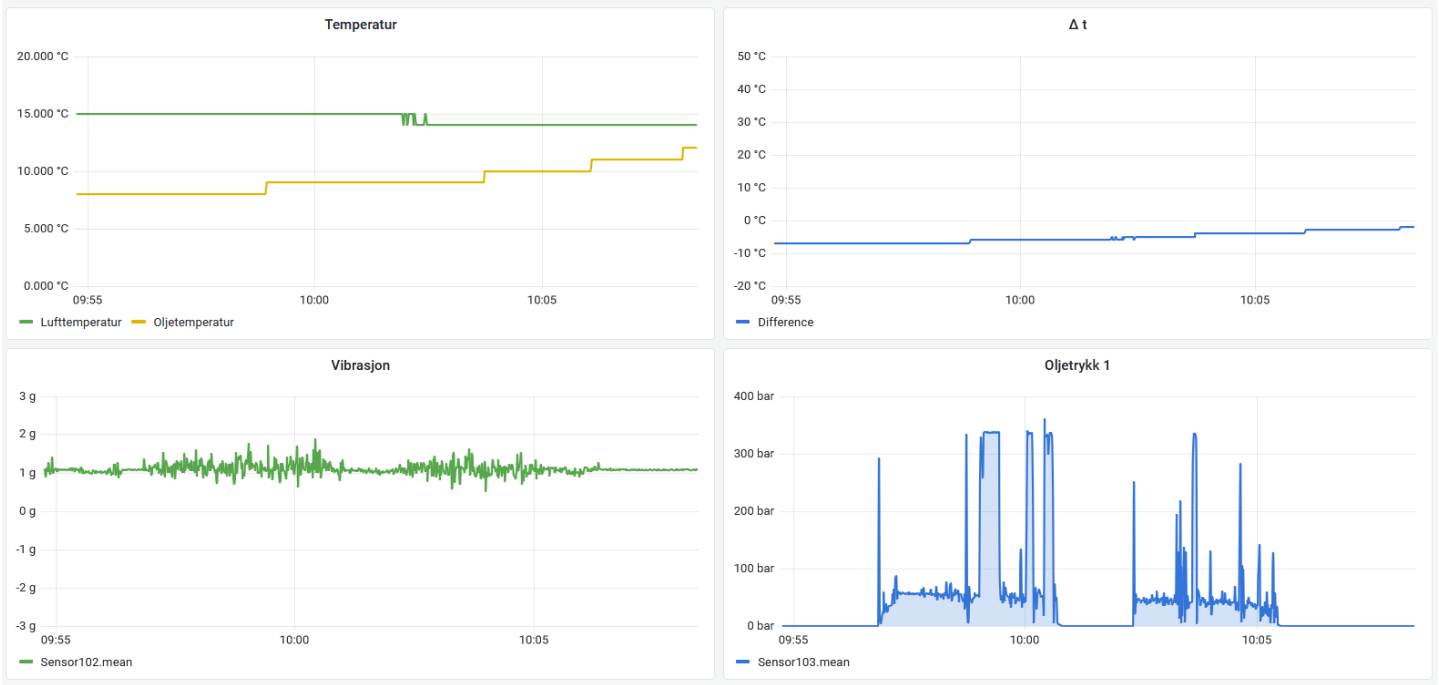

*Figur 3.6: Data lagret i InfluxDB*

### 3.2.4 Grafana

Grafana gir et oversiktlig og lettvint system der man kan lage sitt eget dashbord for å vise data. For å visualisere data lagret i InfluxDB må man først sette det opp som en datakilde, med riktig IP-adresse og en API-nøkkel. Deretter kan man sette opp dashbordet. Figur 3.7 viser hvordan man velger riktig måleverdi og velger hvordan man viser dem frem. På venstre side av bilde ser man hvordan man velger riktig felt fra InfluxDB, her er det valgt Sensor101 i feltet FROM og måleverdien OljeTemp i feltet SELECT. Høyre siden av bildet viser hvordan man vil vise dataen, i dette eksempelet er det valgt en linjegraf. Figur 3.8 viser et eksempel på hvordan dashbordet kan se ut.

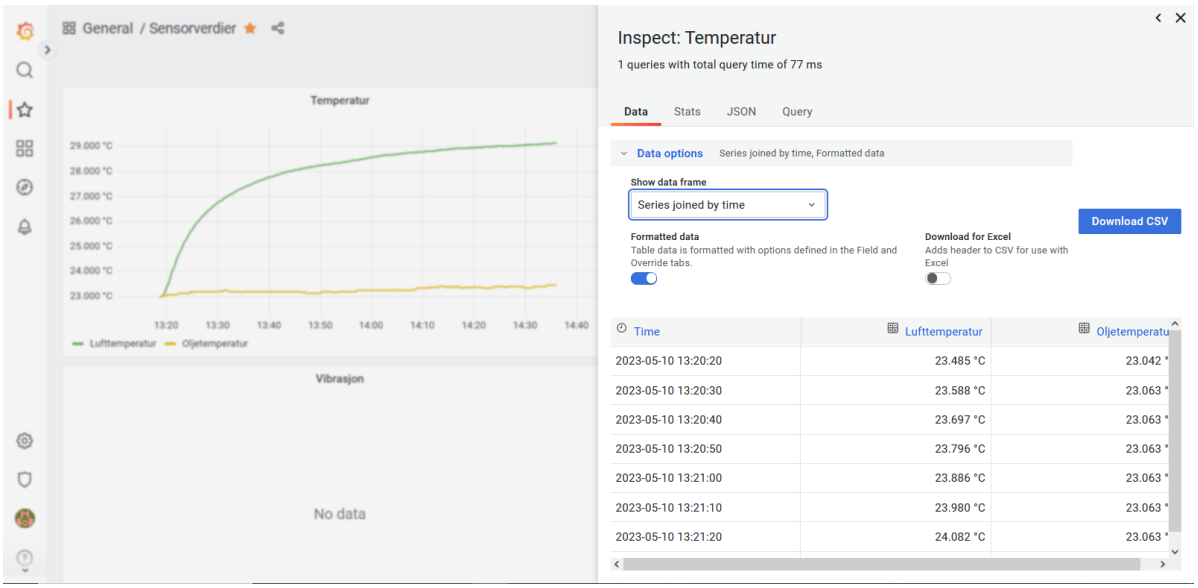



*Figur 3.7: Oppsett av dashbord i Grafana*




*Figur 3.8: Eksempel på dashbord i Grafana*

Data vist i Grafana kan også lastes ned i forskjellige formater. For å hente ut rådata i CSV-format velges ønsket tidsvindu, og trykker deretter på grafen man vil hente ut verdiene fra. Deretter trykker man på feltet «inspect» i dialogboksen som kommer opp. I vinduet som kommer opp, vist i figur 3.9, velges format på filen slik man ønsker.




*Figur 3.9: hente ut data i CSV-format*

### 3.2.5 Kode for ESP-NOW

Hver sensornode er programmert for å kommunisere over ESP-NOW. Koden for hver sensornode ligger som helhet i vedlegg 2, 3 og 4. På linje 10 i figur 3.10 defineres MAC adressen til mottakeren, dette er det eneste som trengs for å sende til riktig enhet ved bruk av ESP-NOW. Devicename som defineres på linje 11 forteller hvilken enhet meldingene sendes fra, i dette tilfellet er det 101, som tilhører sensornode 1. Denne brukes i Node-RED til å sortere de innkommende meldingene.

```
9  // Mottaker MAC adresse og navn
10 uint8_t broadcastAddress[] = {0xE8, 0xDB, 0x84, 0x03, 0xED, 0x7C};
11 #define deviceName (101)
```

*Figur 3.10: Mac adresse*

Figur 3.11 viser loop-funksjonen til sensornode 1. Her pakkes måleverdiene man ønsker å sende inn i et Json-dokument. Dokumentet sendes med ESP-NOW på linje 93 med MAC-adressen definert tidligere.

```
74  void loop() {
75    sensors.requestTemperatures();
76
77    // opprett json dokument
78    StaticJsonDocument <200> doc;
79
80    // Legg til data i json dokument
81    JsonObject obj = doc.createNestedObject();
82    obj["sensor"] = deviceName;
83    obj["LuftTemp"] = (aht20.getTemperature());
84    obj["LuftFukt"] = (aht20.getHumidity());
85    obj["OljeTemp"] = (sensors.getTempCByIndex(0));
86
87    // Konverter json dokument til string
88    serializeJson(doc, data);
89
90    Serial.println(data.c_str());
91
92    // Send data med ESP-NOW
93    esp_now_send(broadcastAddress, (uint8_t *) data.c_str(), data.length());
94
95    // Tøm data
96    data = "";
97    delay(100);
98  }
```

*Figur 3.11: Sende data over ESP-NOW*

## 3.3 Design av kretskortene

### 3.3.1 Batteri og spenningsregulering

Det ble brukt AA-batterier som strømforsyning til de ulike sensornodene. Denne typen strømforsyning har flere fordeler:

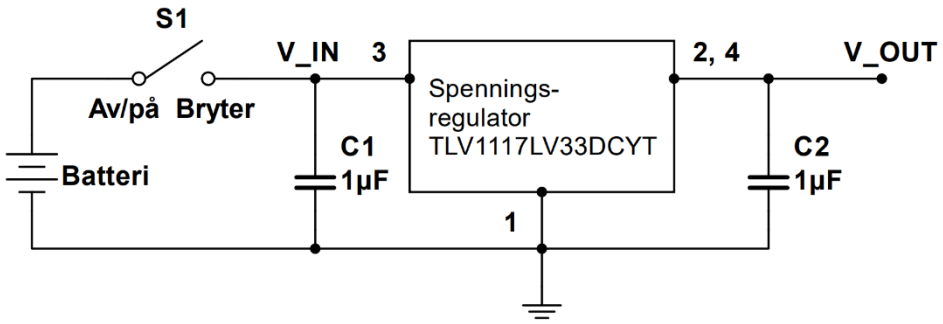
- Mye brukt
- Enkelt å bytte
- Mulig å bruke både engangs og oppladbare batterier
- Brukervennlig

Et AA-batteri har typisk en spenning på 1,5 V når det er fullt. Oppgitt spenningsfall over spenningsregulatoren «Dropout voltage» er oppgitt i datablad [29], og ligger på 230 mV ved et

strømtrekk på 500 mA. Bruker formel 2.1, til å finne spenning $V\_IN_{MIN}$, som blir på 3,53 v. V_IN vil nominelt ligge på 4,5 v, ved bruk av 1,5 v AA-batterier.

Batteriene kobles videre til en av/på bryter med ledninger som loddes fast i kretskortet. Fra bryteren kobles det videre til spenningsregulatoren på kretskortet som på figur 3.12. Spenningsregulatoren må gi ut en fast spenning på 3,3 V for at mikrokontrolleren skal fungere. En 3,3 V fast lineær LDO spenningsregulator av typen «TLV1117LV33DCYT» kan gjøre dette. Ved å bruke en lineær LDO spenningsregulator blir spenningsfallet over regulatoren så lavt som mulig. Fordelen med å bruke en lineær spenningsregulator fremfor svitsjende regulator er at den lineære ikke produserer noe støy. En ulempe er at lineære spenningsregulatorer er mindre effektive og har større varmetap, slik at batteritiden blir noe dårligere.

Tilkoblingen for spenningsregulatoren er likt på alle de produserte kretskortene. Regulatoren gir ut en regulert spenning på 3,3 V. Anbefaling i databladet er å koble en kondensator til jord på både inn og ut spenning [29]. For ekstra stabilitet på utgangen kobles en 1µF keramisk kondensator av typen X7R. Kondensatoren på inngangen til spenningsregulatoren er av samme type og øker ikke stabiliteten, men har andre fordeler som bedre transientrespons og støyavvisning [29]. Den regulerte 3,3 V spenningen fra spenningsregulatoren er forsyningspenningen til mikrokontrolleren og til sensorene som trenger en forsyningsspenning. GND kobles fra negativ terminal på batteriene til alle de ulike GND punktene på en måte som unngår å danne jordsløyfer i kretsen.




*Figur 3.12: Koblingsskjema for spenningsregulatoren, designet i Multisim*

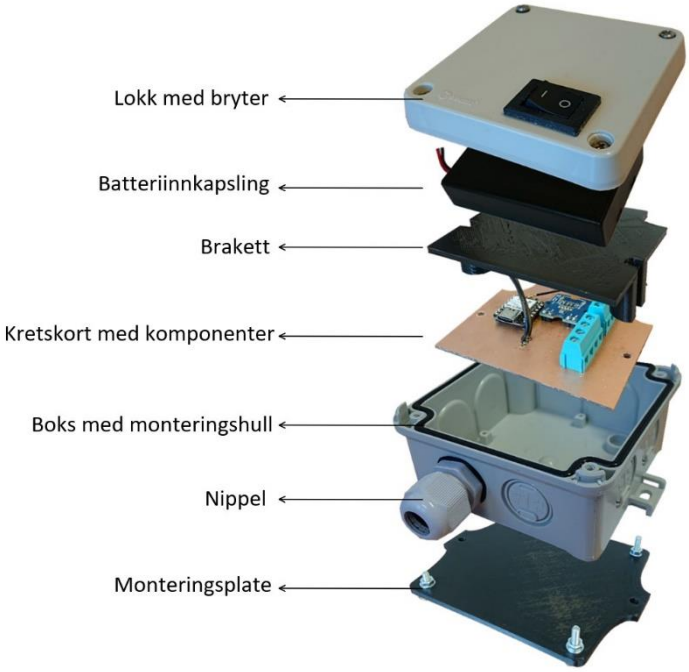
### 3.3.2 Fresing av kretskort

Kretskortet er designet for fresing med fresemaskin. Freser kun ut spor til ledningsbanene og fotavtrykkene for å spare på slitasjen til borene i maskinen. Det er enkelte designvalg som ble tatt for enklere lodding av kretskortet grunnet hvordan kortet produseres, fremfor hva som hadde vært gjort ved bestilling fra fabrikk. Dette for å minimere sannsynligheten for at det skjer feil under

loddeprosessen. Dette inkluderer å ha tykkere ledningsbaner som minimerer risikoen for at de løsner fra kortet under lodding. De tykkere ledningsbanene vil også ha en lavere motstandsverdi. Kretskortet er relativt enkelt og har derfor en god del plass til rådighet.

## 3.4 OPPBYGNING AV SENSORNODENE

Kretskortene er designet for å kunne monteres i en vanntett koblingsboks med en IP-grad på 67. Kabelgjennomføringen for tilkobling av eksterne sensorer gjøres via nippelgjennomføring som vist i figur 3.13. De har en IP-grad på 68 som opprettholder IP karakteristikken på boksen. En av og på bryter med en IP-grad 67 monteres i lokket på boks. Kretskortet festes nederst i boksen med skruer via monteringshullene. En 3D-printet brakett limes på kretskortet for montering av batteriinnkapslingen. AA-batteriene har en egen innkapsling som festes rett på den 3D-printende braketten med borrelås for enkelt bytting av batteriene. Lokket må skrus på godt for å sikre mot vanninntrengning når noden skal tas i bruk. Sensornode-boksen festes direkte på redskapet som det skal måles på, med magneter. Dette gjøres med en 3D-printet monteringsplate med mulighet for å kunne skru fast magneter i hvert hjørne. Oppbygningen av hele sensornoden og alle komponentene er illustrert i figur 3.13.



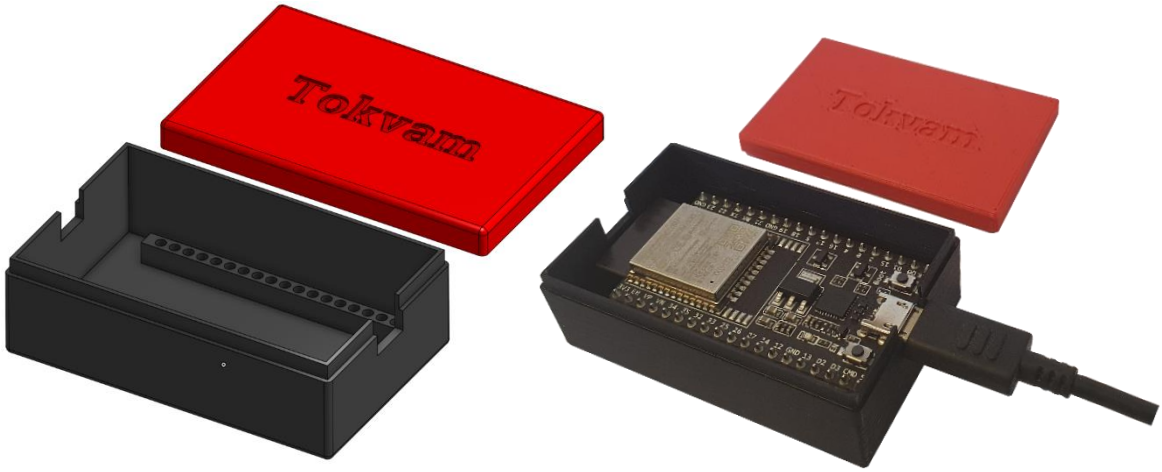

*Figur 3.13: Oppbygning av sensornode 1, 2 og 3 er helt lik. Designet med bruk av Gimp*

## 3.5 MOTTAKER

### 3.5.1 Beskrivelse

Mottakeren beskrevet som ESP32-mottaker i figur 3.2, tar imot trådløs data fra alle sensornodene gjennom ESP-NOW protokollen og videresender informasjonen til datamaskinen for behandling. Den er tilkoblet til PC-en via USB kabel og benytter seg av UART for å sende data til prosessering. Mottakeren består av en ESP-32 mikrokontroller innkapslet i en egendesignet boks.

Boksen er lagd ved 3D-print av plastmaterialet PLA (Polylactic acid), og er designet ved bruk av CAD verktøy. Den er delt i to deler, en boks og et lokk for at mikrokontrolleren skal være lett tilgjengelig ved eventuelle endringer. Boksen har fotfeste for mikrokontrolleren og hull i veggen for USB-kabel. Denne er ikke vannresistant som de resterende komponentene, etterom den er koblet direkte til PCen.




