Bacheloroppgave

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

Hansgård, Ivar
Marthinsen, Marius

Modulære soldrevne sensorbokser med tilkobling til ett trådløst LoRaWAN nettverk

Bacheloroppgave i Elektro
Veileder: Snilsberg, Rolf Kristian
Mai 2023

NTNU
Kunnskap for en bedre verden

Hansgård, Ivar
Marthinsen, Marius

Modulære soldrevne sensorbokser med tilkobling til ett trådløst LoRaWAN nettverk

Bacheloroppgave i Elektro
Veileder: Snilsberg, Rolf Kristian
Mai 2023

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

NTNU
Kunnskap for en bedre verden

Modulære soldrevne bokser for innsamling av sensordata tilknyttet et trådløst LoraWAN nettverk

Ivar Hansgård og Marius Marthinsen

22/05/23

NTNU

Tittelside Bacheloroppgave BIELEKTRO

Oppgavetittel (norsk og engelsk):
Modulære soldrevne sensorbokser med tilkobling til ett trådløst LoRaWAN nettverk

Modular sunpowered sensorboxes connected to a wireless LoRaWAN network

Forfattere:
Ivar Hansgård
Marius Marthinsen

Prosjektnummer:	E2324
Innleveringsdato:	22.05.2021
Gradering:	[x] åpen [ ] lukket

Studium:	Elektroingeniør - BIELEKTRO
Studieretning:	Elektronikk og Sensorsystemer
Veileder internt:	Rolf Kristian Snilsberg
Institutt:	Institutt for elektroniske systemer
Oppdragsgiver:	SINTEF Industri
Kontaktperson:	Bendik Sægrov, bendik.sagrov@sintef.no, 98283890

Sammendrag (norsk og engelsk):
SINTEF har et prosjekt hvor de ser på virkningen av solenergiproduksjon i landbruket. Man ønsker å se på om det er mulig å kombinere elektrisitetsproduksjon samtidig med daglig gårdsdrift. Som en del av prosjektet har man plassert vertikale solcellepanel på Skjetlein gård utenfor Trondheim. Der skal man forske på virkningen av lys/skygge og refleksjoner fra solcellepanelene mot plantevekst. Det har allerede blitt simulert i datamaskinen hva slags påvirkning dette skal ha på planteveksten. I denne bacheloroppgaven skal prosjektgruppen lage et modulert system for innhenting av sensor data for å verifisere disse resultatene. Det ble også laget et dokumentert rammeverk sånn at SINTEF kan sette opp og lage flere sensorbokser eller knytte til nye plasseringer i fremtidige prosjekter.

SINTEF has a project where they are looking at the impact of solar energy production in agriculture. They want to look at whether it is possible to combine electricity production at the same time as daily farm operations. As part of the project, vertical solar panels have been placed on Skjetlein farm outside Trondheim. Research will be done on the effect of light/shadow and reflections from the solar panels on plant growth. It has already been simulated in the computer what kind of effect this should have on the plant growth. In this bachelor's thesis, the project group will create a modular system for acquiring sensor data to verify these results. A documented framework was also created so that SINTEF can set up and create more sensor boxes or link new locations in future projects.

Stikkord norsk:	**Stikkord engelsk:**
Solpanel, elektronikk, sensorer sensorbokser, LoRaWAN, dokumentasjon, programmering	Solarpanels, electronics, sensors, sensorboxes, LoRaWAN, documentation, programming

Annerkjennelser

Denne rapporten konkluderer et 3.år langt elektroingeniørstudium med spesialisering innenfor elektronikk og sensorsystemer. Rapporten er skrevet i vårsemesteret 2023 for SINTEF Industri med målet å lage et modulært system bestående av soldrevne sensorbokser.

Prosjektgruppen vil takke Bendik Sægrov-Sorte ved SINTEF Industri for å ha gjort tilgjengelig utstyr og verksted som har blitt brukt til å jobbe med oppgaven. Gruppen vil også takke Gaute Stokkan og Harry Malhi for tips om hvilke sensorer som kan være lurt å ha med. Til slutt vil gruppen også takke Rolf Kristian Snilsberg som veileder for prosjektet.

Signaturer

Ivar Hansgård
ivarhan@stud.ntnu.no
Elektronikk og Sensorsystemer

Marius Marthinsen
marmarth@stud.ntnu.no
Elektronikk og Sensorsystemer

Innhold

Figurer

Tabeller

Ordliste

Node/Sensorboks/Sensornode er en boks med ett CubeCell dev board, 4 plugger for sensorer, to lyssensorer, batteri og solcellepanel.

Systemet beskriver hele systemet fra noden til du kan se dataen på grafana. Systemet inneholder noder, en gateway, the things stack, webhook tjeneren, databasen og grafana.

ChirpStack er en åpen kildekode versjon av The Things Stack.

CubeCell er Cubecell-dev-board.v2.

Akronymer

IDE - Intergrated Development Environment.

Vevapplikasjon - program som kan kjøres i nettleser.

DBMS - Database Management Systems.

DBMSORDBMS - Object Relational Database Management Systems.

GPIO - General-purpose input/output.

A – Ampere.

Ah - Ampere hours / Ampere timer.

W - Watt.

V - Volt.

Kapittel 1

Innledning

1.1 Bakgrunn

I framtiden vil behovet for høyere energiproduksjon og effektivisering av arealbruk øke. For at Norge skal kunne gå igjennom et grønt skifte er det behov for stor utbygging av grønn energi som vindkraft og solenergi. Energikommisjonen kom nylig med et estimat om at innen 2030 kan man få til en utbygging i størrelsesorden 5-10 TWh med solenergi [1]. De anbefaler at noe av denne utbyggingen skjer som kombinasjonsbruk sammen med jordbruk. Gode kombinasjons løsninger kan bidra til å minke konfliktnivået ved å ivareta ulike interesser noe som man har sett eksempler på ikke har blitt gjort ved utbygging av for eksempel vindkraft. Bakgrunnen for bacheloroppgaven er at SINTEF har et prosjekt hvor de ser på virkningen av solenergiproduksjon i landbruket. Man ønsker å se på om det er mulig å kombinere elektrisitetsproduksjon samtidig med daglig gårdsdrift. Som en del av prosjektet har de plassert vertikale solcellepanel på Skjetlein gård utenfor Trondheim. Der skal de forske på virkningen av lys/skygge og refleksjoner fra solcellepanelene mot plantevekst. Det har allerede blitt simulert i datamaskinen hva slags påvirkning dette skal ha på planteveksten, men for å verifisere resultatene ønsker SINTEF et modulært system for innhenting av sensordata tilknyttet effekten solcellepanelene har på området rundt. Hvis man vitenskapelig kan bekrefte at solcellepanelene har minimal effekt på jordbruket kan man bidra til mindre konflikt ved framtidig utbygging og skape et bedre grunnlag for fremtidig lovgivning rundt kombinasjonen av landbruk og solenergiproduksjon.

1.2 Oppgaven

Før prosjektet kunne starte måtte oppdragsgiver og prosjektdeltakere bli enige om kravene til sensorene, sensornodene og visualiseringen av dataene. Dette ble gjort ved først å ta utgangspunkt i den originale oppdragsbeskrivelsen og så ble det laget en kravspesifikasjon gjennom en samtale med oppdragsgiver hvor begge parter kunne spesifisere hva som skulle og kunne være med.