*Figur 3.14: Mottaker designet i CAD til venstre, og ferdig printet med ESP32 innsatt til høyre*

### 3.5.2 Kodebeskrivelse

Hovedjobben til mottakeren er å lese av innkommende meldinger via ESP-NOW og videresende over USB. Kildekoden for mottaker ligger i vedlegg 1. Figur 3.15 viser linje 8-19 i koden. Her defineres en callback-funksjon som blir utført hver gang det mottas en ESP-NOW melding. Data som mottas her er MAC-adressen til senderen, selve meldingen, og lengden på meldingen. I FOR-løkka leses ett-og-ett symbol fra meldingen og legges til stringen «data». Den teller opp helt til den når lengden på meldingen. Når FOR-løkka har lest hele den innkommende meldingen skrives den ut på USB via Serial.println.

```
8  // callback funksjon som aktiveres ved mottatt data
9  void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len) {
10
11   for (int i = 0; i < len; i++) {
12     data += (char)incomingData[i];
13   }
14   //skriver ut mottatt data via USB
15   Serial.println(String(data));
16
17   //tømmer data variabelen
18   data = "";
19 }
```

*Figur 3.15: Utsnitt av koden for mottakeren*

## 3.6 SENSORNODE 1 TEMPERATUR OG FUKTSENSOR

*Figur 3.17: Bilde av sensornode 1*


*Figur 3.16: Mikrokontrolleren SeeedStudio XIAO ESP32C3 Hentet fra seeed studio[2]*

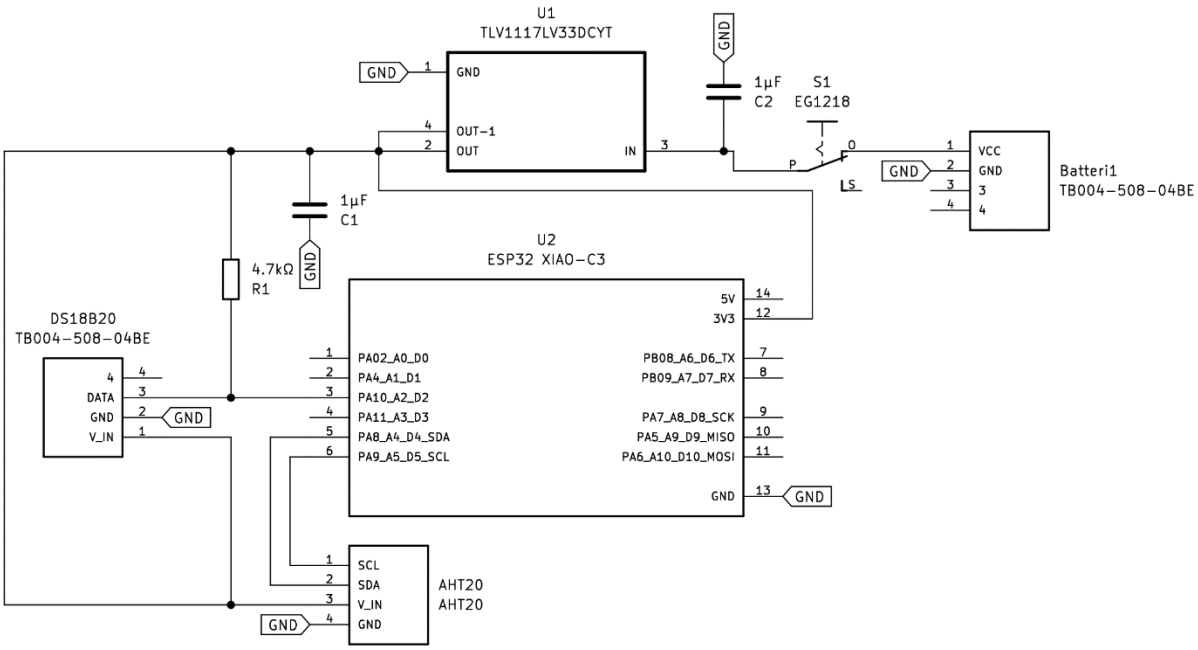
### 3.6.1 Beskrivelse

Første sensornode til testing av trådløs overføring med bruk av ESP-NOW protokollen. ESP32 er tilkoblet to temperatursensorer. Strømforsyningen til kretsen er 3 AA-batterier. Noden inneholder en intern AHT20 sensor, og en ekstern temperatursensor av typen DS18B20.

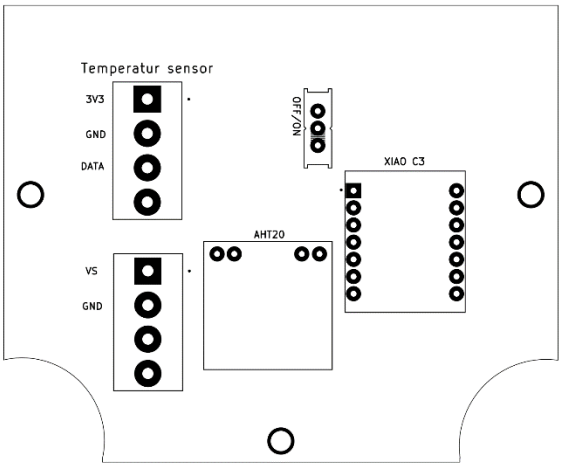
Sensornoden har skruterminaler for tilkobling av batteristrømforsyningen og en ekstern temperatursensor. Skruterminalene gjør det enkelt å bytte ut sensoren med en annen sensor. Samme kretskort kan derfor benyttes til andre sensorer. Dette gjør systemet mer universelt som er et av målene med oppgaven. Mikrokontrolleren som blir brukt er av typen Xiao ESP32-C3. Koblingsskjema vises for kretsen vises i figur 3.18. Utlegg til forsiden og baksiden til kretskortet er designet i KiCad og vises på figur 3.19 og 3.20. Bilder av ferdig sensornode i figur 3.17, og ekstra bilder i vedlegg 5.

### 3.6.1 Seeed Studio XIAO ESP32C3

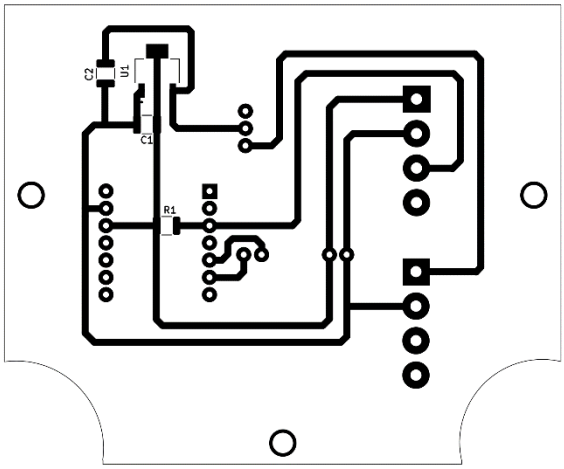
Seeed Studio Xiao ESP32-C3 mikrokontrolleren er utstyrt med ESP32-C3, og har derfor 2,4GHz WiFi 4 og Bluetooth 5.0, se tabell 2.1 for utdypende informasjon. Enheten kan kobles til via USB-C for strømforsyning og seriell kommunikasjon. Seeed Studio XIAO-familien har liten størrelse og ensidig monterte komponenter. Enheten leveres med en ekstern antenne. Det er 11 digitale GPIO pinner som kan brukes som PWM-pinner og 4 analoge som kan brukes som ADC-pinner. Den støtter UART og $I^2C$. [2]




*Figur 3.18: Kretsskjema sensornode 1*




*Figur 3.19: Utlegg for krets til sensornode 1, forsiden.*




*Figur 3.20: Utlegg for krets til sensornode 1, baksiden.*

### 3.6.2 Kodebeskrivelse

Dette er utsnitt av kode til sensornode 1 som ligger i vedlegg 1. Bibliotekene inkludert er:

- Wire for $I^2C$.
- AHT20 for sensoren AHT20.
- OneWire og DallasTemperature for temperatursensoren DS18B20.

Første steg er å initialisere og tilegne begge sensorene navn og buss på linje. Selve oppsettet har behov for få funksjoner. Linje 35, i figur 3.21 starter sensoren med funksjonen begin, mens linje 36 starter $I^2C$ bussen. En if-løkke i linje 38-42 sjekker om AHT20 sensor er koblet til riktig, hvis ikke sendes det en melding i monitoren. Dette har ingen funksjonalitet i praktisk bruk, men hjelper til med eventuell feilsøking.

```
35   sensors.begin();
36   Wire.begin(); //Koble til I2C bus
37   //Sjekk for AHT20 tilkobling
38   if (aht20.begin() == false)
39   {
40     Serial.println("AHT20 not detected. Please check wiring. Freezing.");
41     while (1);
42   }
```

*Figur 3.21: Funksjon for å starte kommunikasjon med sensorene. Utsnitt av kode sensornode 1*

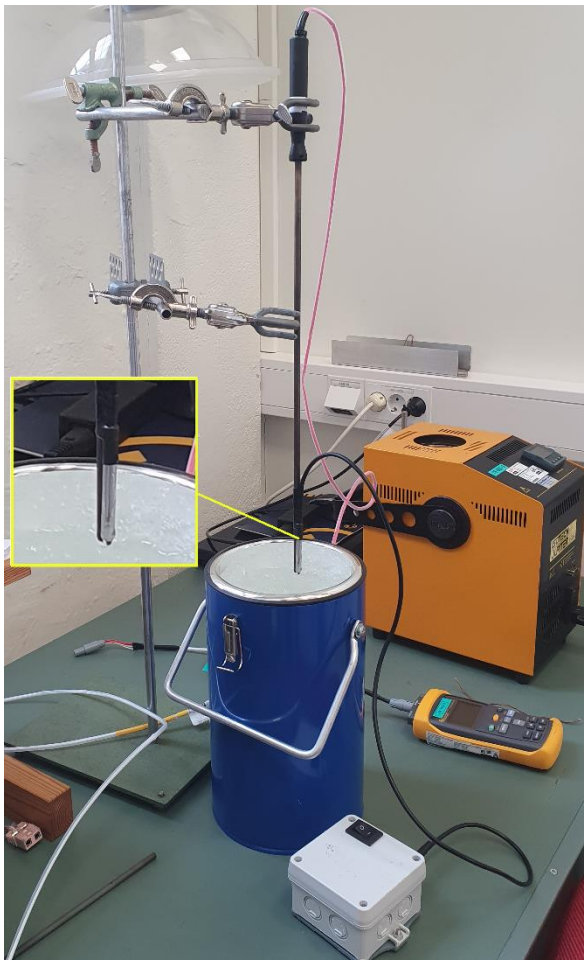
På linje 75 i figur 3.22 ber DS måleren om verdier ved bruk av funksjon requestTemperature(), som er innebygd i DallasTemperature biblioteket. Dette pakkes inn i et JSON dokument for sending. Dokumentet består av 4 objekter med tilhørende navn og verdier. Disse er sensor med navnet til enheten, LuftTemp og LuftFukt med verdiene fra AHT20 sensoren for overflatetemperatur og fuktighet. OljeTemp med temperaturen fra sensoren med kabel. JSON dokumentet sendes med ESP-NOW protokollen over Wi-Fi hvert 100 ms.

```
74  void loop() {
75    sensors.requestTemperatures();
76
77    // opprett json dokument
78    StaticJsonDocument <200> doc;
79
80    // Legg til data i json dokument
81    JsonObject obj = doc.createNestedObject();
82    obj["sensor"] = deviceName;
83    obj["LuftTemp"] = (aht20.getTemperature());
84    obj["LuftFukt"] = (aht20.getHumidity());
85    obj["OljeTemp"] = (sensors.getTempCByIndex(0));
86
87    // Konverter json dokument til string
88    serializeJson(doc, data);
89
90    Serial.println(data.c_str());
91
92    // Send data med ESP-NOW
93    esp_now_send(broadcastAddress, (uint8_t *) data.c_str(), data.length());
94
95    // Tøm data
96    data = "";
97    delay(100);
```

*Figur 3.22: Løkke som mottar verdier fra sensorer og videresender som JSON til mottaker via ESP-NOW*

### 3.6.1 Verifisering av temperatursensor hos IKM Raufoss

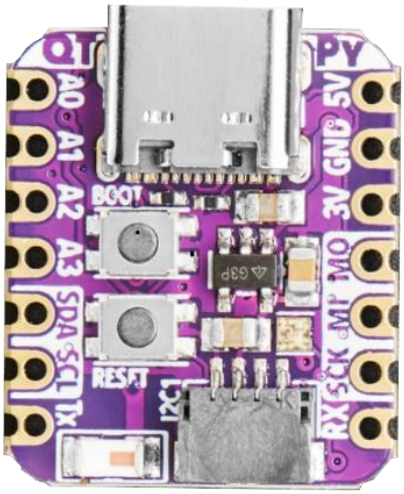
IKM Laboratorium - Raufoss[30] er et kalibreringsfirma som er ISO17025[31] sertifisert. Dette innebærer at de er godkjent som et internasjonalt standard laboratorium, og kan gi sertifikater og attestester. Verifiseringen av sensoren DB18B20 ble utført fra 0 til 100°C ved bruk av isbad og ovn. Referansetemperatur ble målt med et Fluke 1523 kalibreringstermometer med en PT-100 sensor, som har en feilmargin på ±0,011°C[32]. Resultatene ble sammenlignet ved øvre og nedre grense, som vist i kapittel 4.4.1.



*Figur 3.23: Kalibrering av temperatursensor på IKM Laboratorium. PT-100 og DS18B20 vist med gul ramme.*

## 3.7 SENSORNODE 2 OLJETRYKKSENSOR

Figur 3.25: Bilde av sensornode 2




Figur 3.24: Mikrokontrolleren AdaFruit QT PY

### 3.7.1 Beskrivelse

Mikrokontrolleren i sensornode 2 er av typen QT PY ESP-32-S2 WIFI Dev Board. Tilkobling fra sensornoden til oljetrykksensoren er en M12 5-pinn skrutilkobling. Med bruk av en 0,5 m lang kabel fra oljetrykksensoren til ESP-32 noden, gjør plasseringen av noden fleksibel i forhold til tilkoblingspunkt for oljetrykkmåling. Internt i noden kobles ledningene på skruterminalkobingen for $I^2C$ sensorer. M-12 skrutilkoblingen har en IP-grad på 67 som opprettholder kapslingsgraden.

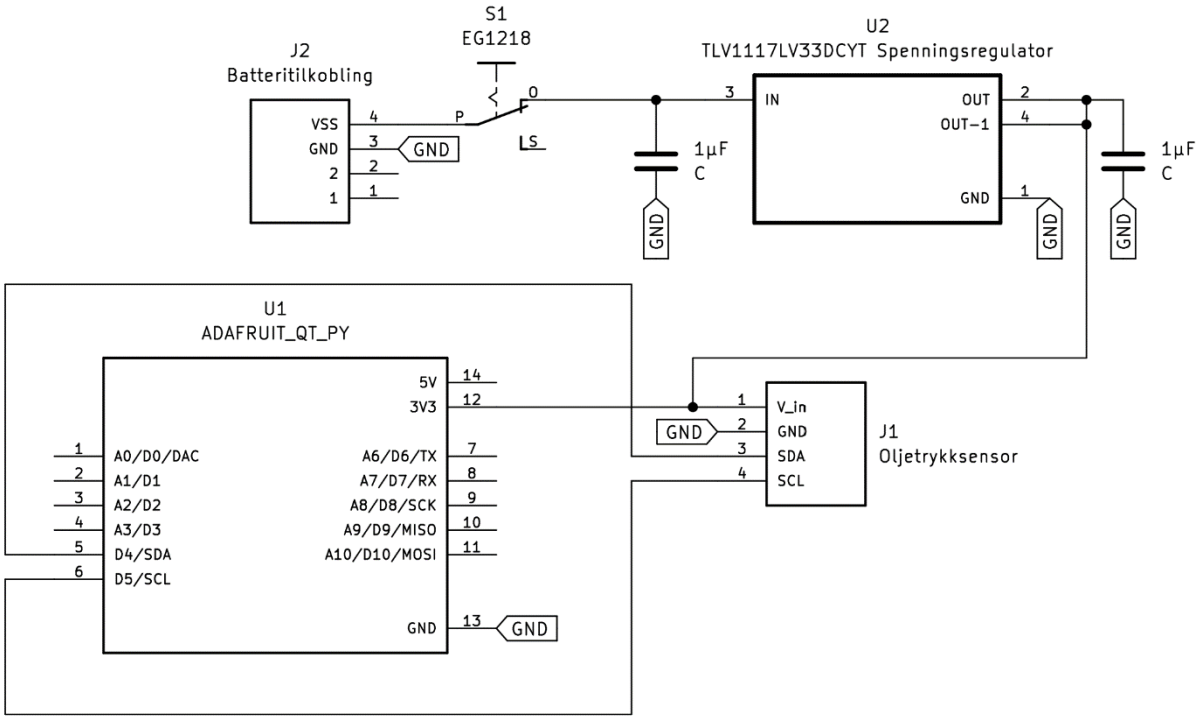
Oljetrykksensoren har innebygd elektronikk som gjør om de analoge måleverdiene til digitale $I^2C$ verdier før sending til mikrokontrolleren.

Det er gjort forbedringer sammenlignet med sensornode 1. Skruterminalene er flyttet til enden av kretskortet for å ha tilgang til skruterminalene når braketten til batteriene er limt fast i kortet. Det ble frest på skrift i kobberplaten for å merke skruterminalene, som øker brukervennligheten. Designet av utlegget er gjort i KiCad og vises i figur 3.27 og 3.28. Koblingsskjema for sensornode 2 vises i figur 3.26. Bilder av ferdig sensornode i figur 3.25, og ekstra bilder i vedlegg 6.

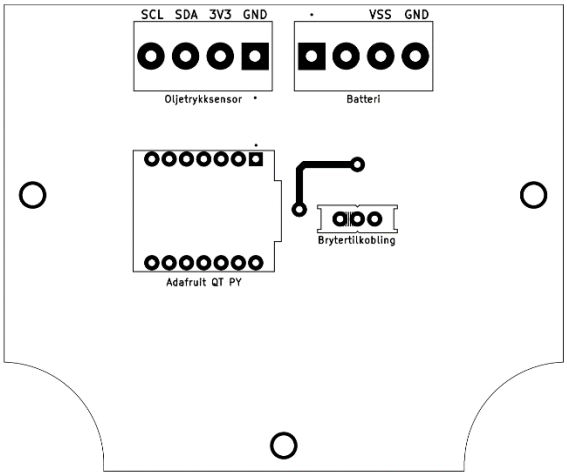
### 3.7.2 AdaFruit QT PY

Adafruit QT Py [33] er designet for å ha en liten fysisk størrelse. Den er bruker ESP32-S2 mikrokontrolleren som støtter 2,4 GHz Wi-Fi 4, se tabell 2.1 for utdypende informasjon. Tilkobling via USB-C for strømforsyning og seriell kommunikasjon.

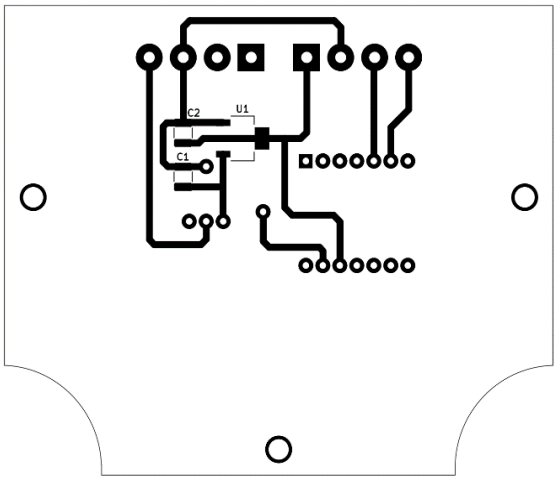
Fotavtrykket og størrelse er lik Xiao mikrokontrolleren beskrevet i kapittel 3.6.1. De kan derfor enkelt byttes om hverandre. Unikt med denne mikrokontrolleren er at $I^2C$ kan benyttes med egen kobling og ledning istedenfor SDA og SCL pinnene.




*Figur 3.26: Kretsskjema for sensornode 2.*




*Figur 3.28: Utlegg for krets til sensornode 2, forsiden.*




*Figur 3.27: Utlegg for krets til sensornode 2, baksiden.*

### 3.7.3 Kodebeskrivelse

Dette er utsnitt av kode til sensornode 2 som ligger i vedlegg 2. Unikt bibliotek for denne koden tilhører selve oljetrykksensoren, som ble tilsendt fra produsent, PTE7300_I2C. Funksjonene i dette biblioteket tilkalles ved uttrykket mySensor, med medfølgende kommando. Først tilkobles sensoren i linje 12. I linje 13-15 deklareres verdier for DSP som er navnet til sensoren i biblioteket. DSP_S er trykk, mens DSP_T er temperatur. Temperaturen er en verdi som brukes internt for kalkulasjonen av trykket, så verdien blir ikke brukt til logging.

```
12 PTE7300_I2C mySensor; // Tilkoble sensor
13 int16_t DSP_S;
14 int16_t DSP_T;
15 int DSP_T_int;
```

*Figur 3.29: Initialisering av sensor med gitte variabler, utsnitt fra kode*

Sensoren blir bedt om å benytte CRC (cyclic redundancy check) som bekrefter at den er koblet til og sender riktige verdier. Dette er kun for feilsøking og kan brukes i monitoren når tilkoblet Arduino IDE.

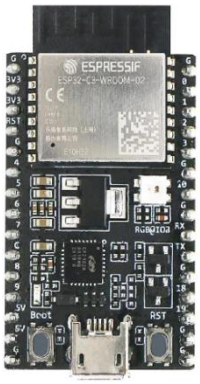
På linje 64 leses verdien for trykket fra sensoren via $I^2C$. På linje 66 blir denne verdien gjort om fra ±16000 som er verdiene definert i databladet [34], til verdiene 0 til 400 bar. Map-funksjonen er innebygd i Arduino IDE og gjør kalkulasjonene som fordeler de 32000 verdiene lineært i området 0-400 for å fremvise ønskede verdier. Dermed blir dette pakket inn i et JSON dokument for trådløs sending til mottakeren. I pakken er det 2 variabler, sensornavn og trykk med etikett «Pressure», og gjentas hvert 100 ms.

```
61
62 void loop() {
63   mySensor.CRC(true);
64   DSP_S = mySensor.readDSP_S();
65   //gjør om fra int16 til bar
66   int mappedBar = map(DSP_S, -16000, 16000, 0, 400);
67
68   //opprett json dokument
69   StaticJsonDocument <200> doc;
70
71   //legg til data i json dokument
72   JsonObject obj = doc.createNestedObject();
73   obj["sensor"] = deviceName;
74   obj["Pressure"] = (mappedBar);
75   serializeJson(doc, data);
76
77   Serial.println(data.c_str());
78
79   // Send data med ESP-NOW
80   esp_now_send(broadcastAddress, (uint8_t *) data.c_str(), data.length());
81
82   //tøm data
83   data = "";
84   delay(100);
85 }
```

*Figur 3.30: Løkke i sensornode 2 som mottar trykkverdier fra sensor, pakker inn i JSON for videresending med ESP-NOW til mottaker.*

## 3.8 SENSORNODE 3 AKSELEROMETER

*Figur 3.32: Bilde av sensornode 3*



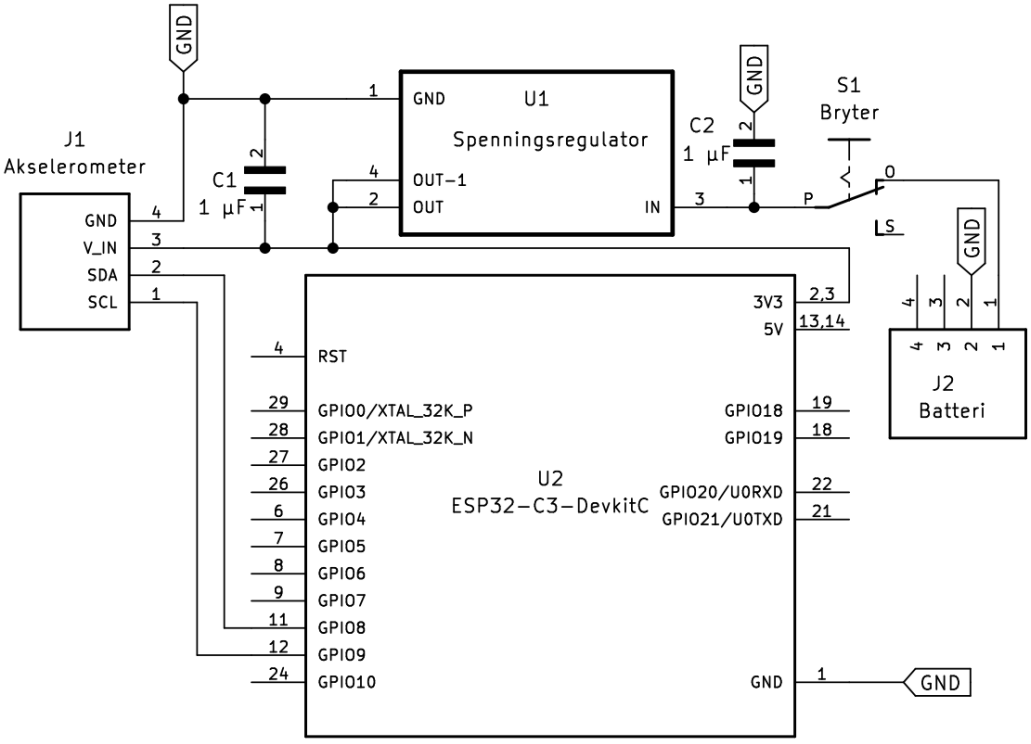

*Figur 3.31: Espressif ESP32-C3-DevkitC modul*

### 3.8.1 Beskrivelse

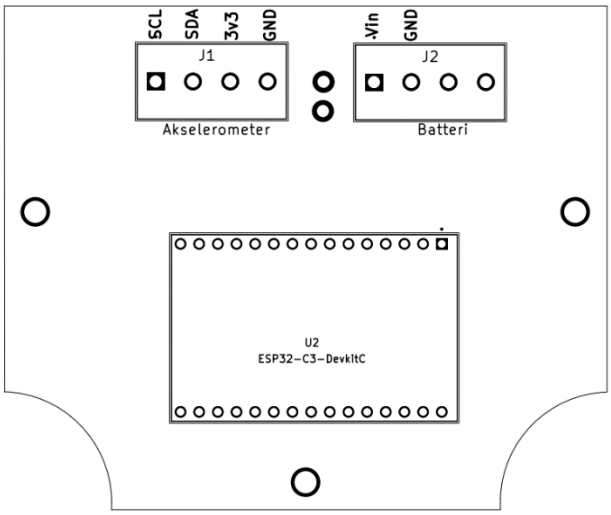
Sensornode 3 består av mikrokontrolleren ESP32-C3-DevkitC. Sensornoden har lik konstruksjon som de andre nodene og er koblet til batterier og spenningsregulator som beskrevet i kapittel 3.3. Fra mikrokontrolleren er det tilkoblet til GPIO 8 og 9 for tilsvarende SDA og SCL for $I^2C$ kommunikasjon med akselerometeret ADXL343. To skruterminaler som vist i kretsskjema og utlegg gjør av og tilkobling av sensor og batteri forenklet. Utlegg til forsiden og baksiden til kretskortet er designet i KiCad og vises på figur 3.34 og 3.35. Bilder av ferdig sensornode i figur 3.32, og ekstra bilder i vedlegg 7.

### 3.8.2 ESP32-C3-DevkitC

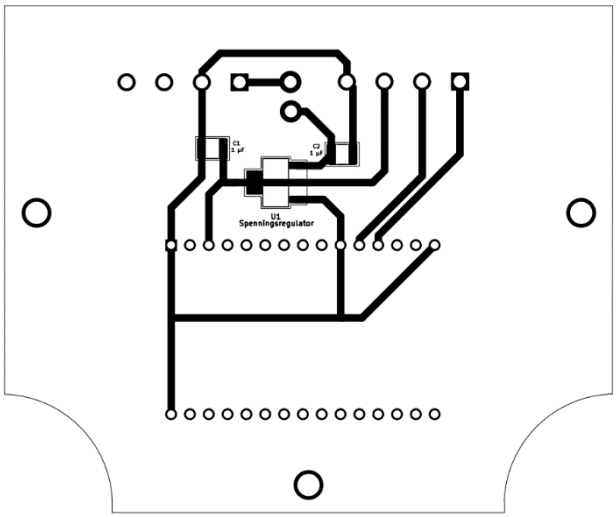
ESP32-C3-DevkitC [35] er en mikrokontrollerenhet laget av Espressif som bruker ESP32-C3. Har tilkoblingsmulighet til 2,4 GHz Wi-Fi 4 og Bluetooth 5.0, se tabell 2.1 for utdypende informasjon. Tilkobling via micro-USB for strømforsyning og seriell kommunikasjon. Den fysiske størrelsen er større sammenlignet med mikrokontrollerne fra sensornode 1 og 2. Fordelen er flere tilgjengelige pinner for tilkobling av komponenter.




*Figur 3.33: Kretsskjema for sensornode 3*



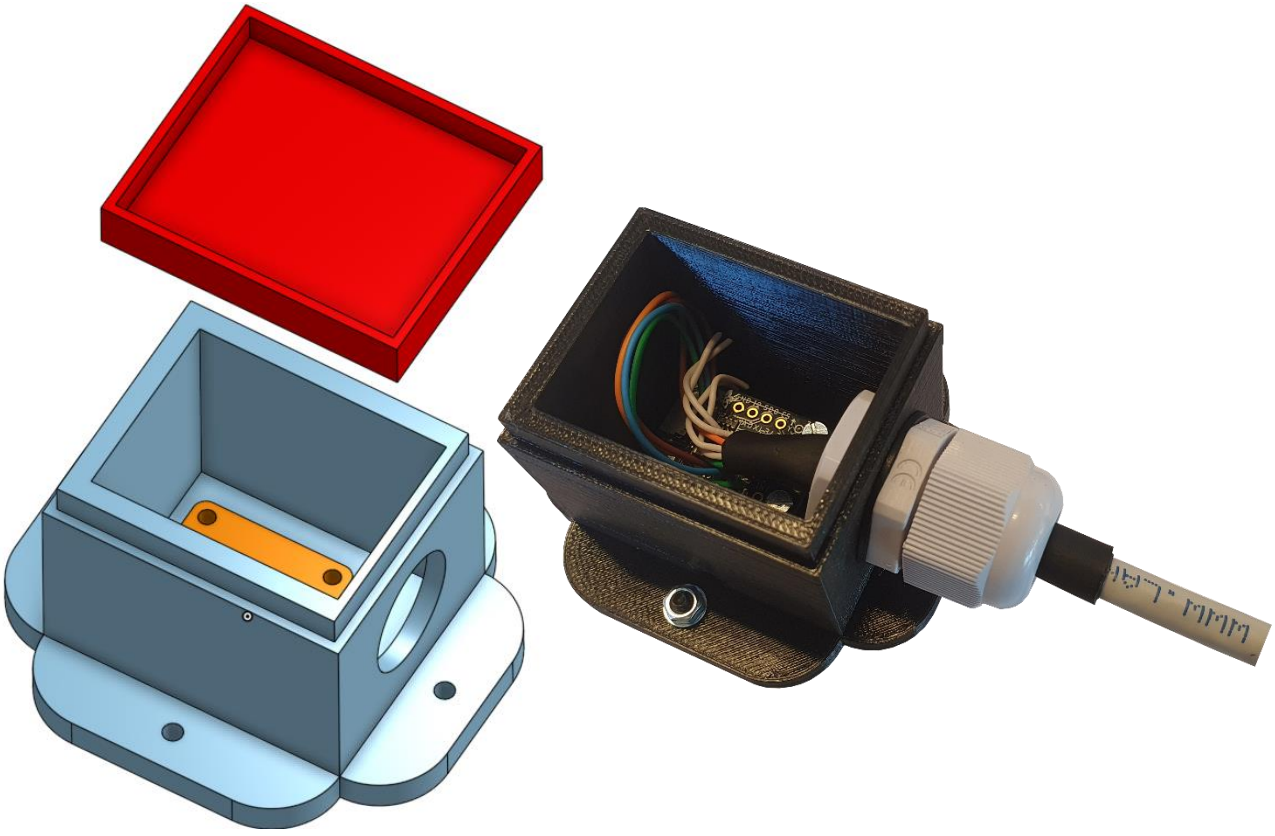

*Figur 3.35: Utlegg fremside kretskort sensornode 3*



*Figur 3.34: Utlegg bakside kretskort sensornode 3*

### 3.8.3 Innkapsling av vibrasjonssensor

For tilkobling av vibrasjonssensoren fra sensornoden er det benyttet en CAT6 Ethernet kabel som går til en 3D-printet boks, som vist i figur 3.36. Det ble benyttet en CAT6 kabel siden den har nok ledere, og er vesentlig rimeligere enn en M12 kabel. Den 3D-printete boksen er av plasttypen PETG (Polyethylene terephthalate glycol). Denne typen plast er i prinsippet vanntett og skal holde seg lengere enn PLA i vær og vann. Vibrasjonssensoren er skrudd fast i boksen, så vibrasjoner kan registreres mer nøyaktig. Det er fire hull i platen rundt boksen for montering av neodym magneter. Dette er de samme magnetene brukt under koblingsboksene, og med bruk av 4 stykk tilsvarer dette en holdekraft på 3.2 kg. Dette gir nok holdekraft til at boksen skal holde seg fast og følge vibrasjonen til utstyr som testes.



*Figur 3.36: Boks med lokk. Utlegg for 3d-print til venstre, og ferdig montert uten lokk til høyre*

### 3.8.4 Kodebeskrivelse

Dette er utsnitt av koden til sensornode 3 som ligger i vedlegg 4. Bibliotekene brukt til denne sensornoden er: Wire for $I^2C$. Adafruit_Sensor og Adafruit_ADXL343 for funksjoner som kommer med akselerometer-modulen. SDA og SCL portene for $I^2C$ blir definert på linje 10 og 11. Linje 14 definerer MS2_TO_G som konverterer verdiene fra akselerasjon til g-krefter. Dette er fordi vibrasjonssensoren måles i $m/s^2$, men g-kraft er mer universalt og gir en fremstilling som er enklere å forstå. På linje 15-17 deklareres verdiene som double variabel, for de har flere desimaler inkludert enn float.

```
10 #define ADXL343_SDA 8
11 #define ADXL343_SCL 9
12
13 //Definerer faktor for konvertering fra m/s^2 til g
14 #define MS2_TO_G 0.101971621
15 double VibrX = 0;
16 double VibrY = 0;
17 double VibrZ = 0;
```

*Figur 3.37: definering av porter og variabler*

I setup på linje 43 settes ønsket måleområde til 16 g med funksjoner i ADXL343 biblioteket. Uttrykket accel. brukes for å kalle på ønsket funksjon. På linje 44 startes akselerometeret med funksjonen begin.

```
42   // Init ADXL343 og sett måleområde til 16g
43   accel.setRange(ADXL343_RANGE_16_G);
44   accel.begin();
```

*Figur 3.38: setup for akselerometeret*

I løkken blir sensordataen tilkalt gjennom SDA med getEvent og lagret til variablene deklarert for hver akse. På Linje 85-87 foretas multiplikasjonen for konvertering til g. På linje 90 omregnes det til totale g-kraften. Dette kan gjøres ved å benytte pythagoras' theorem som vist i ligning 3.1.

```
78 void loop()
79 {
80   // Leser akselerasjon
81   sensors_event_t event;
82   accel.getEvent(&event);
83
84   // Konverterer akselerasjon fra m/s^2 til g
85   VibrX=event.acceleration.x*MS2_TO_G;
86   VibrY=event.acceleration.y*MS2_TO_G;
87   VibrZ=event.acceleration.z*MS2_TO_G;
88
89   // Beregner total akselerasjon
90   double Vibrasjon = sqrt(VibrX * VibrX + VibrY * VibrY + VibrZ * VibrZ);
```

*Figur 3.39: bilde av løkken som samler data fra akselerometeret*

$$Vibrasjon = \sqrt{VibrX^2 + VibrY^2 + VibrZ^2} \quad (3.1)$$

Variablene som blir sendt trådløst til mottaker er sensornavn, vibrasjon fra hver akse og total g-kraft med navn «vibrasjon». Dataen blir pakket inn i et Json dokument og sendt via ESP-NOW protokollen, vist i figur 3.40.

```
92   // opretter json dokument
93   StaticJsonDocument <200> doc;
94
95   // legger til data i json dokument
96   JsonObject obj = doc.createNestedObject();
97   obj["sensor"] = deviceName;
98   obj["Vibrasjon X"] = VibrX;
99   obj["Vibrasjon Y"] = VibrY;
100  obj["Vibrasjon Z"] = VibrZ;
101  obj["Vibrasjon"] = Vibrasjon;
102
103  // konverterer json dokument til string
104  serializeJson(doc, data);
105
106  Serial.println(data.c_str());
```

*Figur 3.40: Sending av vibrasjonsdata over ESP-NOW*

### 3.8.5 Kalibrering av akselerometer

Det ble gjennomført kalibrering av akselerometer i stasjonær tilstand for å deretter kontrollere måleverdiene. Akselerometeret gir ut akselerasjon i 3 akser. Plasserer akselerometret i tre ulike retninger, for å kontrollere at en av aksene gir ut en akselerasjon på 9,81 m/s$^2$ samtidig som de to andre gir ut 0 m/s$^2$. I tillegg blir det kontrollert at g-kreftene i stasjonær tilstanden er 1, uavhengig av retning på sensoren. Dette vil bekrefte om sensoren måler riktig i stasjonær tilstand.

## 3.9 SENSORNODE 4

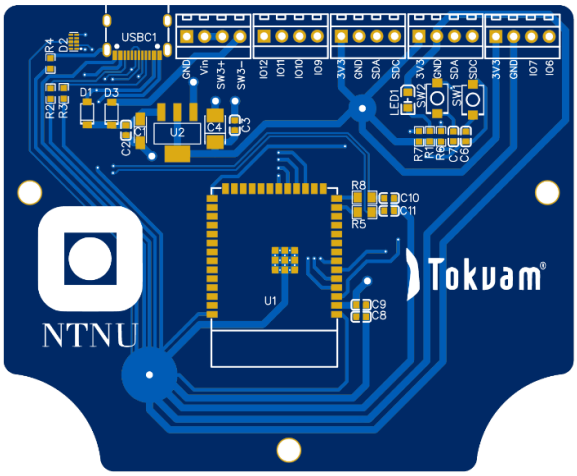
Det ble i tillegg designet en sensornode basert på mikrokontrolleren ESP32-S3-WROOM-1. Dette var en videreutvikling av kretskortene produsert med fresemaskin. Denne noden var tenkt som en modulær plattform som skulle kunne kobles til ulike sensorer. Utlegget ble designet i programvaren Easy EDA[36] og med mulighet for å få det ferdig produsert og loddet fra JLCPCB.