1.2.1 Kravene ut ifra oppgavebeskrivelse:

> Vi ønsker at studentene skal lage et dokumentert rammeverk/fremgangsmåte bestående av modulære soldrevne bokser for innsamling av sensordata tilknyttet et trådløst LoraWAN nettverk. Sensorer kan være temperatur, fukt, pyranometer (lysinnstråling), pH, nitrogen, osv. Studentene skal sette opp LoRaWAN nettverket og ordne tilknytning til SINTEF-database i sky og lage IP67/68 soldrevne datainnsamlingsbokser til bruk i felt. Sensordataene skal lagres i SINTEF-database og gjøres tilgjengelig for både SINTEF-forskere og eksterne kunder, gjerne i form av en nettside med presentasjon av dataene f.eks. ved hjelp av grafana eller tilsvarende. Studentene skal fokusere på Skjetleincaset, men vi ønsker at det skal være enkelt å bygge nye bokser og knytte til nye plasseringer i fremtidige prosjekter

1.2.2 Kravene etter samtale med oppdragsgiver:

Ferdig resultat skal inneholde

Noden:

- Node med fire kontakter for fire forskjellige sensorer.
- Noden skal tåle å stå ute i regn (minimum IP63).
- Noden skal ha solcellepanel og batteriløsning.
- Noden skal kommunisere over LoRaWAN.
- Noden skal være modulær og om sensorer eller noden selv blir ødelagt skal det være lett å bytte ut deler.
- Det skal være mulig å implementere nye sensorer eller bytte sensorer.
- Sensor målinger kan være: *temperatur i luft og/eller i jorden* , *fukt i luft og/eller i jorden*, *lysinnstråling*, *ph, nitrogen, innhold i jord*, eller *vind*.

Database, visualisering:

- Visualisere innsamlet data på en nettside.
- Lagring av innsamlet data i SINTEF database.

Veiledning og dokumentasjon:

- Instruksjon og veiledning på hvordan å lage nye/flere noder.
- Instruksjon og veiledning hvordan å bytte ut sensorer / legge til nye.
- Instruksjon og veiledning på hvordan å bygge hele systemet fra koding av mikrokontroller til oppsett av LoRaWAN Gateway til serverdel og grafana.

1.3 Rapportens oppbygging

Rapporten er delt inn i fem hoveddeler. Teori, metode, resultat, diskusjon og konklusjon. Delene skal gi nok kunnskap til leseren sånn at han kan forstå og gjenskape resultatene. Det er forventet at leseren har tidligere kunnskaper om elektroteknikk, sensorer og koding fra 3.året bachelor i elektronikk.

Teori inneholder alt av teknisk informasjon som trengs for å forstå de senere deler av rapporten.

Metode delen tar for seg utførelse av prosjektet og hvordan de ulike delene av systemet er designet. Kapittelet tar også for seg hvilke komponenter som er tatt i bruk og hvorfor nettopp disse er brukt. Metode inneholder også hva slags programvare som blir brukt i prosjektet. Kapittelet er videre delt inni i følgende underdeler. En maskinvare del som omhandler de fysiske delene til prosjektet. En programvare del som forklarer all programvare utviklet til systemet. En klient del som omhandler de delene brukeren samhandler med.

Resultat delen av rapporten viser det endelig resultatet av prosjektet i form av bilder, tabeller og figurer samt lenker til dokumentasjon på GitBook og filer på GitHub.

Diskusjon delen av rapporten inneholder refleksjon over hvordan gruppen har besvart oppgaven i tillegg til feilkilder og forslag til hvilke forbedringer som kan gjøres på systemet.

Til slutt kommer konklusjons delen som oppsummerer rapporten og sier hva gruppen mener om det endelige resultatet.

Kapittel 2

Teori

2.1 Maskinvare del

2.1.1 Maskinvare

Maskinvare er de fysiske delene en teknologisk innretning består av. Begrepet kan også brukes om hele enheten. Eksempler på maskinvare kan være harddisker, minnebrikker eller mikrokontrollere. [2]

2.1.2 Mikrokontroller

En mikrokontroller er en integrert krets som i tillegg til en prosessor inneholder noen funksjonsblokker, for eksempel minne og inn–ut-enheter, på den samme kretsen[3].

2.1.3 LoRaWAN

LoRaWAN er en forkortelse for Long Range Wireless Area Network. LoraWAN er et MAC (media access controll) lag bygget på toppen av LoRa Modulasjon. Programvarelaget definerer hvordan enheter skal bruke LoRa maskinvare ved å foreksempel definere formatet på meldinger som sendes.[4]

LoRa

LoRa (Long Range) er en trådløs modulasjonsteknikk som bruker Chirp Spread Specturm teknikk for å modulere signalene. [4]. Chrip Spread Spectrum er en lang distanse radio frekvens teknologi for trådløs kommunikasjon. Teknologien bruker en bredbåndsmodulasjonsteknikk som lager lineære frekvensmodulerte signaler som blir kalt «chrips» eller «chirp pulses».[5]

2.1.4 CubeCell – Dev-Board (V2)

CubeCell – Dev-Board (V2) er en Arduino kompatibel mikrokontroller med LoRaWAN 1.0.2 laget av Heltec Automation. Mikrokontrolleren har muligheten til direkte kopling av solcellepanel og litium batteri. Den har et lavt strømforbruk og er designet for å være node-delen i et LoRaWAN-nettverk.[6]

2.1.5 Arduino IDE

Arduino IDE er en IDE som er originalt laget for å brukes sammen med Arduino utviklingsbrettet. Arduino IDE har et bredt utvalg av biblioteker. I tillegg lager produsenter av sensorer sine egne biblioteker slik at kunder kan lett benytte sensorene i Arduino IDE og lage prosjekter med disse. Arduino IDE er en åpen kildekode programvare[7]. Flere mikrokontrollere kan bruke Arduino IDE om de er Arduino kompatibel.

2.1.6 Åpen kildekode

Åpen kildekode programvare som betyr at brukere har rettigheter til å bruke, studere, endre og distribuere programvaren og dens kildekode til hvem som helst og for enhver.[8]

2.1.7 I2C

I2C (Inter-integrated Circuit) er kommunikasjonsprotokoll. Den kan ha flere master enheter (controller) og flere slave enheter (target). Den bruker to ledninger eller to koblingspunkter fra master til slave SCL (serial Clock) og SDA (Serial Data). Disse er koblet til en mostand som er koblet til spenningskilden. For at kommunikasjonen skal starte sender master først en adresse til alle slavene i kretsen. Slaven svarer så ved å dra SDA lav. Master sender så til slave om master skal motta eller sende data til slaven. Deretter sender eller mottar master/slave datapakker på 8-bit, mottaker svarer med enten en bekreftelse eller en ikke bekreftelse bit. Master sender et så en stopp betingelse ved å dra SCL høy deretter SDA høy. Det er en begrensning på hastighet og avstand fra master til slave på denne kommunikasjonsprotokollen ettersom den bekrefter hver datapakke sendt/mottatt.[9]

2.1.8 Bit, bits, byte.

En bit er lagringsform i enten ‘0’ eller ‘1’, ‘av’ eller ‘på’. Bits er flere bit sammenlagt for eksempel, en bit vil være 0 eller 1, to bits vil være 00, 01, 10 eller 11. En byte er enhet av digital informasjon og er 8-bits sammenlagt, det fins 255 kombinasjoner per byte.[10]