ESP32-S3-WROOM-1 er en mikrokontrollerenhet utviklet av Espressif. Den bruker ESP-32-S3 mikrokontrolleren og har 2,4GHz WiFi 4 og Bluetooth 5.0 [37], se tabell 2.1 for utdypende informasjon.

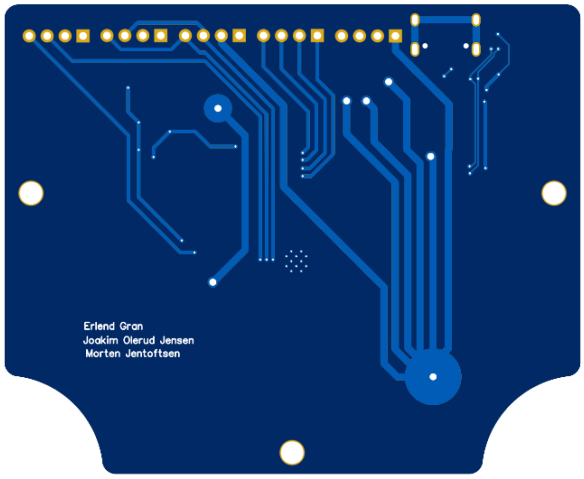

*Figur 3.41: Bilde av ESP32-S3*

Det ble valgt å bruke ESP32 S-serien, siden denne har innebygd USB-OTG, og S3 ble valgt over S2 på grunn av innebygd Bluetooth funksjonalitet. Hovedforskjellen mellom dette designet og kretskortene fra sensornode 1, 2 og 3 er at den er designet ut fra en ESP32-WROOM, som i hovedsak kun inneholder selve mikrokontrolleren. I de tidligere sensornodene ble det brukt ESP32 Devkit kort, som er ferdig kretskort med ESP32 mikrokontroller koblet til USB tilkobling, strømforsyning og ulike utgangspinner.

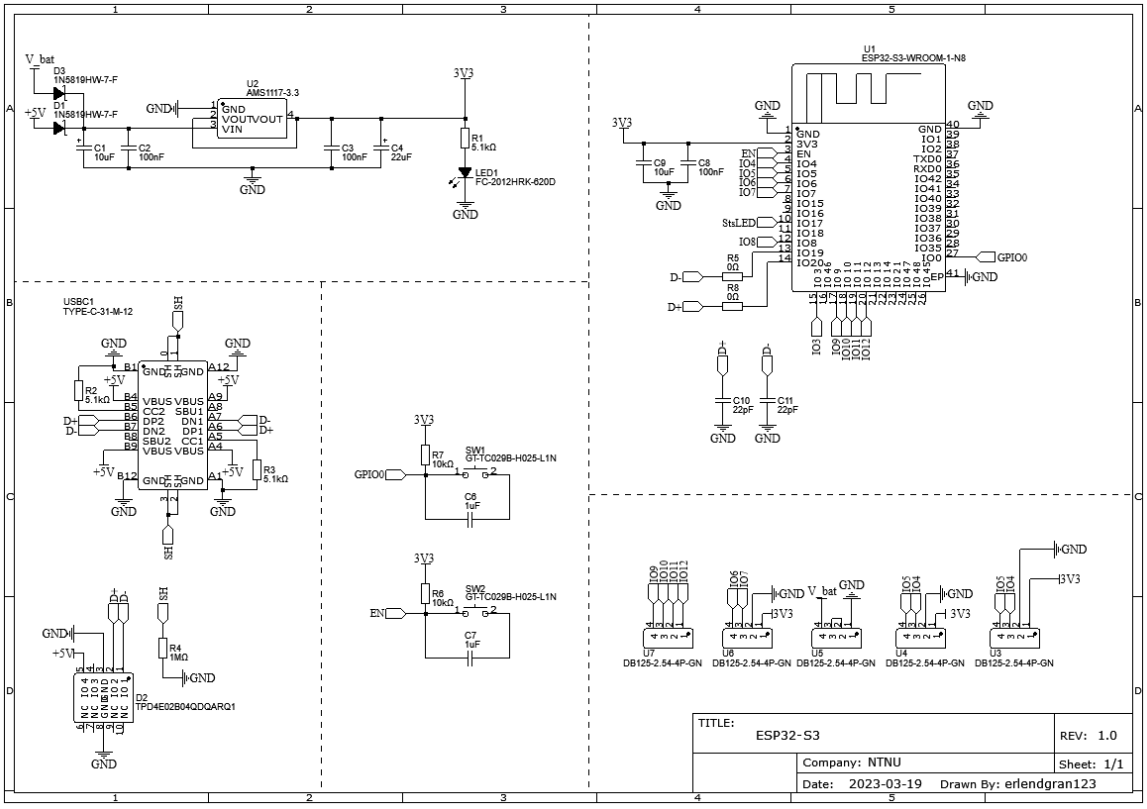
Hovedfordelen med denne noden er at oppdragsgiver kan bestille nye ferdig-loddet kretskort ved behov, og koble til sensorene de har bruk for.




Figur 3.43: Forside av kretskort, lagd i easyEDA




Figur 3.42: Bakside av kretskort




Figur 3.44: Kretsskjema, lagd i easyEDA

Det er tatt utgangspunkt i referansedesignet for mikrokontrolleren hentet fra databladet[37] og en guide hentet fra blnlabs[38]. Designet består i hovedsak av en ESP32-S3-wroom-1 mikrokontroller, en USB-C tilkobling, en spenningsregulator, trykknapper og skruterminaler, se figur 3.44. Spenningsregulatoren som er brukt er en AMS1117-3.3 LDO-regulator. Ifølge databladet til mikrokontrolleren kan spenningstilførselen trekke opp mot 0,5A, det ble derfor valgt en spenningsregulator som kan levere 1A. For å beskytte kretsen ble det benyttet en Schottky-diode. Alle komponenter er valgt fra leverandøren LCSC[39] siden de har et samarbeid med JLCPCB, som gjør det enkelt å få kortet ferdig produsert ved bestilling.

Det ble valgt en USB-C tilkobling i stedet for Mikro-USB. Dette på grunn av at de oppleves som mer robust, og siden det meste av ny elektronikk går over til USB-C. Dette medfører noe mer kompleksitet, litt økt pris og gjør det vanskeligere å lodde dem for hånd. Ved bestilling av ferdig produsert kort, vil ikke dette være noe problem.

Designet har tre forskjellige bredder på ledningsbanene 0,245mm, 0,5mm og 1mm. 0,245mm er i hovedsak brukt for signalbanene til USB tilkoblingen, siden det ikke er plass til bredere på tilkoblingen. På resten av designet er det brukt 1mm der det er plass og 0,5mm på resten. Siden spenningstilførselen til mikrokontrolleren kan trekke opp mot 0,5A, og med en kobbertykkelse på 35µm bør bredden på banene være over 0,3mm.

På grunn av manglende tid ble det besluttet å ikke produsere denne sensornoden, men underlagene for å produsere denne vil være tilgjengelig for Tokvam for videreutvikling.

# 4 RESULTATER

Dette kapitelet tar for seg resultater fra ulike tester av systemet, og resultatene i oppgaven som en helhet. Utførte tester er rekkeviddetest, batterilevetid, nøyaktighet av målinger, samt en helhetlig test på utstyr hos Tokvam.

## 4.1 DESIGN OG PLASSERING AV KOMPONENTER

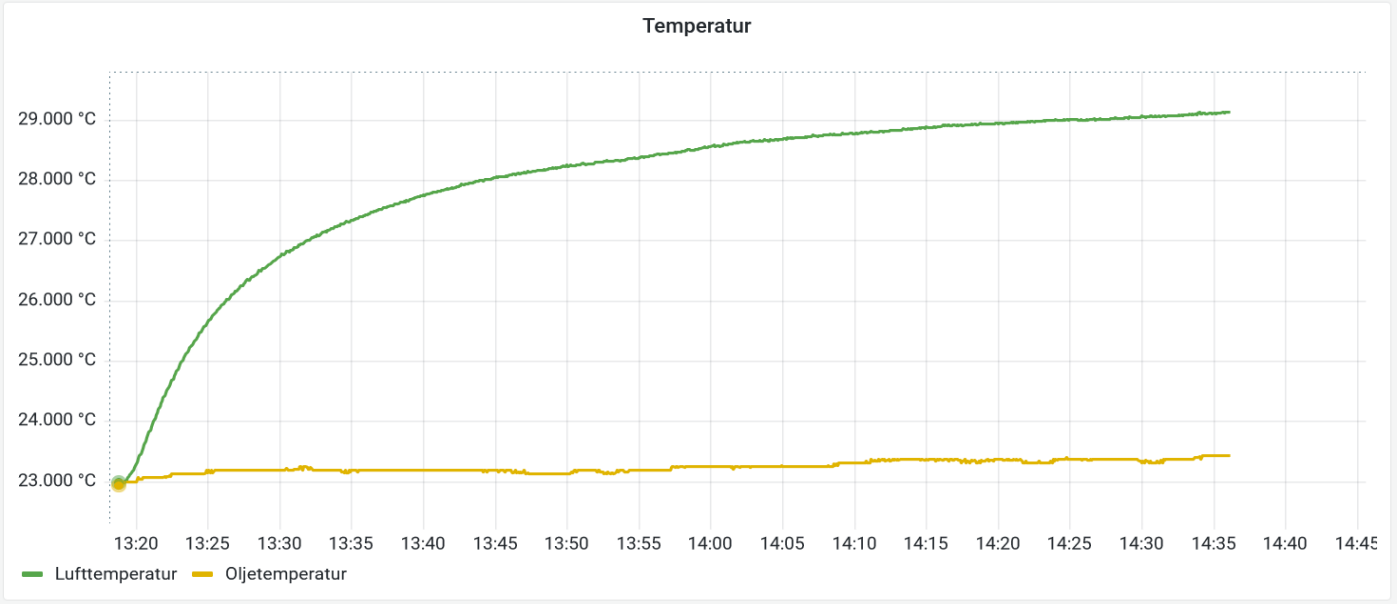
Alle komponentene passet i boksen som tiltenkt. Plassering av bryteren i lokket gjør det enkelt å skru av og på noden. Designet på sensornode 1 har noen praktiske ulemper. Skrutilkoblingene er plassert slik at når braketten er limt på kretskortet, har man ikke lenger tilgang på skruterminalene. Løsningen med å lime braketten til kretskortet på denne versjonen passet derfor dårlig. Sensornode 2 og 3 fungerte bedre grunnet ny plassering av skruterminalene som sørger for tilgang til skruene. Plasseringen på USB-C portene er ulik på alle de tre kretskortene, hvor plasseringen av USB porten på sensornode 2 er litt vanskelig å få plugget inn.

## 4.2 SPENNINGSREGULERING

Mikrokontrollerne er avhengig av en spenning på 3,3 V fra spenningsregulatoren for å fungere. Under testing av strømforsyningen kan spenningen over de 3 AA-batteriene falle under 3,53 V avhengig av type batterier som benyttes, slik at kretskortet slutter å fungere. Hvis batterispenningen faller under 3,53 V vil derfor noden slutte å fungere selv om batteriene ikke er helt tomme. Målinger på spenningsregulatoren viser at den gir ut nesten helt nøyaktig 3,3 V så lenge V_IN har en spenning over 3,53 V.

## 4.3 TEMPERATURMÅLINGER

Varme produsert fra mikrokontrolleren øker temperaturen på innsiden av boksen. Boksen er helt tett og isolert fra utetemperaturen, så den interne temperaturmålingen er derfor høyere. Det ble gjennomført temperaturmålinger fra oppstart på node 1 for å undersøke hvor lang tid det tar å nå en konstant $\Delta t$, og hvilket avvik den har. Før oppstart har noden vært avskrudd i 23 °C varmt rom. Resultatene av testen vises i Figur 4.1. Temperaturøkningen internt i noden følger en logaritmisk funksjon, og stabiliserer seg etter ca. 1 time. Det stasjonære avviket $\Delta t$ til den interne temperatursensorer ligger på omtrent 29,1 °C – 23,5 °C = 5,6 °C.

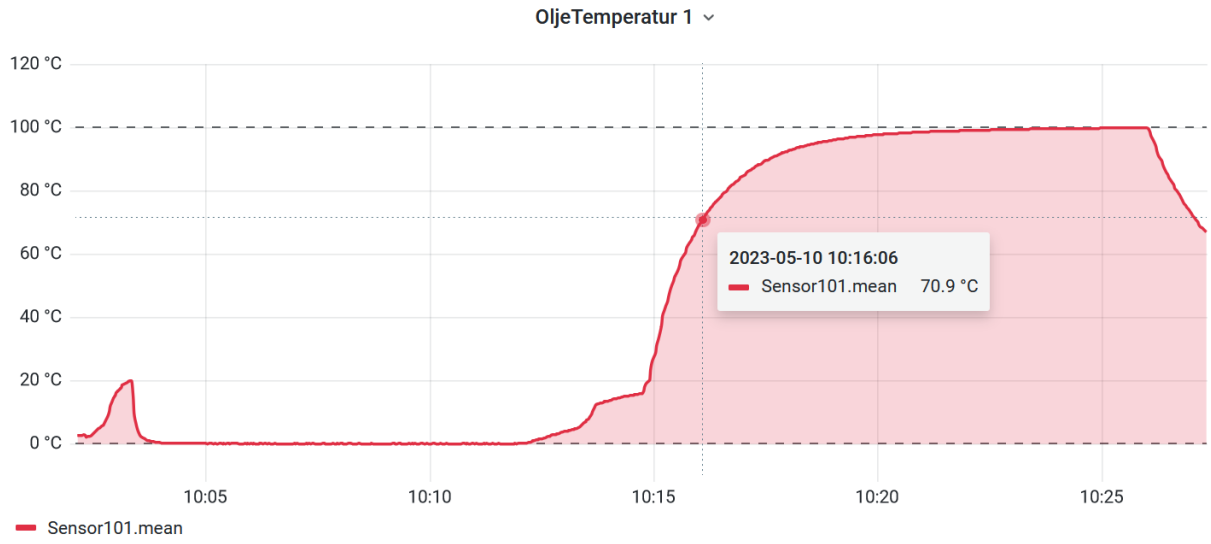



*Figur 4.1 Test av intern temperatursensor oljetemperatursensor ved lik temperatur, graf hentet fra Grafana*

## 4.4 VERIFISERING

### 4.4.1 Verifisering av oljetemperatursensor hos IKM

Oljetemperaturmålingene under hele testperioden vises i figur 4.2. Etter 8 minutter ved klokka 10:12 i isvannet var referansetemperaturen på 0,008 °C og oljetemperatursensoren var på 0,0675 °C. Når oljetemperatursensoren ble plassert i varmeovnen klokka 10:15 økte temperaturen raskt til den stabiliserte seg etter 10 minutter. Referansetemperaturen var på 99,59 °C og oljetemperatursensoren var på 99,8 °C. Responstiden $T_r$ på oljetemperatursensoren i varmeovn er når sensoren viser 63 % av stasjonær verdi. Ved temperatur på 0,63*80 + 20 = 70,4 °C, som markert på figur 4.2, ligger responstiden på 1 minutt og 6 sekunder. Resultatene etter test av temperatursensor hos IKM, viste at oljetemperatursensoren DS18B20 var nøyaktig.




*Figur 4.2: Temperaturmålinger fra oljetemperatursensoren under testing*

### 4.4.2 Kalibrering av akselerometer

Kalibrering av akselerometeret ble gjort etter metode beskrevet i kapittel 3.8.4. Resultatene fra målte akselerasjonsverdier i alle tre retninger vises fra skjermklipp i seriellmonitoren i Arduino IDE. Generelt for alle tre målingene er at feilmarginen er til dels like i alle målingene. Trekker fra en konstant verdi for å kalibrere opp mot stasjonære viberasjonsmålinger.

Test før kalibrering i normal posisjon

```
"Vibrasjon X":0,"Vibrasjon Y":0.008,"Vibrasjon Z":1.048000027
```

Test før kalibrering i vertikal posisjon

```
"Vibrasjon X":1.019999979,"Vibrasjon Y":0.004,"Vibrasjon Z":0.044
```

Test før kalibrering i horisontal posisjon

```
"Vibrasjon X":0,"Vibrasjon Y":1.023999972,"Vibrasjon Z":0.04
```

Etter observasjon av målte stasjonære verdier som vises over. Justerer verdi av x og y med å trekke fra 0.01 g og trekker fra 0.04 g på z-retning.

Resultatene etter utført kalibrering:

Test etter kalibrering i normal posisjon

"Vibrasjon X":-0.01,"Vibrasjon Y":0.008,"Vibrasjon Z":1.008000027

Test etter kalibrering i vertikal posisjon

"Vibrasjon X":1.013999972,"Vibrasjon Y":-0.002,"Vibrasjon Z":0.004

Test etter kalibrering i horisontal posisjon

"Vibrasjon X":-0.006,"Vibrasjon Y":1.013999972,"Vibrasjon Z":4.213548607e-10

Retning x og y blir litt nærmere 0 enn før kalibrering, men ettersom disse verdien skal være henholdsvis 0 eller 1, er verdien på den ene siden litt for lav og den andre litt for høy. X og Y retning har derfor ikke et helt lineært målområde fra 0 til 1 g. Z-aksen ble mer nøyaktig etter kalibrering og er nær 0 g og henholdsvis 1 g når den skal være det.