2.1.9 3D-Printing

3D-printing eller additiv produksjon er en produksjons teknikk som bruker et skriverhode og smeltet plastikk til å bygge opp lag til å lage en tredimensjonal figur eller modell. Prosessen er fleksibel blir benyttet til å lage prototyper og gjenstander i små kvanta.[11]

Slicer

Innen 3D-utskrift er en slicer en programvare som brukes for å konvertere eller dele opp en 3D-objektfil til spesifikke instruksjoner for skriveren. Et vanlig eksempel er å konvertere en 3D-modell i STL-format til skriverkommandoer i g-kodeformat.[12]

2.1.10 Gateway

En gateway (egentlig engelsk, direkte oversatt til norsk: inngangsport) er en maskinvarekomponent som brukes i sammenkobling mellom ulike nettverk for telekommunikasjon eller trådløs datakommunikasjon. I et kommunikasjonsnettverk er gatewayen en node som utgjør et grensesnitt mot et annet nettverk som bruker andre protokoller. En gateway gir mulighet til å bygge bro mellom enheter som befinner seg i felten (arbeidsplassen, hjemmet, etc.) og nettskyen, der data er lagret.[13]

2.2 Tjener del

2.2.1 Tjener

Tjener (også kjent som server) er programmvare som tilbyr en eller flere tjenester til andre datamaskiner over et datanettverk. Begrepet kan også brukes om selve maskinvaren programmvaren kjører på. [14]

2.2.2 The Things Network

The Things Network er en LoRaWAN nettverks tjener utviklet av The Things Network. The Things Network inneholder ett komplett sett med tjenester for å sette opp ett eget LoRaWAN nettverk.[15]

2.2.3 JavaScript

JavaScript er et høynivå programmeringsspråk som brukes sammen med HTML og CSS for å lage moderne nettsider. Språket kan også brukes for å lage komplekse webapplikasjoner og som et rent skripting språk. [16]

2.2.4 JSON

JSON eller Java Script Object Notation er en tekst basert standard for å formatere meldinger som brukes i datautveksling. Den er opprinnelig avledet fra JavaScript for å representere enkle datastrukturer. Standarden er imidlertid uavhengig av JavaScript eller andre programmeringsspråk.[17]

2.2.5 Webhook

En webhook er ett brukerdefinert HTTP-tilbakekall som vanligvis blir aktivert ved en eller annen hending på en nettside. Når hendingen skjer sendes en HTTP-POST forespørsel til nettaddressen som er konfigurert for webhooken. Formatet er vanligvis JSON. Ved hjelp av en webhook kan man konfigurere hendinger på en nettside til å forårsake en hending på en annen nettside. [18]

2.2.6 Node JS

NodeJS er et åpent kildekode asynkron kjøretidsmiljø for utførelse av JavaScriptkode på servere og nettverksapplikasjoner. Det at NodeJS er asynkront betyr at hele programmet kjører i en tråd, men når en I/O opperasjon tar lang tid venter ikke NodeJS på at den skal bli ferdig og eksekverer heller neste kodelinje. Når I/Ooperasjonen returnerer utløser den en callback-funksjon som så kan prosessere resultatet. På grunn av denne egenskapen blir NodeJS i all hovvedsak brukt i tjener programmvare som er I/O-intensiv, der hastighet og skalerbarhet er viktige ikke funksjonelle krav. Dette kan foreksempel være en webserver som legger inn data i en database. [19]

2.2.7 Database

En database er et begrep for strukturert samling av relaterte data. Et databasesystem blir delt opp i to deler. Databasehåndertingssystemet (DBMS) som er programmvare som administrerer systemet og sikrer at endringer ikke fører til motsetninger eller feil. Den andre delen er den selve databasen som er lagret data. Eksempler på databasehåndertingssystemer er MySQL, Oracle Database, MongoDB og PostgreSQL [20]

PostgreSQL

PostgreSQL et fritt objektrelasjonelt databasesystem (ORDBMS). Systemet er basert på POSTGRES og støtter SQL92 og SQL99 Standardene. [21]

2.3 Klient del

2.3.1 Gitbook

Gitbook er en dokumentasjonsplattform. På plattformen kan man invitere og samarbeide med flere samtidig samt å publisere eller dele innholdet med bestemte eller flere. [22]

2.3.2 Grafana

Grafana er en vevapplikasjon for å interaktiv visualisering av statistikk. Det betyr at man kan visualisere data fra en database i form av grafer eller diagrammer.[23]

Grafana Cloud

Tjeneste fra Grafana hvor de kjører Grafana vevapplikasjonen for deg. [24]

Kapittel 3

Metode

3.1 Utstyr

3.1.1 Komponenter

Komponenter til ett sett med sensorer og en node, samt gateway for tilkobling til The Things Network.

Node komponenter

Komponent	Delnummer	Antall enheter
CubeCell - Dev-Board (V2)	ZC-149-868	1
Adafruit TCA9548A I2C Multiplexer	2717	1
Wiska CLWIB 2 120x160x70mm boks	CLWIB 2	1
TE Connectivity hann plugg	T4111501041-000	4
TE Connectivity hunn plugg	T4131512041-000	4
Seeed studio 0.5W Sol Panel 55x70	313070004	2
Ansmann lithium-ion batteri 3.6V 2600m	1307-0000	2
Keystone, 2x18650 batteri holder	1049	1
M5 mutter	BN 628 M5	2
Adafruit Perma-Proto Quarter-sized Breadboard	1608	1
Adafruit Perma-Proto Half-sized Breadboard	1609	1
Adafruit ADS1115 16-Bit ADC	1085	1
BD679	BD679	1
Motstand 10kohm	MBB02070C1002FCT00	1
RE520-HP, FR-2 Stripboard	RE520-HP	1

Tabell 3.1: Node komponenter

Sensor komponenter

Komponent	Delnummer	Antall enheter
Adafruit AS7341	4698	1
Adafruit BH1750	4681	1
Seeed studio SCD30	101020634	1
DFRobot SEN0308	SEN0308	1
Alpha Wire Cat5 Ethernet Kabel 30m	9854C-30,5M	1
RS PRO Type J Thermoelement	219-4725	1
Seed studio MCP9600 I2C Thermoelement forsterker	101020594	1
Glide bryter 5A, 28V	734-7292	1
RS PRO 2-pin PCB skruterminal	897-1332	1
Adafruit I2C STEMMA Cable JST PH	3955	2

Tabell 3.2: Sensor komponenter

Ekstra komponenter

Komponent	Delnummer	Antall enheter
HT-M02 Edge LoRa Gateway (V2)	ZC-183-868	1
HellermannTyton Krympestrømpe sett, 3:1 Ratio	380-03001	1
Forskjellig ledning 7.6m, SparkFun Electronics	PRT-11367	1
HY4060 Epoxy	2298837	2
RS PRO krympestrømpe, 25.4mm diameter	700-4668	1

Tabell 3.3: Ekstra komponenter

3.1.2 Programvare

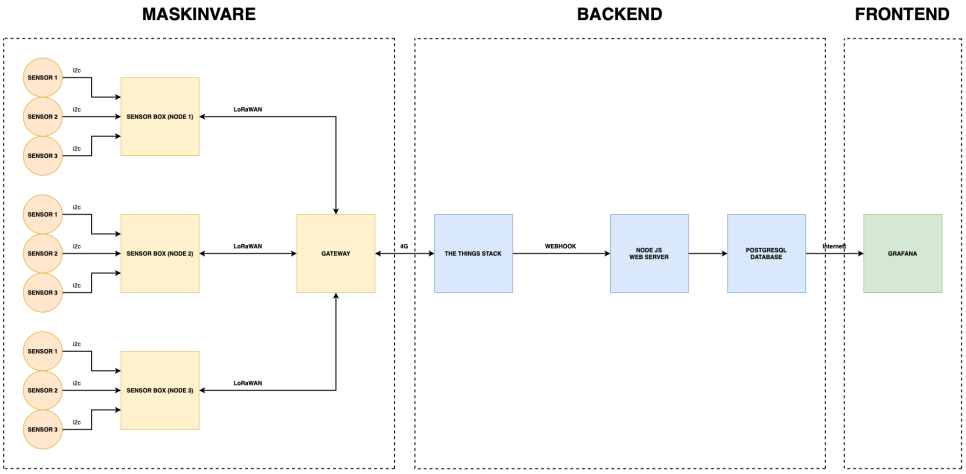
Programvare brukt under prosjektet for å designe, kode/konfigurere, og rapportere.

Program	**Utvikler**	**Funksjon**
Fusion 360	Autodesk	3D-moddelering
Bambu Studio	Bambu Lab	3D-printing slicer
drawio.com	JGraph Ltd	Skjema tegner
Arduino IDE	Arduino	IDE
VSCode	Microsoft	IDE
DataGrip	JetBrains	Redigering/oppsett av database
NodeJS	OpenJS Foundation	Webserver
PostgreSQL	PostgreSQL Global Development Group	Database
The Things Network	Frivilige	LoRaWAN nettverks server
Excel	Microsoft	Regneark program
Teams	Microsoft	Deling av dokumenter og filer
Overelaf	Digital Science UK	Rapport skriving
GitBook	GitBook INC	Dokumentasjon
Team Gantt	Team Gantt	Prosjektorganisering
Toms planner	Toms Planner	Prosjektorganisering

Tabell 3.4: Programvare

3.2 Systemets design

Ut ifra kravene funnet i samtale med oppdragsgiver er det bestemt hvordan systemet skal fungere og se ut.

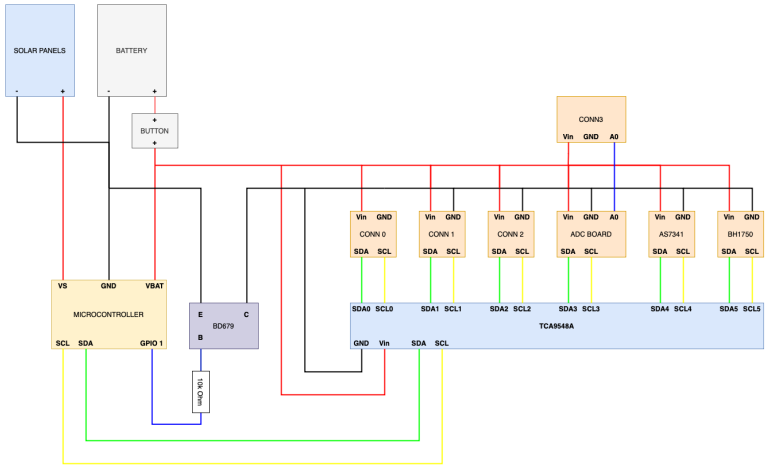


Figur 3.1: Diagram over dataflyt i systemet

Figur 3.1 viser oppsettet som er bestemt. Der ser man at de forskjellige nodene tar inn data fra sensorene via I2C kommunikasjon og sender dataen videre med LoRaWAN til Gatewayen. Gatewayen sender så dataen videre til The Things Network. Fra The Things Network blir dataen sendt videre ved hjelp av en webhook til webhook tjeneren som så setter dataen inn i databasen. Grafana kan så hente data fra databasen for å visualisere den.

3.2.1 Kretskjema

Kretsskjemaet i figur 3.2 viser oppkoblingen av noden. Kretskjemaet inneholder hvordan mikrokontrolleren er koblet opp mot multiplexeren via I2C og hvordan multiplexeren er koblet til alle pluggene og lyssensorene. Kretskjemaet viser også hvordan strøm til alle sensorer er koblet fra batteri igjennom en transistor med styring fra GPIO 1 på mikrokontrolleren for at man kan slå av sensorene når de ikke er i bruk. I tillegg er det koblet på solpanel for å lade batteriet når noden er i sovemodus. Du kan også slå av hele noden ved hjelp av en bryter ved batteriet.



Figur 3.2: Kretsskjema

3.3 Maskinvare

Maskinvare avsnittet tar for seg de fysiske delene benyttet i prosjektet. Den går igjennom hvorfor valg er tatt for at systemet skal møte kravene gitt fra oppdragsgiver.

3.3.1 I2C

Et av kravene var at systemet skulle være modulært og om noe blir ødelagt skal det være mulig å reparere eller bytte ut sensoren relativt billig og enkelt. I2C ble valgt som kommunikasjonsprotokoll for sensorene fordi det er en vanlig kommunikasjonsprotokoll som er implementert på de fleste sensorer. Protokollen er også lett å implementere ved bruk av Arduino sitt wire bibliotek. I2C enheter har vanligvis en standard adresse så hvis sensorer blir ødelagt kan man bytte ut sensoren med en ny en uten å endre noe i kode eller konfigurere nodene på nytt. En ulempe med I2C er at protokollen har en begrensing på bare 127 adresser så det kan hende man skaffer to sensorer med samme adresse. Dette kan fikses ved at de fleste sensorer har mulighet for å endre adresse ved å sette en eller flere pinner høy eller lav.

3.3.2 Valg av deler til node

Analog til digital omformer

Noen sensorer er analoge og kan derfor ikke kommunisere over I2C. CubeCellen har en innebygd analog til digital omformer, men oppløsningen er ikke tilfredsstillende. For å få høyere oppløsning og for å beholde samme standard på alle sensorer blir den analoge til digitale omformeren Adafruit ADS1115 benyttet. Denne omformeren har en oppløsning på 16 bit som vil si at ADCen kan dele opp spenningen fra sensoren i $\frac{spenning}{2^{16}}$ steg. Omformeren tar inn analog data og bruker I2C protokollen til å sende data videre.

I2C multiplekser

CubeCellen har bare en I2C bus. Hvis man har lyst til å ha flere av samme sensortype plugget inn i en boks går ikke dette. Det blir derfor brukt en I2C multiplekser. Multiplekseren bruk i prosjektetet er en Adafruit TCA9548A. Den kan skifte mellom åtte forskjellige I2C busser. Alle pluggene på noden og de to lyssensorene er koblet inn i I2C mulitplekseren. Man kan da plugge flere av samme type sensor inn i de forskjellige pluggene og gjør systemet mer modulært.