Før kalibreringen gikk den totale g-kraften i utregningen fra:

"Vibrasjon":-0.043958167

Til en verdi på

"Vibrasjon":0.04819085

Etter kalibrering gikk den totale g-kraften etter utregningen fra:

"Vibrasjon":-0.005386483

Til en verdi på

"Vibrasjon":0.039834664

## 4.5 REKKEVIDDETEST

### 4.5.1 Fri luftlinje test 1

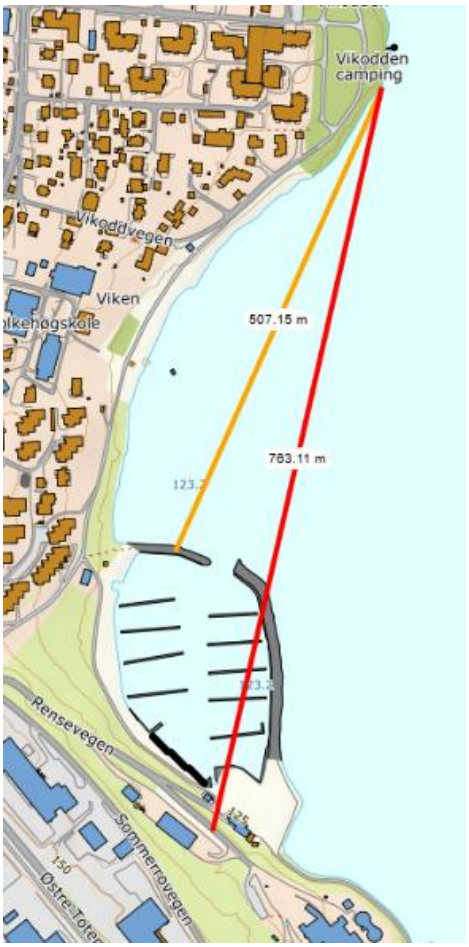
Testing av rekkevidde ble uført den 24. april, i 0 °C og snøvær. Fikk derfor også testet rekkevidden i mindre ideelle forhold som ofte kan oppstå når utstyret skal tas i bruk. Denne testen ble utført i mest mulig åpen luftlinje. ESP32 mottakeren sammen med PC-en ble plassert på en gangbru noen meter over bakken for å sørge for fri luftlinje, mens sensornodene ble fraktet lengere unna. Resultatene fra testen vises i figur 4.3. Sensornode 2, rød strek, sluttet å sende data stabilt ved 322 m. Sensornode 1 og 3 var fortsatt i stand til å sende data ut til 395 m fra mottaker. Begrensingen i denne testen ble at det ikke var mulig å gå lenger unna og fortsatt ha fri luftlinje, så testen stoppet på 395 m. Ved omtrent 200 m ble hyppigheten på mottatt data lavere. Hyppigheten på mottatt data sank relativt med økt avstand.


*Figur 4.3: Bilde av området brukt for test av fri luftlinje 1. Utsnitt fra Norgeskart.*

### 4.5.2 Fri luftlinje test 2

Fredag 28.04 ble det utført en sekundær rekkeviddetest med klart vær. Testen ble utført med mest mulig ideelle forhold for å teste ut maks rekkevidde til sensornodene. Den ble utført ved to lengder, vist i figur 4.4, en ved 507 m markert med gul strek, og ved 783 m vist med rød strek. Mottakeren forholdt seg stasjonært under hele testen mens sensornodene ble flyttet. I hver lengde ble alle nodene testet hver for seg og deretter alle nodene samtidig.

I første lengde (gul strek) ble måleverdiene sendt i helt fri luftlinje over en vannflate. Her oppsto det ingen problemer og alle signaler ble mottatt i riktig intervall. I andre lengde (rød strek) ble nodene flyttet til en avstand på 783 m på en høyde over en båthavn. Her varierte hyppigheten på mottatt data. Det ble mottatt signaler fra alle nodene, både hver for seg og alle samtidig.

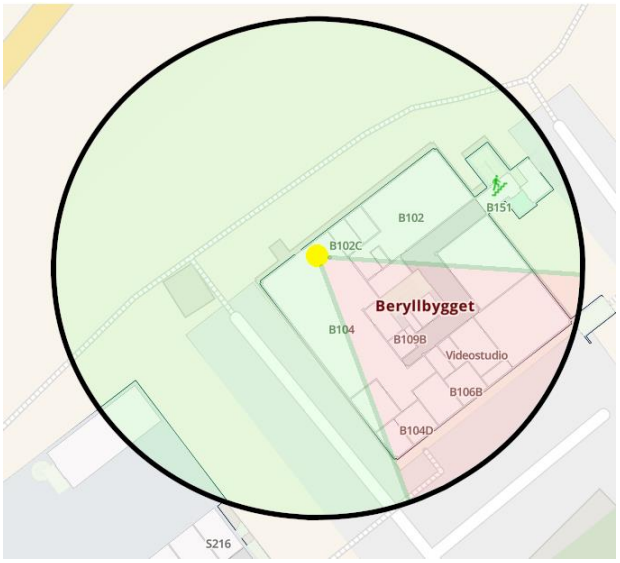



*Figur 4.4: Bilde av område brukt for test av fri luftlinje 2. Utsnitt fra Norgeskart*

### 4.5.3 Signaltest ved hindringer

Test av signal igjennom bygg ble gjennomført 25. april i Beryll-bygget på NTNU i Gjøvik. Dette ble testet for å se hvordan vegger i bygg kan påvirke signalstyrken fra sensornodene. Wi-Fi signaler kan bli veldig begrenset av vegger i bygg, så testen ble gjennomført for å få en indikasjon for utfordringer med signalstyrken mellom mottaker og sensornodene.

Beryll bygget består av tykke betongvegger og flere indre rom som vist i figur 4.5. Testen ble gjennomført med at mottakeren forble stasjonært inne, indikert med gul prikk, samtidig som sensornoden ble tatt ut av bygget og fulgte den sorte sirkelen rundt bygget. Tap av signal ble kommunisert, og det ble merket på kartet. Som vist i figuren med rød farge, ble signalet borte når det var tre eller flere vegger imellom. Ett vindu fremstår ikke som en begrensning for kommunikasjonen, som vil tilsvare en kupé på traktor. Dette gir godt grunnlag for test av hele systemet, hvor mottaker og dashbord kan være tilgjengelig og synlig for sjåfør inne i kjøretøyet mens systemet er i bruk.




*Figur 4.5: Kart over testområde for test av signalstyrke. Markert område gjelder kun utenfor byget. Utsnitt fra Mazemap over Beryllbygget*

## 4.6 BATTERILEVETIDSTEST

Det ble utført to tester for å undersøke levetiden til batteriene i sensornodene. Testen ble utført i to omganger, en der ESP-NOW meldingene ble sendt hvert sekund og en hvert 100 ms. Målet med test nummer to var å undersøke om hyppigheten av sendte meldinger over ESP-NOW påvirker batterilevetiden.

### 4.6.1 Batterilevetidstest 1

Testen ble utført ved å starte alle nodene likt med fulladede batterier for så å ta tiden til de sluttet å sende data. Alle nodene ble satt opp slik at de sendte data en gang hvert sekund, og med 1,2 V ladbare NiMH AA-batterier. Testen ble startet klokken 10:50 30.04.2023. Resultatene kan ses i Tabell 2.

*Tabell 2: Resultater fra batterilevetidstest ved sendefrekvens på 1 s*

| Sensornode | Start | Slutt | Levetid |
|---|---|---|---|
| 1 | 10:50 30.04.2023 | 13:18 01.05.2023 | 26t 28m |
| 2 | 10:50 30.04.2023 | 14:33 01.05.2023 | 27t 43m |
| 3 | 10:50 30.04.2023 | 07:05 01.05.2023 | 20t 15m |

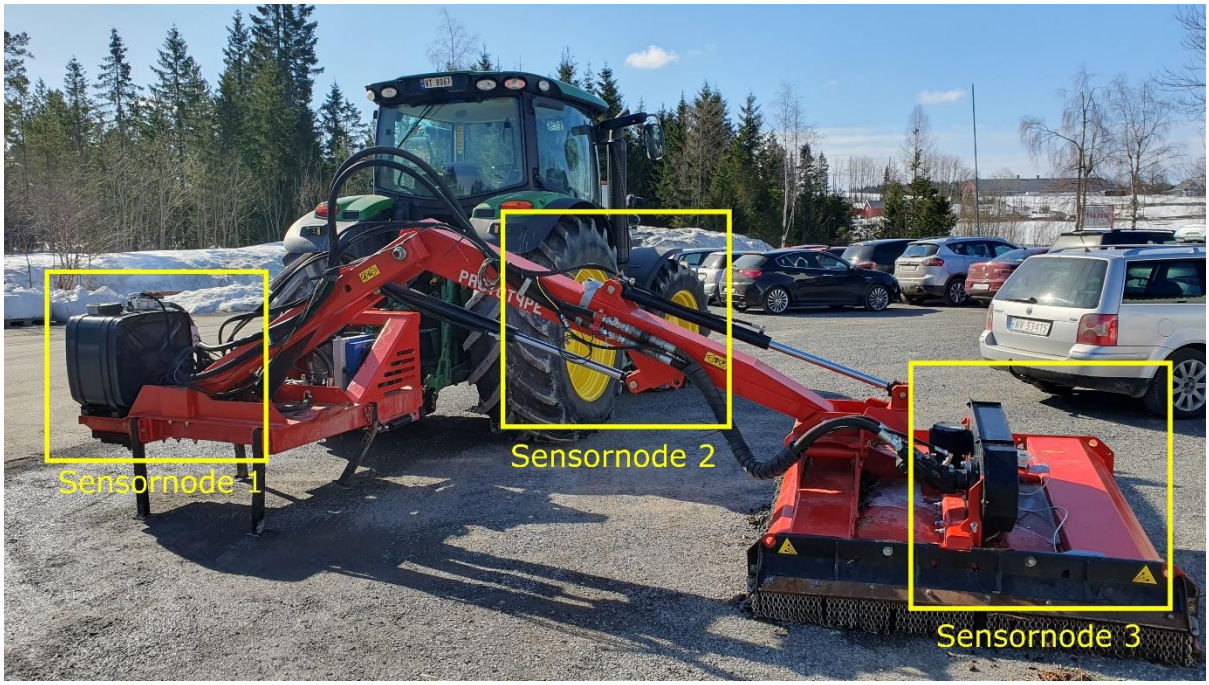
### 4.6.2 Batterilevetidstest 2

Test nummer 2 ble utført på samme måte som test 1, med 100 ms sendefrekvens. Resultatet er at batteriene i liten, eller ingen grad påvirkes av sendefrekvensen, som vises i tabell 3.

*Tabell 3: Resultater fra batterilevetidstest ved sendefrekvens på 100 ms*

| Sensornode | Start | Slutt | Levetid |
|---|---|---|---|
| 1 | 10:14 12.05.2023 | 12:44 13.05.2023 | 26t 30m |
| 2 | 10:14 12.05.2023 | 13:08 13.05.2023 | 26t 54m |
| 3 | 10:14 12.05.2023 | 06:23 13.05.2023 | 20t 09m |

## 4.7 FUNKSJONELL TEST AV SENSORSYSTEMET PÅ KRATTKNUSER




*Figur 4.6: Oversiktsbilde over traktor med prototype arm til kratknuser*

Funksjonell test av sensorsystemet hos Tokvam på prototype-hydraulikkarm med krattknuser ble utført. Mottakeren ble plassert inne i kupéen til traktoren med fører, samtidig som alle 3 sensornodene var tilkoblet armen. Det var 7 °C med sol på testdagen. Første test var om signalet mellom nodene og mottakeren ville fungere gjennom glasset på kupéen og om eventuelt metallet i armen ville sperre for signalene. Det oppstod ingen problemer med den trådløse kommunikasjonen, all data ble mottatt med riktig intervall.

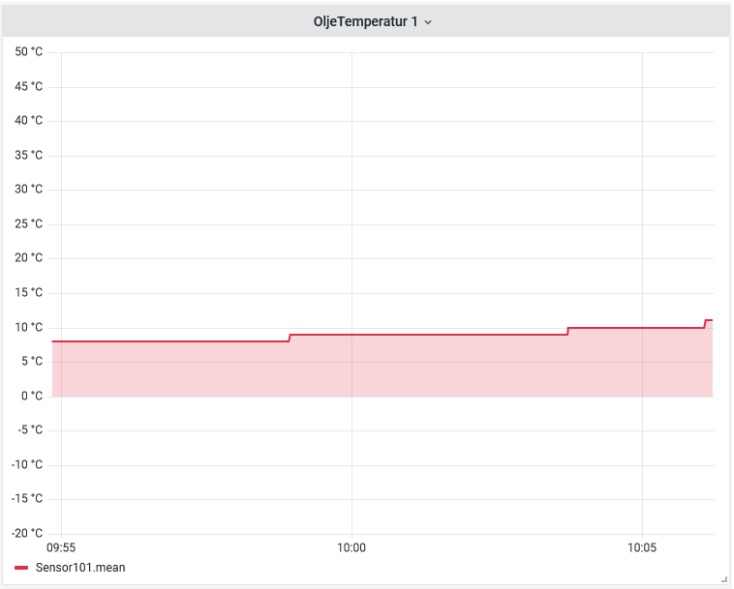
Traktor med montert arm ble kjørt i nærområdet og testet i drift. Krattknuseren ble utsatt for høye rotasjonshastigheter og armen ble testet ved flere stillinger. Krattknuseren ble senket ned i snø for å simulere normal bruk, ettersom testen ble utført på våren.

### 4.7.1 Sensornode 1

Sensornode 1 ble festet ved siden av hydraulikkoljetanken med magnetene og sikret med gaffateip. Oljetemperatursensoren markert med grønn ramme i figur 4.7, ble plassert i hydraulikkoljetanken. Driftstemperatur på oljen ved kontinuerlig drift vil typisk ligge på 50 °C, men tar en stund å komme til denne temperaturen ettsom oljetanken rommer mange liter olje.

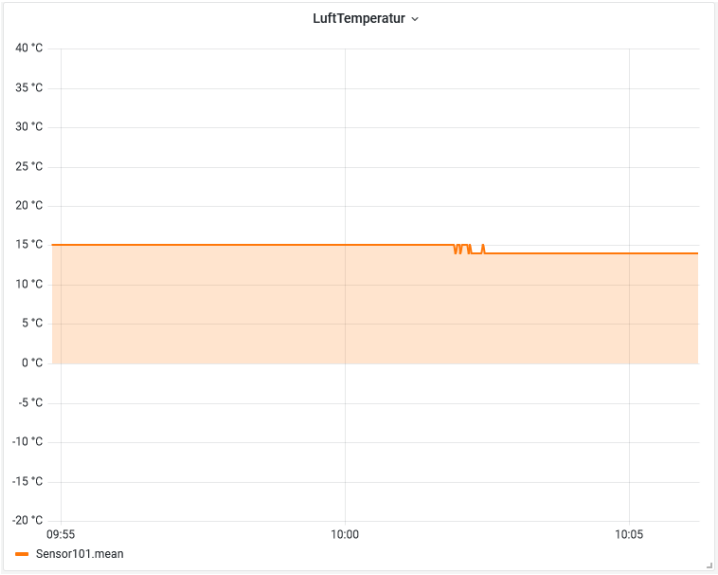
*Figur 4.7: Sensornode 1 festet og sikret ved oljetanken*

Resultatene av målte oljetemperaturer vises i figur 4.8. Sensoren vil gi en god indikasjon på temperatur økning, og om temperaturen kommer på et kritisk nivå. Denne sensornoden har også en temperatursensor inne i boksen for å sammenligne med temperaturen i omgivelsene. Differansen mellom oljetemperaturen og omgivelsestemperaturen vises i figur 4.10.



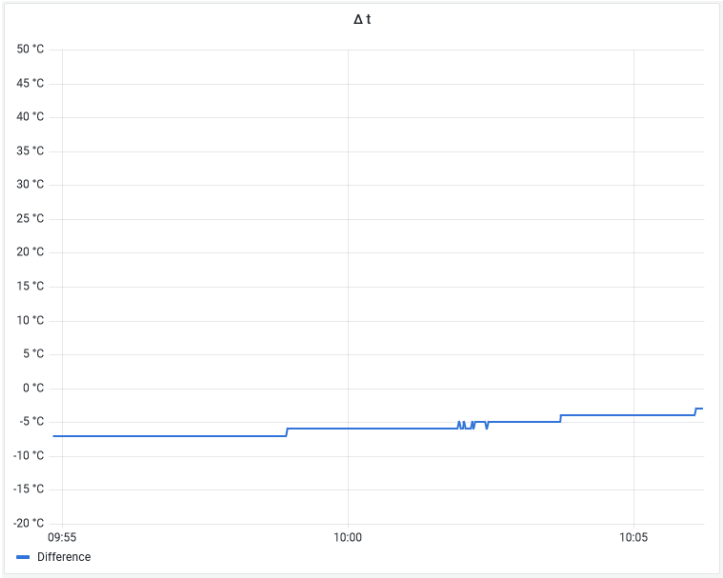

*Figur 4.8: detaljert bilde av målt temperaturendringer under funksjonell test*

Lufttemperaturmålingene viser en nedgang. Dette viser at boksen ikke hadde oppnådd reell lufttemperatur ved oppstart, som var 7°C. Det vil ta tid før denne oppnår omgivelsestemperatur grunnet at den er isolert inne i boksen, som forklart i kapittel 4.3. Datatypen på testtidspunktet var satt til int, temperaturen vises derfor kun i hele grader.




*Figur 4.9: Graf for lufttemperatur under test*

Resultatene på Δt var også som forventet. Den trekker omgivelsestemperatur fra oljetemperatur. Siden omgivelsestemperaturen var 7°C varmere enn hydraulikkoljen vises negative verdier. Ved lengere drift forventes en økning i oljetemperaturen, og Δt vil få positive verdier.




*Figur 4.10: Graf for endringer i Δt under test*

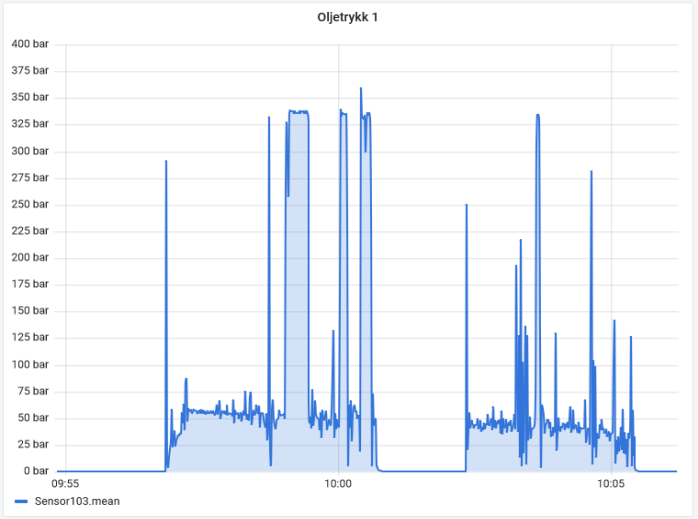
### 4.7.2 Sensornode 2

Sensornode 2 ble festet med magnetene og sikret med gaffateip på kranarmen og trykksensoren indikert med grønn firkant i figur 4.11, ble koblet på hydraulikkslangen. Her ble det forbeholdt sikkerhetsavstand ved oppstart og drift, i samsvar med risikovurdering og Tokvam sine retningslinjer.

*Figur 4.11: Sensornode 2 festet oppå armen med trykksensor stripset fast i slange*

Trykket vil øke signifikant når fører av traktor øker rotasjonen på kjettingen. Det er flere hydraulikk slanger som kan måles, men ved denne testen var det den som spesifikt gikk til krattknuseren som ble målt. Det var tilkoblet en MINIMESS[40] adapter på uttaket, slik at prosessen med å koble til og fra var forenklet, og risikoen for lekkasje minimal.

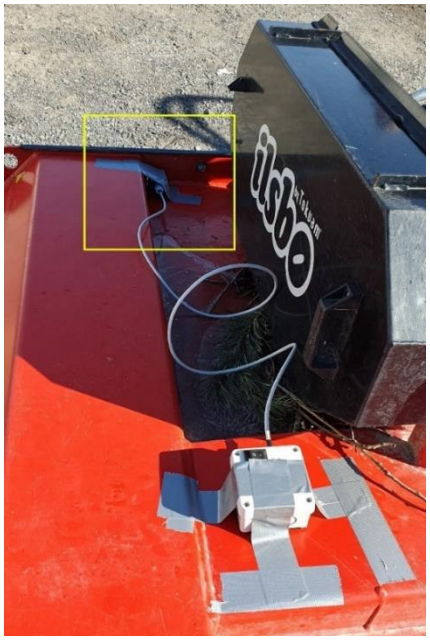
Verdier for oljetrykk viser når kjettingen i krattknuseren roterer, da ligger trykket på omtrent 50 bar, se figur 4.12. Krattknuseren ble senket ned i snø under testen. Dette førte til økt trykk, og det vises som de store økningene i målte trykkverdier. Hydraulikksystemet består av en sikkerhetsventil som slår ut på omtrent 335 bar. Grafen viser tydelig når sikkerhetsventilen slår ut.




*Figur 4.12: Målt oljetrykk*

### 4.7.3 Sensornode 3

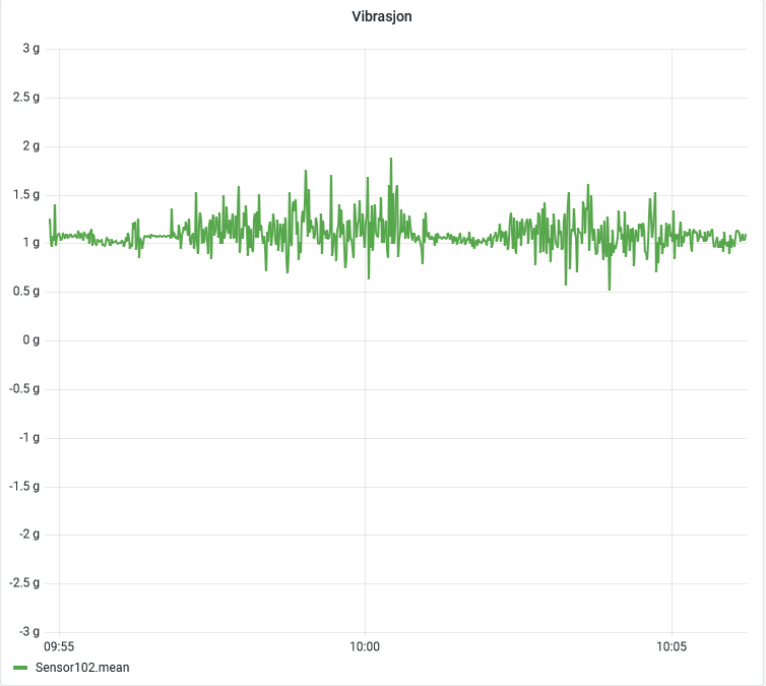
Sensornode 3 ble plassert på metallskjoldet til krattknuseren slik at vibrasjoner fra både roterende kjetting og motor kan måles. Hensikten er å kunne oppdage og eventuelt analysere sammenheng mellom vibrasjonen og rotasjon av kjettingen som vist i figur 4.14 under bruk, i tillegg til å avdekke feil med ustyret.



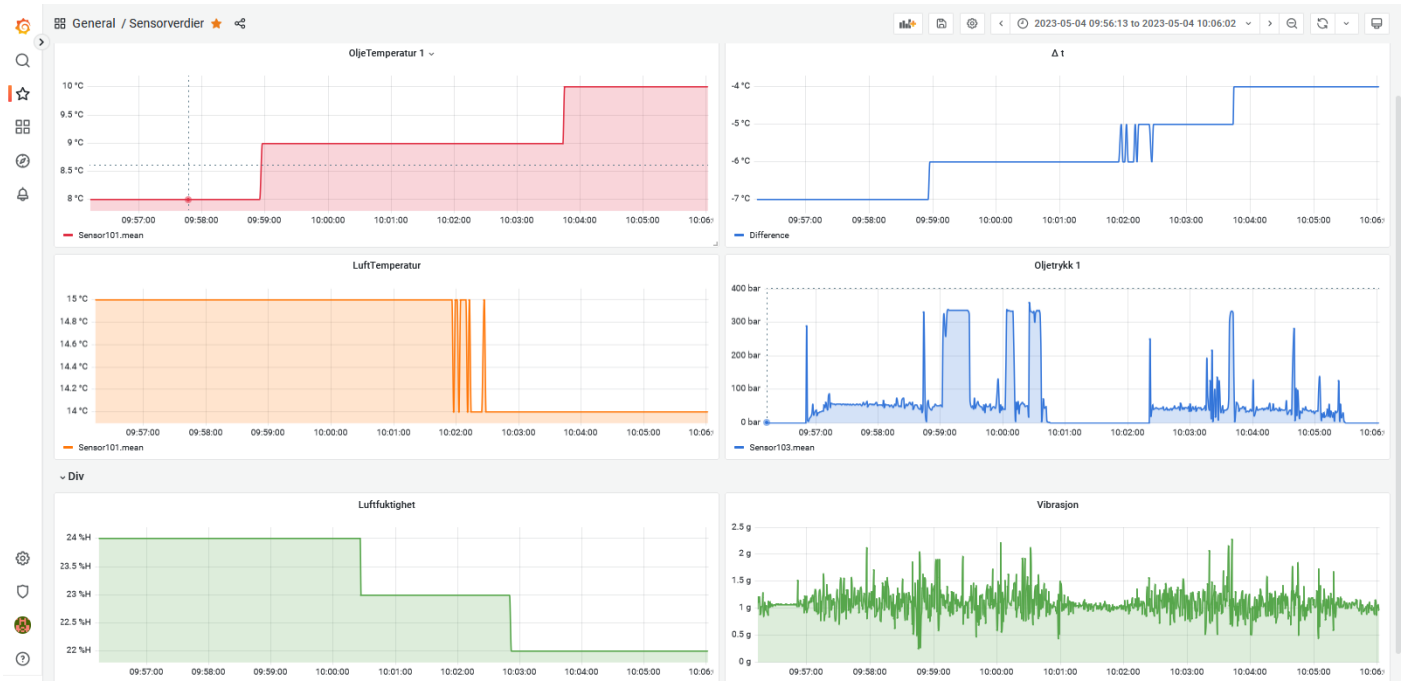
*Figur 4.13: Sensornode 3 sikret med teip. Akselerometer med hus indikert med gul firkant*

*Figur 4.14: Underside av krattknuser med kjede. Bildet er av en annen modell med tilsvarende kjede hentet fra Tokvams produktkatalog 2021[6]*

Vibrasjonsdata fra test viser når krattknuseren er i drift. Målte viberasjonsverdier vises i figur 4.15. Figur 4.12 for oljetrykk og 4.15 viser at det er en sammenheng mellom målte viberasjonsverdier og oljetrykket. Når krattknuseren er avskrudd går vibrasjonene ned til 1 g, og den høyeste målte viberasjonsverdien var på 1,8 g. Skjermbilde av samlet data fra testen vises i figur 4.16.




*Figur 4.15: Målt vibrasjon*




*Figur 4.16: Dashbord med data fra funksjonell test*

# 5 DISKUSJON

I dette kapitelet diskuteres resultatene oppnådd i kapittel 4. Denne delen tar for seg hvilke utfordringer som oppsto underveis, diskusjon rundt resultater, hva som kunne blitt gjort bedre og etikk rundt prosjektet.

## 5.1 UTFORDRINGER

### 5.1.1 Bestilling og valg av mikrokontrollere

Det var en god del utfordringer knyttet til bestilling av komponenter med tanke på leveringstid. To ganger gikk bestillingen mye tregere enn antatt på grunn av uforutsette utfordringer, som at det ble levert feil sted og ble borte. Bestilling av mikrokontrollere var også utfordrende med tanke på lover og regler i USA som begrenser eksporten av disse, slik at bestillingen måte manuelt kontrolleres. Dette gjorde det vanskelig å gjøre små endringen som trengte bestilling av nye komponenter, på grunn av usikkerheten rundt leveringstid. På grunn av nevnte utfordringer ble resultatet å bruke de første mikrokontrollerne som ble bestilt, fremfor å prøve å bestille flere for å ha like mikrokontrollere til sensornodene.

### 5.1.2 Datamaskin for logging

Løsningen for å logge og visualisere måleverdiene på en laptop kunne ha blitt gjort bedre. En utfordring er at PC-en må være slått på og logget inn hele tiden for å kunne lagre måleverdiene. Dette førte til at PC-en kan gå tom for strøm uten muligheter for opplading inne i traktoren.

Det var i utgangspunktet tenkt å bruke en Raspberry pi til logging og fremvisning av måleverdier. Grunnet en global mangel på Raspberry pi var de vanskelig å få tak i til dette prosjektet. Ved å bruke en Raspberry pi eller tilsvarende kunne man brukt 12 v-uttaket i traktoren som strømtilførsel. I tillegg ville den tatt mindre plass og startet automatisk når den ble koblet til strømtilførsel.

Det var også utfordringer med server PC-en pga. intern programvare, begrenset tillatelser og en episode med at en oppdatering slettet all tidligere lagret logg. Dette er ganske kritisk, når selve dataloggen kan bli slettet hvis bruker ikke er bevisst på å ikke oppdatere unødvendig. Det vil også derfor være viktig å sikkerhetskopiere grafer og tallverdier fra målingene.

### 5.1.3 Skrive i Word med bruk av Endnote som referanseverktøy

Var knyttet enkelte utfordringer til å bruke Word i kombinasjon med Endnote som referanseverktøy. Rekkefølgen på referansetallene i rapporten er ikke helt riktig, ettersom Endnote prioriterer å telle referansen i tekstboksene før den teller referansen fra resten av teksten. Fantes ingen måte å kunne endre på dette slik at første referansen i rapporten begynte med 1.

## 5.2 ANALYSE AV RESULTATENE

### 5.2.1 Rekkeviddetest

Forventet rekkevidde til ESP-NOW protokollen er på 480 meter, med stabil kommunikasjon opp til 190 meter som beskrevet i teorien. Under testene, som hovedsakelig var for å bekrefte en makslengde, ble det større utfordringer å finne områder med lang nok luftlinje for å nå kritisk punkt for den trådløse kommunikasjonen. Lengste frie luftlinjen testet på alle tre sensornodene var 783m og data ble fortsatt mottatt dog med litt varierende intervaller enn optimalt.

Test rundt Beryll bygget fikk også bekreftet forventningene om tap av signal når hindringer er i veien. Testen viste at Wi-Fi mister mye signalstyrke når det er flere tykke vegger imellom, men vil fungere fint med f.eks. glass og tynnere hindringer.

Satt sammen ble det et godt grunnlag for funksjonell test, og det ble ikke nødvendig å forbedre signalstyrke fra sensornodene med eventuelle eksterne antenner eller forandring av kapsling. Når tap av signal, enten ved brudd i kommunikasjon eller ved av og på skruing fungerte ESP-NOW protokollen bra på grunn av at det er en «connectionless» protokoll. Slik at tilkoblingen ble automatisk igjenopprettet når det var signalstyrke igjen.

### 5.2.2 Batterilevetidstest

Resultatene fra begge testene er at batteritiden på sensornode 1 og 2 ganske like, samtidig har sensornode 3 noe kortere levetid. Dette skyldes trolig at Mikrokontrolleren i sensornode 3 har en innebygd RGB-led som lyser ganske sterkt og vil trekke en del strøm. Generelt sett er batteritiden på nodene bedre enn forventet, og er bra nok til å holde flere arbeidsdager med produkttest.

I test 2 ser man at batteritiden ble lite påvirket av hyppigere målinger og sending av data over ESP-NOW. Forventet resultat var at batteritiden ville synke, og det var derfor bedre enn forventet. På bakgrunn av denne testen ble det besluttet å fortsette å sende data med 100 ms intervaller. En raskere sendefrekvens er spesielt nyttig til måling av trykk og vibrasjon, da disse målingene endres raskt.

### 5.2.3 Funksjonell test av sensorsystem

Det første resultatet under produkttesting var at magnetløsningen var dårligere enn forventet. Selv om angitt kilo holdekraft til magnetene var mye større enn vekten til sensornodene så mistet de effekt grunnet lakk, rust og støv på metallet. De fungerte til å holde de fast, men sklidde langs med overflaten. Dermed var det nødvendig med gaffateip for å sikre sensornodene. Dette er kritisk for at systemet skal fungere som tilegnet. Det er mye krefter og mekanikk som er i bevegelse, så det er viktig at sensornodene ikke løsner under drift.

Oljetemperaturmålingene tilsvarte med forventningene og som i figur 4.8 viser, går temperaturen opp i samsvar med driftstid. Oljen er lagret i en stor tank og vil ha omgivelsestemperatur som utgangspunkt. Det tar lang tid å se temperaturendringer, som viser at kjøleeffekten på hydraulikken har god effekt. Under test ble målingene fra sensornode 1 sendt som int, dette ble endret til float for å fremvise mer nøyaktige måleverdier.

## 5.3 FORBEDRINGSMULIGHETER

Generell konstruksjon av sensornoder har noen momenter som kan utbedres. Å forbedre den strukturelle integriteten til innholdet med skruer istedenfor lim på brakett og batteriholder er et moment som kan gjøres annerledes. Det er også en utfordring med plass i boksene til sensornodene.

Er forbedringspotensial knyttet til design på strømforsyningsdelen av kretsen til nodene. Jamfør teorien til spenningsregulatoren, kapitel 2.8, hadde en høyere spenning fra batteriene vært bedre, som viser at batterikapasiteten er avhengige av den nominelle spenningen på AA-batteriene. Dette kommer igjen som en utfordring for å få plass til ett AA-batteri til, hvor eventuelt overgang til litium-ion batterier som tar mindre plass kan være en potensiell forbedring. Litium-ion batterier vil også klare å holde spenningen oppe selv når det blir mindre batterikapasitet[41]. Beskyttelse mot å sette inn batteriene feil vei (motsatt polaritet) er ikke bygd inn i de konstruerte sensornodene. Dette er en forbedring som ligger i sensornode 4, hvor en schottky diode beskytter mot motsatt polaritet, men denne ble ikke konstruert.

Som beskrevet i kapittel 5.2.3 hadde magnetene under sensornodene for liten holdekraft til å være effektive. Det ble forsøkt med større magneter med mer holdekraft som fremstår som en potensiell løsning på utfordringen. Under test måtte det teip til for å holde de på plass, noe som ikke er ønsket. Eventuelt kan det benyttes skruer, men dette blir en løsning som går vekk fra enkelheten og portabiliteten som et underordnet mål for systemet.

Temperatursensor for lufttemperatur i sensornode 1 sitter inne i boksen, og dette gir utfordringer i å få reelle og øyeblikkelige målinger. Siden boksens konstruksjon har IP67 standard vil den være isolert for omgivelsene, og det vil derfor være en forsinkelse i temperaturen avhengig av boksen tilholdssted før bruk. Kretsen og mikrokontrolleren inni vil også ha påvirkning på temperaturen, og det vil skape en offset på verdiene som må kalibreres. En utvendig temperatursensor hadde vært en bedre løsning.

Som beskrevet i kapittel 3.2.1 var utgangspunktet å bruke Raspberry Pi for en mer dedikert stasjon/server for å samle inn data og loggføre, men valget falt på å bruke en Windows-PC. Dette fordi den var tilgjengelig fra Tokvam, og Windows er generelt sett mer brukervennlig enn Linux-basert programvare. Det betyr at PC-en må slåes på før oppstart sammen med resten av systemet for å registrere og loggføre verdier. I en forbedret versjon av systemet ville det ideelt vært en mottaker som alltid var klar til å ta imot verdier og kun hadde jobben med å loggføre, hvorpå dataen kunne bli avlest og analysert når ønsket.

## 5.4 Etikk

Det er etiske utfordringer knyttet til produksjon av elektronikk når det kommer til mineraler som utvinnes i land med stor risiko for menneskerettighetsbrudd. Ofte kalt 3TG mineraler som ofte vil være mineraler som gull, tantal, tinn og wolfram. Dette er mineraler som ofte brukes i produksjon av elektronikk.

Produksjon av elektronikk bidrar til store globale klimautslipp, dette på grunn av mengden som produseres. Ny estimater viser at det gjennomsnittlige e-avfallet per person i verden var på 7,3 kg i 2020. Dette er en økning på 21 % sammenlignet med hva det var i 2014. Estimatene viser også at avfallet vil fortsette å øke fram mot 2030 [42].

Et viktig tiltak vil være å produsere elektronikk som er laget for å fungere lengst mulig, for å redusere forbruket. Systemet i oppgaven er designet for å være modulerbart slik at det har lengst mulig levetid.

Det kom ny regulering i EU 1. januar 2021 [43] som har som mål å sørge for at mineralene gull, tinn, wolfram og tantal utvinnes og produseres etter standarder satt av OECD. I tillegg til å stoppe konflikter, mishandling og utnyttelse av lokalbefolkningen.

Produktet er laget for å kunne bedre utstyr til framkommelighet som vil bidra til FN bærekraftmålet 9.1 «*Utvikle pålitelig, bærekraftig og solid infrastruktur av høy kvalitet, inkludert regional og grensekryssende infrastruktur, for å støtte økonomisk utvikling og livskvalitet med vekt på overkommelig pris og likeverdig tilgang for alle*» [44].

# 6 KONKLUSJON

Hensikten med denne rapporten var å beskrive valg og metode for en mulig løsning for loggføring av sensorverdier under produkttesting hos Tokvam. Ønsket fra oppdragsgiver var å kunne ha et bedre og enklere system for å kunne samle inn ulike fysiske verdier under testing av traktorredskaper for snøbrøyting og kantklipping. Rapporten gikk ut på å undersøke muligheter for innsamling av data over trådløs kommunikasjon, knyttet til ulike sensorer. I tillegg til å lage en oversikt over sensorverdier i sanntid og loggføring, for sensorer for temperatur, oljetrykk og vibrasjon.

Løsningen ble å lage flere ulike noder knyttet til forskjellige målinger, alle med en egen mikrokontroller. De ulike sensornodene sender data til en mottakernode, som igjen er koblet direkte over USB til en PC, hvor måleverdiene loggføres og fremstilles. ESP-NOW protokollen ble benyttet for å sende data trådløst fra de ulike sensornodene til mottakernoden. Det ble designet flere sensornoder tilknyttet målingene på utstyret. Sensornodene er alle konstruert i vanntett boks for å tåle det miljøet de kan bli utsatt for under testing. Kontroll av målte verdier i forhold til en referanseverdi ble kun gjort på temperaturmålingen, som var knyttet til utfordringer med å ha tilgang til verktøy og utstyr for å kunne måle dette for oljetrykk og vibrasjon.