3.3.3 Valg av batteri

Noden skal stå ute over lengre tid og har ingen tilgang til strøm. Den må derfor ha en batteriløsning som varer over lengre perioder. CubeCellen har muligheten til å koble på et litium batteri dersom batteriet er mellom 3,3V og 4,2V[6]. Batteriet

ble valgt fordi det har et innebygde sikkerhets brett og tåler lave temperaturer ettersom noden skal stå ute. Ut ifra begrensinger på plass inne i noden ble det valgt to litium batteri som ble koblet sammen i parallell for å øke kapasiteten til batteriet. Fra formelen $\frac{(100-Q)*B}{100*I} = t$ kan man beregne hvor lenge batteriet varer på en ladning. Der hvor I er strøm belastning, B er størrelse på batteriet i ampertimer og Q er prosent ladning på batteriet man vil ha igjen til slutt, den blir satt til 20% ettersom litium batterier ikke burde utlades mer [25]. Ut ifra vedlegg D.2, ser man at strømtrekket var på 0.23A, størrelsen på to batterier i parallell er 5.2Ah. $\frac{(100-20)*5,2Ah}{100*0,23A} = 18timer$. Dette er om systemet er i konstant maks strømtrekk. Det vil si at den tar inn data konstant. Man ser og at maks effektforbruk er 0.9W. Det er lagt på 10% ekstra forbruk på watt for å gi rom for feil.

Valg av solcellepanel

Ut ifra beregningene på maks effektforbruk ser man at solcellene må genere mer enn 0.9W for at batteriet skal kunne lade. På databladet til CubeCellen står det at solcellepanelene må være mellom 5.5V til 7V. Det ble derfor valgt ett solpanel på 5.5V med en maks effekt på 0.5W. Dersom man plasserer to av solcellepanelene i parallell dobler det effekten og man får 1W effekt totalt. Det blir derfor benyttet to solcellepanel og batteriet kan bli ladet.

Strømstyring til sensorer

Ved testing av systemet ble det bemerket at I2C multiplexeren mistet kommunikasjon med CubeCellen etter en omstart av systemet når den fikk strøm rett fra mikrokontrolleren. For å fikse dette ble strømforsyningen endret sånn at multiplexeren og sensorene fikk strøm rett fra batteriet. For å få kontroll over strømforbruket det brukt en darlington transistor og en GPIO pin fra mikrokontrolleren for å lage en av og på bryter for sensorene og multiplexeren. Jord fra sensorer og multiplexer er koblet på Kollektor på transistoren, jord fra batteri og mikrokontroller er koblet til emitter på transistoren og for å styre strømmen av og på blir GPIO 1 på mikrokontrolleren koblet inn til base på transistoren. Se figur3.2. Det fører til at strømforbruket blir redusert når man kun aktiverer kretsen om man skal samle data. Merk man må ta hensyn til at noen enheter trenger noe tid for å starte når man slår på strømmen. Dette blir tatt hensyn til i programvare ved hjelp av forsinkelser.

3.3.4 Valg av sensorer

Ut ifra samtale med oppdragsgiver ble det gitt forslagene *temperatur i luft og/eller i jorden* , *fukt i luft og/eller i jorden*, *lysinnstråling*, *ph, nitrogen, innhold i jord*, eller *vind*. Det ble bestemt å fokusere på å lage et bra fundament for senere utvikling for oppdragsgiver og derfor ble disse sensorene valgt: luftfuktighet, temperatur i lufta, jord fuktighet og jord temperatur, lysstyrke og spektrometer.

Sensorene som ble valgt var:

SCD30

SCD30 er en sensor som måler tre verdier. Den måler luft fuktighet, luft temperatur og CO2. Denne sensoren ble valgt ettersom den har tre sensorverdier som er relevant. I tillegg har den god dokumentasjon, I2C og et rammeverk for kalibrering.

RS PRO type J termoelement og MCP9600

For å måle temperatur i bakken ble en RS PRO TYPE J termoelement valgt. Denne ble valgt på grunn av lengden på den inkluderte kabelen og fordi den kan måle temperaturer fra -60 til 350 grader som er godt innenfor systemets bruksområde. Denne krevde en forsterker og derfor ble MCP9600 valgt siden denne har I2C kommunikasjon og et arduino kompatibelt bibliotek. Forsterkeren ble integrert i kabelen med krympestrømpe rundt. Sensoren ble dermed «I2C» kompatibel.

DFRobot SEN0308

SEN0308 er en analog sensor som måler fuktighet i bakke. Denne ble valgt fordi den er vanntett og har lengde så den rekker langt ned i jorden. Den blir koplet til en ADS1115 som er en analog digital konverterer som har I2C.

Adafruit BH1750

BH1750 er en lyssensor som gir ut verdier i lux. Senoren har bra rammeverk med bibliotek og eksempelkode og er derfor lett å implementere i koden. Den ble valgt slik at man kunne se forskjell på lysforhold der hvor nodene ble plassert.

Adafruit AS7341

AS7341 er en sensor som tar inn lys i ulike fargespekter. Ut ifra samtaler med oppdragsgiver ble det nevnt at spektrometer kan være lønnsomt for å vite hvor mye av forskjellig lys i forskjellig bølgelengde som treffer plantene rundt noden. AS7341 kan ta inn lys fra 10 forskjellige bølgelengder og er ett bra valg for bruk i noden. Den har I2C implementert og bibliotek fra produsent som gjør den enkel å implementere i koden.

3.3.5 Valg av plugger

Det har blitt Implementert en analog plugg og tre I2C plugger på noden. Pluggene som er benyttet er hun og han kompatible plugger fra TE Connecftivity med 4 kontakter 3.1. Alle pluggene har en kapslingsgrad på IP67 og er derfor støvtette og vannmotstandige. Analog pluggen er ment for sensoren SEN3008 som er en ren analog sensor, men kan også benyttes av fremtidige rene analoge sensorer. Resten av sensorene er I2C kompatible.

3.3.6 Valg av sensorkabel

Det ble valgt en ethernet kabel som utvendig kabel for sensorene. Denne kabelen ble valgt siden den har god isolering og skal tåle å stå ute uten å bli særlig påvirket av vær. Ethernet kabelen er i tilleg tvunnet som gjør at data integritet holder seg bedre.

3.3.7 Valg av boks

Det ble originalt bestilt tre bokser fra Wiska CLWIB serien i forskjellige størrelser. Ut ifra disse ble *Wiska CLWIB 2 120x160x70mm* valgt ut ifra størrelsen på det som skulle inn i noden. Boksen er IP65 sikker og har ett gjennomsiktig plastikk lokk som er viktig for at solcellepanel og lyssensorene BH1750 og AS7341 skal kunne bli plassert på innsiden av lokket sånn at man slipper å borre unødvendige hull i boksen.

3.3.8 Design av 3D-printede deler

For å sørge for at komponenter og ledninger inne i noden skulle holdes fast og for å kunne feste noden til et jordspyd ble det ved hjelp av Fusion 360 designet ulike deler som ble 3D-printet i PLA-plastikk. PLA-plastikk ble brukt fordi det er det er det mest tilgjengelige og mest brukte materialet for 3D-printing. Det var heller ingen spesielle krav for brannsikkerhet eller styrke på materialet. Delene er printet på en X1 Carbon 3D-printer fra Bambu Labs som SINTEF har. På grunn av valget av 3D-printer er det brukt Bambu Labs sin egen slicer, Bambu Studio for å kutte opp modellene. Alle printene ble gjort med standard instillinger untatt SCD30 toppen og hovedkort holderen som trengte støtter og sensor holderene/skinnene som trengte en brem.

Hovedkort holder

Hovedkortet har mikrokontrolleren og I2C multiplexeren på seg, samt antennen for LoRaWAN kommunikasjon. Hovedkortet bruker et Adafruit Perma-Proto Half-sized breadboard med en størrelse på 81mm x 51mm. Designet inkluderer en plass for å skru inn antennen og et utkutt på venstre side for å ha plass til føttene til mikrokontrolleren. Det er også stor plass bak kortet for å legge ledninger bak. Vedlegg A.1 viser tegning av designet til hovedkort holderen.

ADC-kort holder

ADC-kortet har har ADCen på seg. ADC-kortet bruker et Adafruit Perma-Proto Quarter-sized Breadboard med en størrelse på 43mm x 50.8mm. Designet er basert på hovedkort holderen og har også stor plass bak kortet for å legge ledninger bak. Vedlegg A.2 viser tegning av designet til ADC-kort holderen.

Mutter holder

Mutter holderen er designet for at man kan lime fast en M5 mutter inni uten at den skal spinne når man skrur inn en skrue. Dette er gjort ved å lage et hexagonalt mønster i bunnen som er akkurat stort nok for en M5 mutter. Vedlegg A.3 viser tegning av designet.

Stripsanker

Stripsankeret ble funnet på printables og er designet av LoboCNC [26]

AS7430/BH1750 sensorhus

AS7340 og BH1750 sensorene har samme størrelse og kan derfor bruke samme type sensorhus. Sensorene må skjermes fra lys fra sidene for at sensordata skall være mest mulig rett og ikke bli påvirket av lys fra innsiden av boksen. Sensorene skal festes til lokket av noden og må derfor ha punkter hvor man kan legge på lim. Det må også være mulig å koble til en data/strøm ledning til sensoren etter den er festet på lokket. Vedlegg A.4 viser tegning av designet.

SCD30 sensorhus

SCD30 sensorhuset sitt design er basert på den offisielle design guiden fra sensoren [27]. Sensorhuset er designet i to deler for at huset skal være vanntett. Bunnen av sensorhuset er designet for å ha plass til ethernetkabelen som er valgt som sensorkabel og inkluderer en liten plass for silicon for vanntetting. Det er og designet stolper for å indeksere og feste SCD30 sensoren til bunnen. Vedlegg A.5 viser tegning av designet. Toppen av sensorhuset inkluderer vegger som indekserer med innsiden av bunnen for at man skal kunne lime de to halvsidene sammen. Det er inkludert et hull for at luft skal kunne treffe sensoren for måledata. Vedlegg A.6 viser tegning av designet.

Sensorfeste / Sensorskinne

SCD30 sensoren trenger en måte å kunne bli festet på siden av noden. Det har derfor blitt designet ett sensorfeste med en skinne. Skinnen og festet ble designet med nok toleranse så de passet inni hverandre og med en flat bakside som man kan lime skinnen og festet på sensoren og noden. Vedlegg A.7 viser designet til festet og Vedlegg A.8 viser designet av skinnen.

3.4 Programvare

3.4.1 CubeCell Kode

CubeCellen er programmert ved bruk av Arduino IDE. Hovedmålet under designet av CubeCell koden er å lage koden oversiktlig og så enkel som mulig for oppdragsgiver å modifisere. Det skal også være lett å legge til nye sensorer etter behov. CubeCell koden har to hoveddeler *Sensor data innsamling* og *data sending*.
Det blir brukt flere bibliotek i koden. Bibliotekene blir brukt for å kommunisere og få data fra sensorene. Sensorene som er valgt er Arduino kompatible og produsentene av disse bruker å lage rammeverk i form av bibliotek og eksempelkode. Disse bibliotekene og deler av eksempelkode er brukt i koden for datainnsamling og sending.

CubeCell koden starter med å inkludere bibliotekene 3.5 og definere I2C adressene til sensorene 3.6, samt laget åtte int32_t data arrayer med størrelsen på 12 verdier, en til hver I2C port på I2C multiplekseren.

Komponent	**Bibliotek**
AS7341	Adafruit AS7341 library.
BH1750	BH1750 library by Christopher Laws.
ADS1115	ads1x15-WE library by Rob Tillaart.
MCP9600	Adafruit MCP9600 library.
SCD30	Seeed SCD30 library.
I2C multiplexer	wire.h

Tabell 3.5: Sensorbibliotek

Komponent	**Adresse i hex**
AS7341	0x39
BH1750	0x23
ADS1115	0x48
MCP9600	0x60
SCD30	0x61
I2C multiplekser	0x70

Tabell 3.6: Standard sensor adresser

For hver sensor blir det laget variabler for lagring av innsamlet sensor data. Om det er en sensor med flere verdier enn en, blir det laget array med nødvendig størrelse.

3.4.2 Sensor datainnsamling

For å implementere nye sensorer senere må man modifisere deler av denne koden. Hoveddelen av denne koden er å lese igjennom portene på I2C multiplekseren og se hvilke sensorer som er koplet til, deretter velge hvilken funksjon å bruke til den sensoren for å samle inn sensor data, og til slutt lagre dataen inn i ett array.
For å implementere nye sensorer må man legge til en ny funksjon for datainnsamling til den nye sensoren og en variabel som den skal fylle. Deretter legge den til som en case i fill_data_array(x,data_array_x).

Sensorer I bruk

Sensorinnsamlingen bruker I2C adressene til sensorene 3.6 for å skille hvilke sensorer som er koplet til og hvilken funksjon å bruke. Denne adressen blir og brukt til å indentifisere sensoren i senere deler av systemet.

Komponent	**Funksjon**
AS7341	multispecter_as7341()
BH1750	lightsensor_BH1750()
ADS1115	SEN0308()
MCP9600	mcp9600()
SCD30	Bruker funksjonen til biblioteket.
I2C multiplexer	sensor_address(int t) og tcaselect(int i)

Tabell 3.7: Data innhenting funksjoner

multispecter_as7341()

Funksjonen fyller data arrayet multispecter_sensor_value[10] med verdier fra as7341.

lightsensor_BH1750()

Funksjonen fyller data variabelen sensor_value_bh1750 med verdier fra BH1750.

SEN0308()

Funksjonen fyller data variabelen sensor_value_SEN0308 med verdier fra SEN0308 med å bruke Ads1115

mcp9600()

Funksjonen fyller data variabelen sensor_value_ mcp9600 med verdier fra termoelement ved å bruke MCP9600

SCD30

Funksjonen fyller data arrayet scd30_sensor_value[3] med verdier fra SCD30.

I2C multiplekser

Funksjonen tcaselect(int i) velger hvilken port på I2C multiplekser som er på for kommunikasjon, der hvor i er hvilken port. Funksjonen sensor_address(int t) bruker tcaselect(int i) for å velge port i, deretter retunerer den adressen til I2C enheten som er koplet til den porten.

get_sensor_data

Funksjonen *get_sensor_data()* starter med å bruke funksjonen *turn_on_off_sensors* for å slå på strøm til sensorkrets. Deretter setter den alle data arrayene til «0». Den går så igjennom en for loop med switch caser som har en case per port på I2C multiplekseren, og ett data array per case. Alle casene bruker funksjonen *fill_data_array(x,data_array_)*. Den fyller port nummer på x, og data_array_x, der hvor x i begge tilfeller er samme nummer som port nummer.

fill_data_array(x,data_array_x)

fill_data_array(x,data_array_x) blir brukt for å fylle data arrayet, den tar inn et port nummer, og ett array. Deretter bruker den det port nummeret i sensor_address(int t) som da returnerer adressen til sensoren som er tilkoplet denne porten. Deretter blir denne adressen brukt i en switch case der hvor adressene til sensorene er casene. Når den kommer inn en case blir den funksjonen som tilhører den sensor adressen kallet og fyller arrayet som ble tatt inn i funksjonen. Den blir fylt element 0 med portnummer, element 1 med sensor adresse og de andre elementene med sensor data 3.3a.