Testing av sensornodene ble utført for å avdekke potensielle feil, utfordringer og forbedringer. Den funksjonelle testen på Tokvam sine lokaler viste at sensornodene fungerte som forventet og møtte kravene satt i problemstillingen. Måleverdier ble kontinuerlig mottatt og loggført i mottakeren og måleverdiene ble representert grafisk på PC-en og lagret til senere bruk.

Med resultatene og produkt oppnådd ble løsningen en suksess med momenter som kan utbedres.

# REFERANSER


[1] Analog Devices. Data Sheet ADXL323. 2022.

[2] Seeed Studio. Product Details XIAO ESP32C3 2022 [hentet 24.04 2023]. Tilgjengelig fra: https://www.seeedstudio.com/Seeed-XIAO-ESP32C3-p-5431.html.

[3] Maxim Integrated. Programmable Resolution 1-Wire Digital Thermometer 2019 [hentet 25.04 2023]. Tilgjengelig fra: https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf.

[4] Elektroimportøren. Shelly DS18B20 Temp. sensor [hentet 25.04 2023]. Tilgjengelig fra: https://www.elektroimportoren.no/shelly-ds18b20-temp-sensor/94402/Product.html.

[5] Seeed Studio. Grove - AHT20 I2C Industrial Grade Temperature&Humidity Sensor [hentet 25.04 2023]. Tilgjengelig fra: https://wiki.seeedstudio.com/Grove-AHT20-I2C-Industrial-Grade-Temperature&Humidity-Sensor/.

[6] Tokvam. Katalog Vår 2021 2021 [hentet 05.05 2023]. Tilgjengelig fra: https://www.felleskjopet.no/globalassets/varemerker-og-leverandoerer/maskin/tokvam/tokvam_katalog_norsk_uten_ilsbo_mek_2021_120dpi-1.pdf.

[7] Espressif. ESP32 Technical Reference Manual. 4.8 utg2022.

[8] Espressif. ESP-32 [hentet 13.04 2023]. Tilgjengelig fra: https://www.espressif.com/en/products/socs/esp32.

[9] Espressif. Chip Series Comparison [hentet 25.04 2023]. Tilgjengelig fra: https://docs.espressif.com/projects/esp-idf/en/v5.0.1/esp32/hw-reference/chip-series-comparison.html.

[10] Espressif. ESP-NOW [hentet 08.03 2023]. Tilgjengelig fra: https://www.espressif.com/en/products/software/esp-now/overview.

[11] Pasic R, Kuzmanov I, Atanasovski K. ESP-NOW communication protocol with ESP32. Journal of Universal Excellence. 2021;6(1):53-60.

[12] Mankar J, Darode C, Trivedi K, Kanoje M, Shahare P. Review of I2C protocol. International Journal of Research in Advent Technology. 2014;2(1).

[13] Campbell S. Basics of The I2C Communication Protocol [hentet 09.03 2023]. Tilgjengelig fra: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/.

[14] Wikipedia. Node-RED 2023 [oppdatert 03.12.2022; hentet 31.03 2023]. Tilgjengelig fra: https://en.wikipedia.org/wiki/Node-RED.

[15] Wikipedia. InfluxDB 2023 [oppdatert 30.03.2023; hentet 31.03 2023]. Tilgjengelig fra: https://en.wikipedia.org/wiki/InfluxDB.

[16] Wikipedia. Grafana 2023 [hentet 31.03 2023]. Tilgjengelig fra: https://en.wikipedia.org/wiki/Grafana.

[17] Arduino. About Arduino 2021 [hentet 25.04 2023]. Tilgjengelig fra: https://www.arduino.cc/en/about.

[18] Electronic Projects. OneWire Library 2007 [hentet 28.04 2023]. Tilgjengelig fra: https://www.pjrc.com/teensy/td_libs_OneWire.html.

[19] Arduino. WiFi 2023 [hentet 03.05 2023]. Tilgjengelig fra: https://www.arduino.cc/reference/en/libraries/wifi/.

[20] Randomnedtutorials. ESP32 Useful Wi-Fi Library Functions 2023 [hentet 03.05 2023]. Tilgjengelig fra: https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/.

[21] ArduinoJSON. Quickstart 2023 [hentet 10.03 2023]. Tilgjengelig fra: https://arduinojson.org/.

[22] Floyd TL. Electronic Devices Conventional Current Version. 10 utg. London: Pearson; 2018.

[23] Sensata Technologies. PTE7300 SERIES. 2021.

[24] ASAIR. AHT20 Product manuals 2019 [hentet 24.05 2023]. Tilgjengelig fra: https://files.seeedstudio.com/wiki/Grove-AHT20_I2C_Industrial_Grade_Temperature_and_Humidity_Sensor/AHT20-datasheet-2020-4-16.pdf.

[25] Wikepedia. MEMS 2023 [hentet 24.04 2023]. Tilgjengelig fra: https://en.wikipedia.org/wiki/MEMS.

[26] Adafruit. Technical Details 2022 [hentet 15.03 2023]. Tilgjengelig fra: https://www.adafruit.com/product/4097#technical-details.

[27] Rosvold KA. Kapslingsgrad 2023 [oppdatert 25.01.2023; hentet 27.03 2023]. Tilgjengelig fra: https://snl.no/kapslingsgrad.

[28] IOTStack 2023 [hentet 31.03 2023]. Tilgjengelig fra: https://sensorsiot.github.io/IOTstack/.

[29] TLV1117LV 1-A, Positive Fixed-Voltage, Low-Dropout Regulator: Texas Instruments. 2020.

[30] IKM Labroratorium. Kalibrering 2023 [hentet 10.05 2023]. Tilgjengelig fra: https://www.ikm.no/ikm-laboratorium/kalibrering/.

[31] ISO. ISO/IEC 17025 General requirements for the competence of testing and calibration laboratories 2017 [hentet 10.05 2023]. Tilgjengelig fra: https://www.iso.org/files/live/sites/isoorg/files/store/en/PUB100424.pdf.

[32] Fluke. TECHNICAL DATA 2023 [hentet 10.05 2023]. Tilgjengelig fra: https://www.fluke.com/en/product/calibration-tools/temperature-calibrators/fluke-calibration-1523/ds.

[33] Adafruit. Adafruit QT Py Datasheet. 2023.

[34] Sensata Technologies. Instructions & Safety information PTE7300 Series. 2022.

[35] Espressif. User Guide. ESP32-C3-DevKitC 2023 [hentet 24.04 2023]. Tilgjengelig fra: https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/hw-reference/esp32c3/user-guide-devkitc-02.html.

[36] Easy EDA 2023 [hentet 31.03 2023]. Tilgjengelig fra: https://easyeda.com/.

[37] ESP32-S3-WROOM-1 Datasheet 2023 [hentet 31.03 2023]. Tilgjengelig fra: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf.

[38] Nikolić B. How to add USB type C to ESP32 development board 2023 [hentet 31.03 2023]. Tilgjengelig fra: https://blnlabs.com/how-to-add-usb-type-c-to-esp32-development-board/.

[39] LCSC 2023 [Tilgjengelig fra: https://www.lcsc.com/.

[40] Hyrotechnik. MINIMESS 2022 [hentet 08.05 2023]. Tilgjengelig fra: https://www.hydrotechnik.com/fileadmin/hydrotechnik/downloads/katalog2022/flip-catalogue/index.html#page=11.

[41] Silicon Lightworks. Li-ion Voltage Analysis 2023 [hentet 08.05 2023]. Tilgjengelig fra: https://siliconlightworks.com/li-ion-voltage.

[42] Singh N, Ogunseitan OA. Disentangling the worldwide web of e-waste and climate change co-benefits. Circular Economy. 2022;1(2):100011.

[43] European Commission. Conflict Minerals Regulation: The regulation explained [hentet 25.04 2023]. Tilgjengelig fra: https://policy.trade.ec.europa.eu/development-and-sustainability/conflict-minerals-regulation/regulation-explained_en.

[44] FN. Industri, innovasjon og infrastruktur 2023 [oppdatert 31.01.2023; hentet 25.04 2023]. Tilgjengelig fra: https://www.fn.no/om-fn/fns-baerekraftsmaal/industri-innovasjon-og-infrastruktur.

# VEDLEGG

**Vedlegg 1: Kode mottaker:**

```
#include <Arduino.h>
#include <esp_now.h>
#include <WiFi.h>

//String for å lagre mottatt data
String data;

// callback funksjon som aktiveres ved mottatt data
void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int
len) {

  for (int i = 0; i < len; i++) {
    data += (char)incomingData[i];
  }
  //skriver ut mottatt data via USB
  Serial.println(String(data));

  //Slett innhold i data
  data = "";
}

void setup() {
  //start serial monitor
  Serial.begin(115200);
  while(!Serial);

  //Sett enhet som Wi-Fi stasjon
  WiFi.mode(WIFI_STA);

  //start ESP-NOW
  esp_now_init();

  //callback ved mottatt data
  esp_now_register_recv_cb(OnDataRecv);
}

void loop() {

}
```

**Vedlegg 2: Kode sensornode 1**

```
#include <esp_now.h>
#include <WiFi.h>
#include <Wire.h>
#include <AHT20.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <ArduinoJson.h>

// Mottaker MAC adresse og navn
uint8_t broadcastAddress[] = [8 0x03, 0xED, 0x7C];
#define deviceName (101)

String data;
AHT20 aht20;

// struktur for å lagre mottaker info
esp_now_peer_info_t peerInfo;

// callback ved sendt data
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  //Serial.print("\r\nLast Packet Send Status:\t");
   //Serial.println(status  ==  ESP_NOW_SEND_SUCCESS  ?  "Delivery  Success"  :
"Delivery Fail");
}

// GPIO hvor DS18B20 er tilkoblet
const int oneWireBus = 4;
// Setup oneWire og DallasTemperature
OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);

void setup() {

  // Initialiser Serial Monitor
  Serial.begin(115200);
  // Start DS18B20 sensor
  sensors.begin();
  Wire.begin(); //Koble til I2C bus
  //Sjekk for AHT20 tilkobling
  if (aht20.begin() == false)
  {
    Serial.println("AHT20 not detected.");
    while (1);
  }
  // Sett enhet som Wi-Fi stasjon
```

```
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Callback ved sendt data
  esp_now_register_send_cb(OnDataSent);

  // Lagrer MAC adresse til mottaker
  memset(&peerInfo, 0, sizeof(peerInfo));
  for (int ii = 0; ii < 6; ++ii )
  {
    peerInfo.peer_addr[ii] = (uint8_t) broadcastAddress[ii];
  }

  // WiFi kanal og kryptering
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // Legg til mottaker
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }
}

void loop() {
  sensors.requestTemperatures();

  // opprett json dokument
  StaticJsonDocument <200> doc;

  // Legg til data i json dokument
  JsonObject obj = doc.createNestedObject();
  obj["sensor"] = deviceName;
  obj["LuftTemp"] = (aht20.getTemperature());
  obj["LuftFukt"] = (aht20.getHumidity());
  obj["OljeTemp"] = (sensors.getTempCByIndex(0));

  // Konverter json dokument til string
  serializeJson(doc, data);

  Serial.println(data.c_str());
```

```
  // Send data med ESP-NOW
  esp_now_send(broadcastAddress, (uint8_t *) data.c_str(), data.length());

  // Slett data
  data = "";
  delay(100);
}
```

**Vedlegg 3: Kode Sensornode 2**

```
#include <Arduino.h>
#include <esp_now.h>
#include <WiFi.h>
#include <PTE7300_I2C.h>
#include <ArduinoJson.h>

// Mottaker adresse og navn
uint8_t broadcastAddress[] = {0xE8, 0xDB, 0x84, 0x03, 0xED, 0x7C};
#define deviceName (103)
String data;

// initialiser sensor
PTE7300_I2C mySensor;
int16_t DSP_S;
int16_t DSP_T;
int DSP_T_int;

// struktur for å lagre mottaker info
esp_now_peer_info_t peerInfo;

// callback ved sendt data
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  //Serial.print("\r\nLast Packet Send Status:\t");
  //Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" :
"Delivery Fail");
}

void setup() {
  // Initialiser Serial Monitor
  Serial.begin(115200);
  bool status;
  // Sett enhet som Wi-Fi stasjon
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Lagrer MAC adresse til mottaker
  esp_now_peer_info_t peerInfo;
  memset(&peerInfo, 0, sizeof(peerInfo));
  for (int ii = 0; ii < 6; ++ii )
  {
```

```
    peerInfo.peer_addr[ii] = (uint8_t) broadcastAddress[ii];
  }

  // WiFi kanal og kryptering
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // Legg til mottaker
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }

  // callback ved sendt data
  esp_now_register_send_cb(OnDataSent);
}

void loop() {
  mySensor.CRC(true);
  DSP_S = mySensor.readDSP_S();
  //gjør om fra int16 til bar
  int mappedBar = map(DSP_S, -16000, 16000, 0, 400);

  //opprett json dokument
  StaticJsonDocument <200> doc;

  //legg til data i json dokument
  JsonObject obj = doc.createNestedObject();
  obj["sensor"] = deviceName;
  obj["Pressure"] = (mappedBar);
  serializeJson(doc, data);

  Serial.println(data.c_str());

  // Send data med ESP-NOW
  esp_now_send(broadcastAddress, (uint8_t *) data.c_str(), data.length());

  //slett data
  data = "";
  delay(100);
}
```

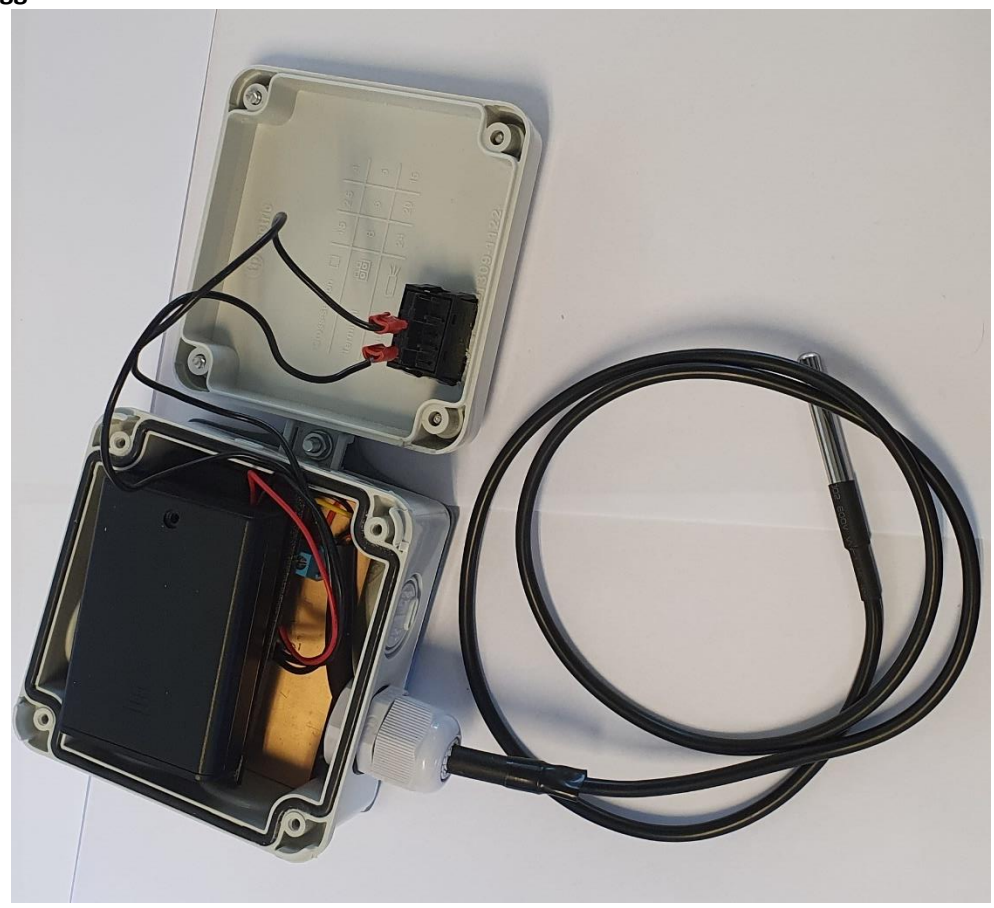
**Vedlegg 4: Kode Sensornode 3**

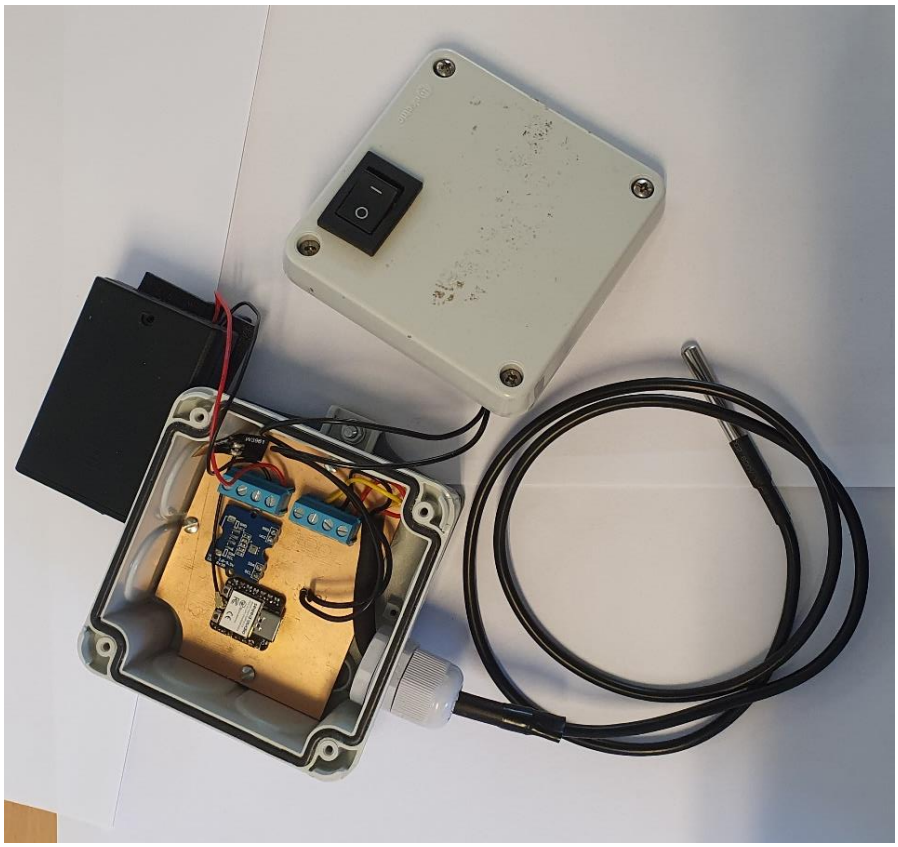
```
#include <Wire.h>
#include <math.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL343.h>
#include <esp_now.h>
#include <WiFi.h>
#include <ArduinoJson.h>


#define ADXL343_SDA 8
#define ADXL343_SCL 9

//Definerer faktor for konvertering fra m/s^2 til g
#define MS2_TO_G 0.101971621

double VibrX = 0;
double VibrY = 0;
double VibrZ = 0;
String data;

Adafruit_ADXL343 accel = Adafruit_ADXL343(12345);

// mottaker MAC adresse og navn
uint8_t broadcastAddress[] = {0xE8, 0xDB, 0x84, 0x03, 0xED, 0x7C};
#define deviceName (102)

// Create peer interface
esp_now_peer_info_t peerInfo;

// callback when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  //Serial.print("\r\nLast Packet Send Status:\t");
   //Serial.println(status  ==  ESP_NOW_SEND_SUCCESS  ?  "Delivery  Success"  :
"Delivery Fail");
}

void setup()
{
  // Init Serial Monitor
  Serial.begin(115200);
  while (!Serial);

  // Init ADXL343 og sett måleområde til 16g
  accel.setRange(ADXL343_RANGE_16_G);
  accel.begin();
```

```
  bool status;
  // Sett enhet som Wi-Fi stasjon
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Lagrer MAC adresse til mottaker
  esp_now_peer_info_t peerInfo;
  memset(&peerInfo, 0, sizeof(peerInfo));
  for (int ii = 0; ii < 6; ++ii )
  {
    peerInfo.peer_addr[ii] = (uint8_t) broadcastAddress[ii];
  }

  // WiFi kanal og kryptering
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // legg til mottaker
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }
  // Callback ved sendt data
  esp_now_register_send_cb(OnDataSent);
}


void loop()
{
  // Leser akselerasjon
  sensors_event_t event;
  accel.getEvent(&event);

  // Konverterer akselerasjon fra m/s^2 til g
  VibrX=event.acceleration.x*MS2_TO_G;
  VibrY=event.acceleration.y*MS2_TO_G;
  VibrZ=event.acceleration.z*MS2_TO_G;

  // Beregner total akselerasjon
  double Vibrasjon = sqrt(VibrX * VibrX + VibrY * VibrY + VibrZ * VibrZ);
```