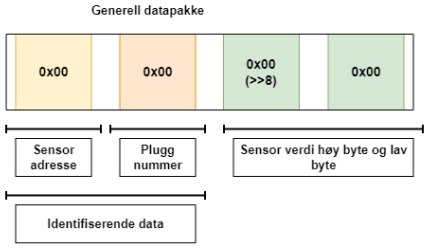
Sending av data

For å sende dataen fra alle data arrayene blir det brukt en funksjon som heter prepareTxFrame. Hovedoppgaven til denne funksjonen er å dele opp dataen som kommer fra sensorene inn i pakker på størrelse med en byte som CubeCellen kan sende ut. Funksjonen tar inn to variabler, port og sending_data som begge er variabler av typen uint8_t. Port variabelen brukes ikke i denne koden, men kan brukes for å sende data på forskjellige porter inn til The Things Network. Sending_data er hvilken data_array som det skal sendes data fra. I starten av funksjonen brukes det en switch for å fylle en lokal array med navn data med verdiene fra data_array. Det brukes en switch for å fylle ett nytt array fordi da slipper man å bruke nestede switcher hvor man kopierer samme kode flere ganger. Dette gjør koden mer lesbar og mindre. Når arrayen har blitt fylt med sensor data går koden videre til en ny switch som tar inn andre elementet i data arrayen. Dette elementet inneholder adressen til sensoren som dataen kommer ifra figur 3.6 . Deretter er det en case

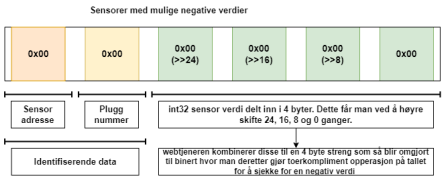
per sensor adresse som man kan koble på noden. Dette gjør det lett å legge til nye sensorer senere. Det blir brukt unike caser per sensor fordi dataen som kommer fra hver sensor er unik og trenger ulik mengde pakker. Det er også en egen case for sending av batteridata.

Oppdeling av sensordata

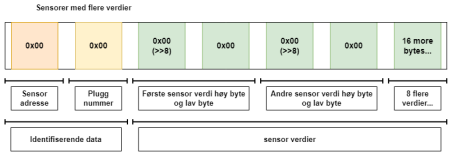
Siden sensordataen er på formatet int32 må den deles opp for at CubeCellen skal kunne sende den. Inne i hver case for sensorene blir dataen delt opp i henhold til hva slags data den sensoren henter inn. For sensorer med negative tall må det sendes hele inten for å få med fortegns biten 3.3b. Dette gjøres ved å høyreskifte inten 24,16,8 og 0 ganger og så bare sende de første åtte bitene hver gang man skifter. For tall som man vet at ikke blir negative trenger gjør man det samme men man høyre skifter bare 8 og 0 ganger 3.3a. Man kan også gjøre dette flere ganger for å sende flere verdier 3.3c. Når pakkene blir sendt til The Things Network blir dataen formatert av et script og så sendt videre til webhook tjeneren.



(a) Generell datapakke



(b) Pakke med negative verdier



(c) Pakke med flere verdier

Figur 3.3: Datapakker

3.4.3 Payload formateringsskript

The Things Network har en nyttig funksjon hvor man kan formatere dataen som kommer inn fra nodene også sende dataen videre som en JSON fil ved hjelp av en webhook. Formatterings skriptet brukt i prosjektet returnerer en JSON fil med følgende data

- rawInput: all data som kom inn (mest brukt for feilsøking).
- battery: batteri spenning i mV (denne blir bare fylt hvis det kommer inn data fra plugg 9).
- connector: hvilken plugg som dataen kom fra (mest brukt for feilsøking).
- sensor: hvilken sensoradresse dataen kom fra.
- dataArray: en array med sensordataen som kommer inn (blir kombinert senere i webhook tjeneren).

3.4.4 Webhooken

Webhooken i The Things Network er satt opp sånn at den sendes ut hver gang det kommer inn data. Webhooken er på JSON format og inneholder

- IDen til noden som sendte inn data.
- Formatert data fra payload formateringsskritet.
- Porten dataen kom inn på (ikke brukt i vår applikasjon, men kan brukes for fremtidig funksjonalitet).
- Tidsstempel på når dataen kom inn.

3.4.5 Webhook tjener

Webhook tjeneren er skrevet i VScode og har som jobb er å få inn data fra webhooken som blir sendt fra The Things Network og formatere den på en sånn måte at den kan bli lagt inn i SINTEF sin database. Tjeneren er kodet i JavaScript og bruker NodeJS. Koden er basert på Marco Suter sin guide for å koble postgreSQL til NodeJS [28] og Bearer sin JSON webhook guide for NodeJS [29].

3.4.6 Database

Databasen er en postgreSQL database og ble satt opp av SINTEF IT internt. Databasen inneholder en tabell med 19 kolonner.

Kolonner

Det er to typer kolonner, kolonner med identifiserende informasjon og kolonner for sensorverdier. For å få ut riktige data fra databasen må man ta ut verdier fra en av kolonnene i kolonner med sensorinformasjon og filtrere ved hjelp av sensoradresse og sensorbox id. For å ikke ha unødvendige kolonner i databasen ble det valgt å ha en kolonne med verdier for sensorer som bare gir en verdi. Det var fortsatt behov for unike kolonner på AS7340 sensoren og SCD30 sensoren på grunn av at det kommer flere verdier fra samme sensoren.

3.5 Klient del

3.5.1 Bruksanvisning

Som en del av prosjektet er det behov for dokumentasjon på hvordan man kunne bygge og sette opp nye noder. Dokumentasjonen er laget i Gitbook fordi det er lett å lage en dokumentasjon som ser oversiktlig og bra ut ved hjelp av en nettside. I tillegg kan dokumentasjonen lett gjøres tilgjengelig for andre via en link. Oppdragsgiver ønsket dokumentasjonen på engelsk slik at den var tilgjengelig for flere. Dokumentasjonen delt opp i tre deler kode, hardware og backend som selv har sine underkategorier.

3.5.2 Grafana

For å visualisere dataen fra nodene har Grafana blitt valgt. En av grunnene til at Grafana er en bra løsning er at det kan kjøres selv på egen tjener eller man kan velge å betale for Grafana Cloud. Grafana er også en god løsning på grunn av at det støtter tilkobling til mange forskjellige typer databaser inkludert postgreSQL og har bra med tilpasnings muligheter for dataen.

3.6 Testing

Det er utført to tester på systemet. En test for å se om hele systemet fungerer som er beskrevet i vedlegg E.1 og en for å finne ut det reelle strømforbruket til nodene som er beskrevet i vedlegg E.2.