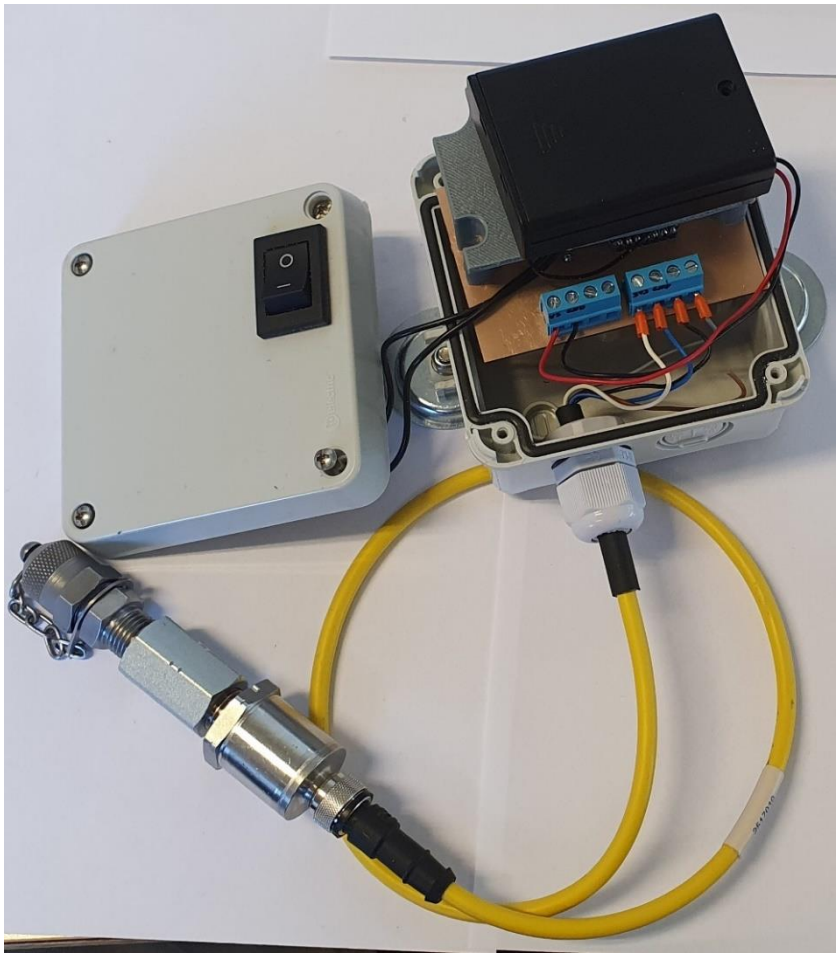
```
  // opretter json dokument
  StaticJsonDocument <200> doc;

  // legger til data i json dokument
  JsonObject obj = doc.createNestedObject();
  obj["sensor"] = deviceName;
  obj["Vibrasjon X"] = VibrX;
  obj["Vibrasjon Y"] = VibrY;
  obj["Vibrasjon Z"] = VibrZ;
  obj["Vibrasjon"] = Vibrasjon;

  // konverterer json dokument til string
  serializeJson(doc, data);

  Serial.println(data.c_str());

  // Send message via ESP-NOW
  esp_now_send(broadcastAddress, (uint8_t *) data.c_str(), data.length());

  // slett data
  data = "";

  delay(100);
}
```

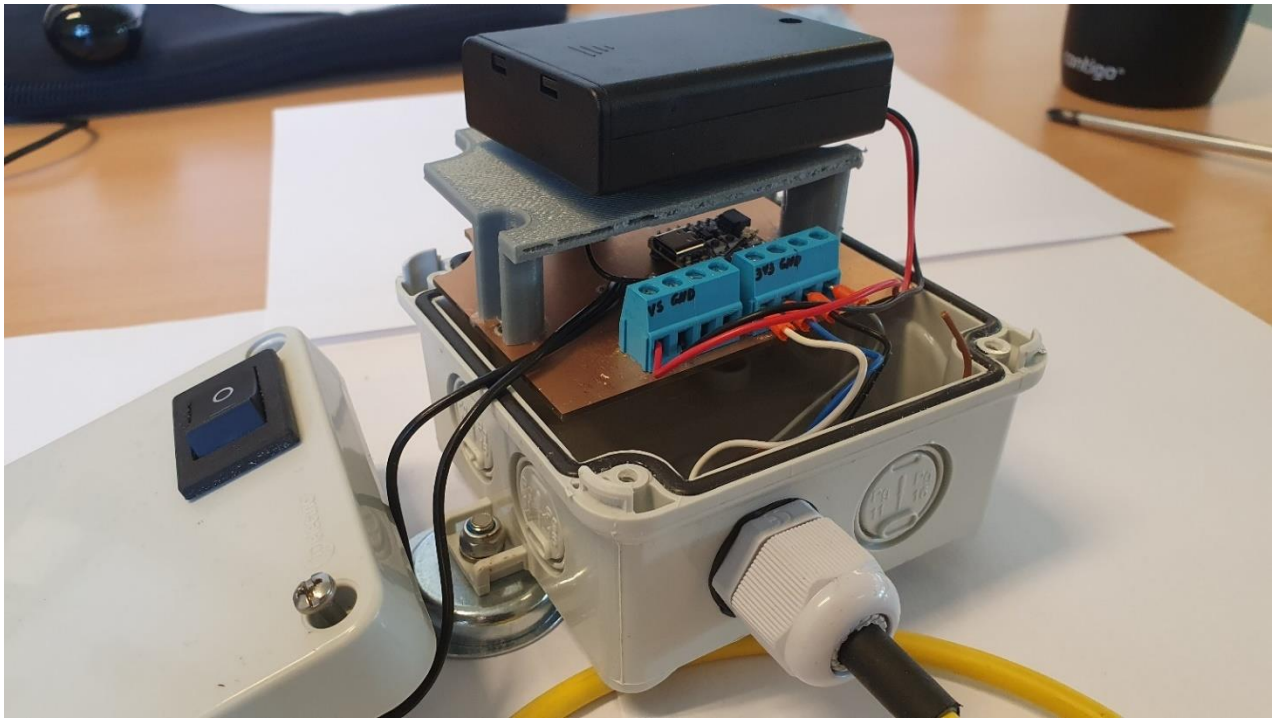
**Vedlegg 5: Bilder av sensornode 1**



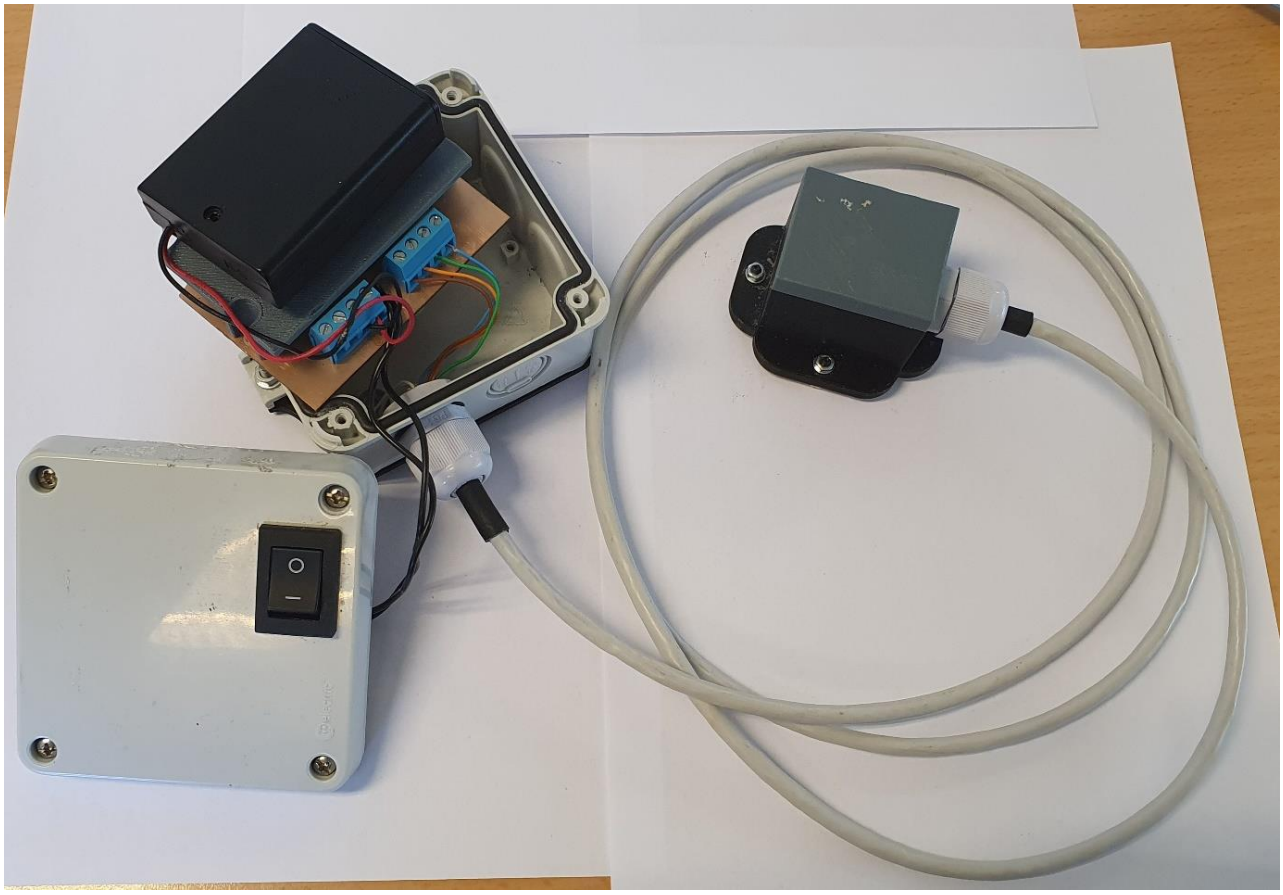


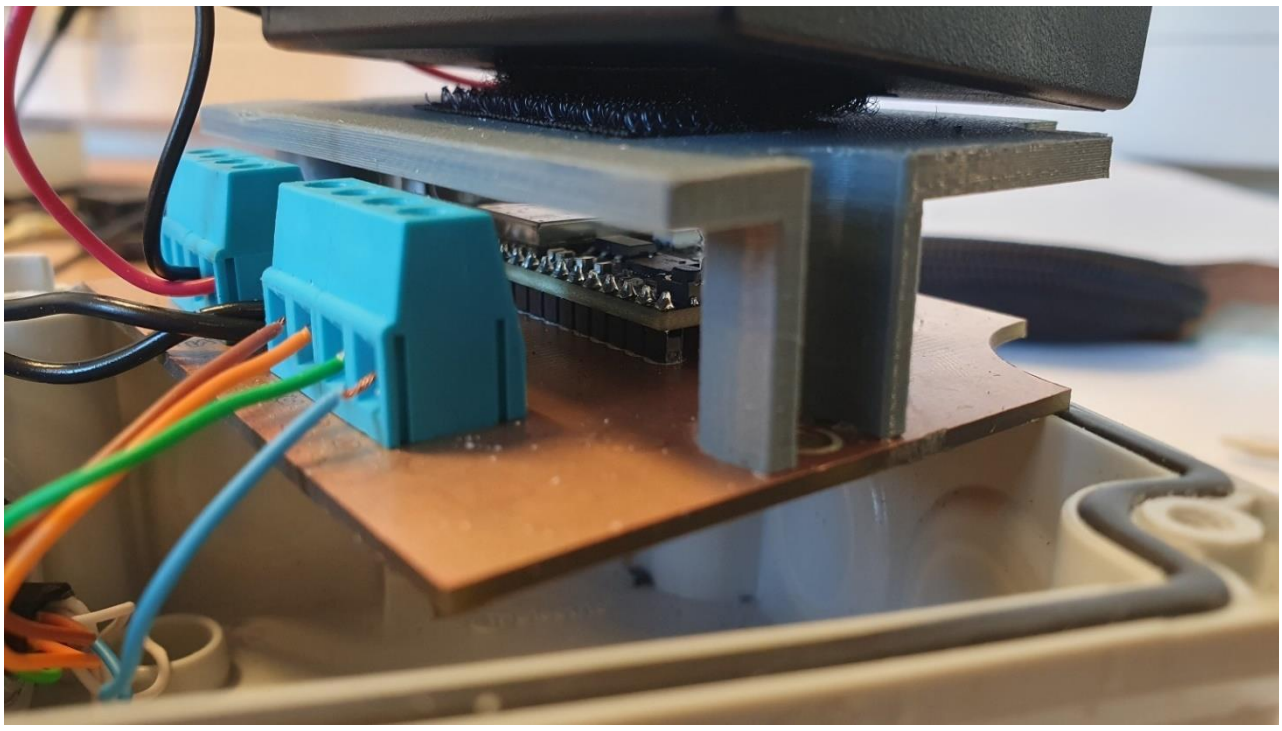
**Vedlegg 6: bilder av sensornode 2**

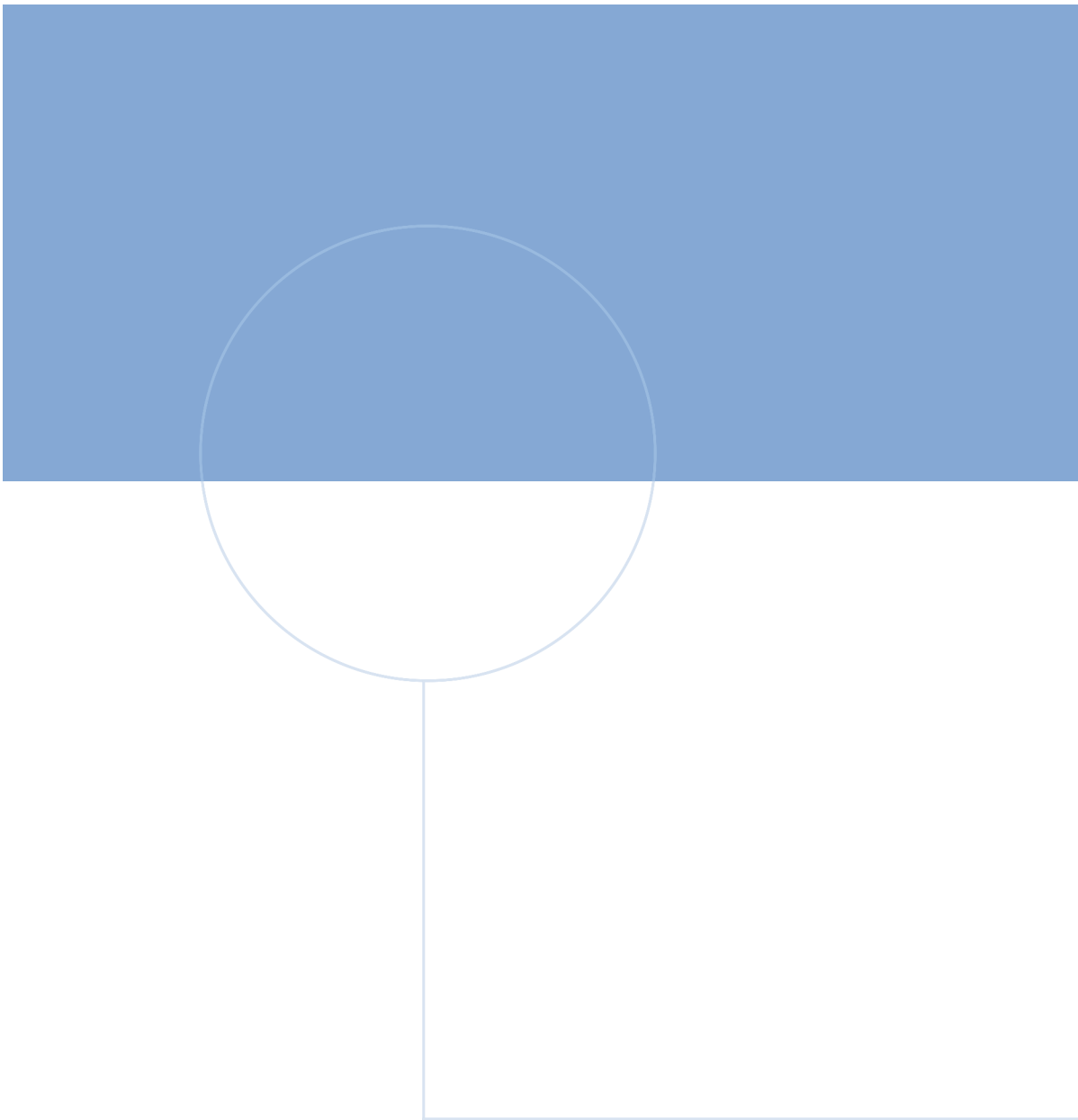




**Vedlegg 7: bilder av sensornode 3**







NTNU
Kunnskap for en bedre verden

Tokvam®