3.7 Frafall fra gruppen

Det var egentlig planlagt å lage to protoyper så man kunne teste den første på gården før man fikk en bedre ferdig prototype på slutten av oppgaven, men på grunn av frafall av en person fra gruppen måtte planer endres. Det førte til at den første prototypen ble bare brukt til å teste ulike deler av systemet før versjon to av noden kunne bli laget. Det ble også laget et nytt gantt-skjema vedlegg G.1. Gantt-skjemaet inneholdt nye oppdaterte arbeidspakker og tidsfrister og ble laget i programmet TeamGantt. Grunnen til bruk av TeamGantt er at programmet er oversiktlig og man kan blant annet lett kan se hvilke oppgaver man har igjen og hvor lenge det er til de skal være levert inn.

Kapittel 4

Resultat

Kravene til prosjekter etter diskusjon med oppdragsgiver var at en node skulle ha fire kontakter for fire forskjellige sensorer. Noden skal tåle å stå ute i regn (minimum IP63). Den skal ha solcellepanel og batteriløsning. Noden skal kommunisere over LoRaWAN og den skal være modulær så om sensorer eller noden selv blir ødelagt skal det være lett å bytte ut deler. Det skal være mulig å implementere nye sensorer. Forslag til målinger kunne være: *temperatur i luft og/eller i jorden* , *fukt i luft og/eller i jorden*, *lysinnstråling*, *ph, nitrogen, innhold i jord*, eller *vind*. Prosjektet har gitt følgende resultater.

4.1 Maskinvare

4.1.1 Node

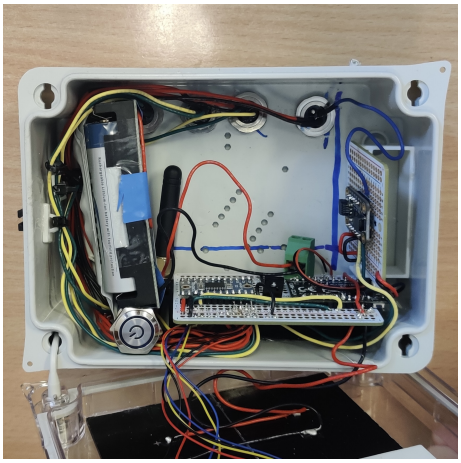
Det ble produsert to noder. Hver node er utstyr med tre I2C kontakter og en analog kontakt. På lokket av noden er to solcellepanel og senorene BH1750 og AS7341. På innsiden av boksen er det en batteripakke med to litium batterier i parallell og knapp for av og på, en analog til digital konverterer for ene pluggen og et hovedkort med ett CubeCell Dev-Board (V2), krets for sensorstrøm av og på og en I2C multiplekseren.

(a) Node side med plugger

(b) Node framside

Figur 4.1: Bilder av node.1

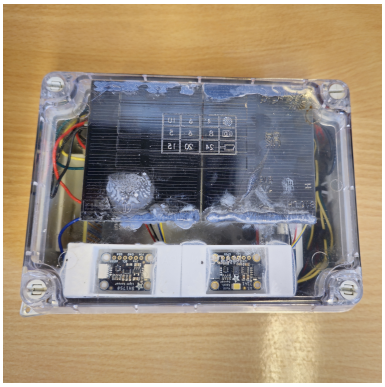
(a) Node side med feste



(b) Node innside

Figur 4.2: Bilder av node.2

(a) Node Underside



(b) Node Topp

Figur 4.3: Bilder av node.3

4.1.2 Sensorer

Det ble produsert to sett med sensorer. Ett sett til hver av nodene. Alle sensorene bruker I2C for kommunikasjon. Sensorene er enten allerede I2C kompatibel, koplet til en digital analog konverterer som har I2C eller koplet til en forsterker med I2C. Sensorene SCD30, AS7341 og BH1750 er allerede I2C kompatibel. BH1750 og AS7341 ble limt på lokket til boksen til noden slik man ser på 4.3. SCD30 er koplet direkte til en kontakt og plassert i ett etui se figur: 4.5. Termoelement er en sensor som krever en forsterker med I2C, Den ble koplet til og det ble brukt krympestrømpe for å isolere forsterkeren MCP9600, se figur: 4.4. SEN0308 er en analog sensor se figur: 4.4 og krever en analog digital konverterer. Denne sensoren blir beholdt som analog og koplet til analog kontakten på noden.

(a) Termoelement med MCP9600 i krympestrømpe, kabel og kontakt

(b) SEN0308

Figur 4.4: SEN0308 og Termoelement med MCP9600 .1

Figur 4.5: SCD30 med kabel, kontakt og hus

4.1.3 Stykklister

Det har blitt laget to stykklister for systemet. Den første inneholder liste med deler for ett helt system. Vedlegg C.1 Den andre stykklister inneholder deler man trenger for å lage en ekstra ny node med tilhørende sensorer. Vedlegg C.2

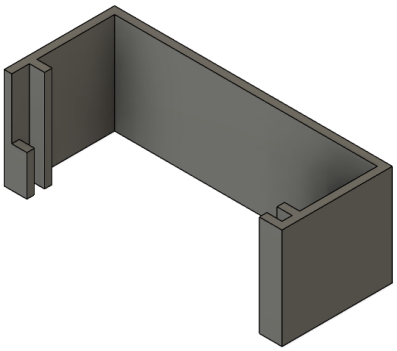
4.1.4 3D-printede deler

Figur 4.6 viser ferdig modell av hovedkort holderen og hovedkort holderen med kort inni.

(a) Holder alene **(b)** Holder med kort inni

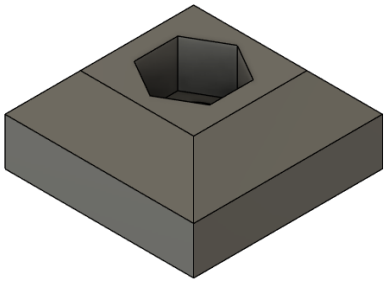
Figur 4.6: Hovedkort holder

Figur 4.7 viser ferdig design av ADC-kort holderen.



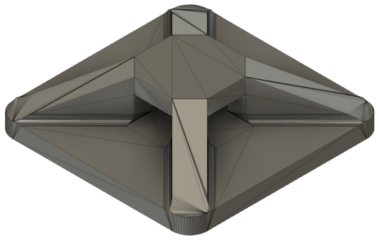
Figur 4.7: ADC-kort holder

Figur 4.8 ser man ferdig design av mutter holderen.



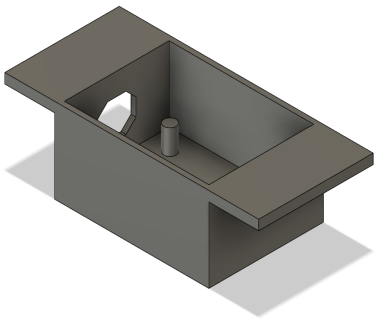
Figur 4.8: Mutter holder

Figur 4.9 ser man designet til stripsankeret designet av LoboCNC. [26].



Figur 4.9: Stripsanker

Figur 4.10 ser man designet til sensorhuset brukt for både AS7340 sensoren og BH1750 sensoren.



Figur 4.10: AS7340/BH1750 Sensorhus

Figur 4.11 viser top 4.11b og bunn 4.11a av sensorhuset til SCD30 sensoren.

(a) Sensorhus bunn **(b)** Sensorhus top

Figur 4.11: SCD30 Sensorhus

Figur 4.12 viser designet til sensor skinnen 4.12a/festet 4.12b for å feste scd30 sensoren.

(a) Skinne **(b)** Feste

Figur 4.12: Sensorfeste/skinne

4.2 Programvare

4.2.1 CubeCell kode

Prosjektet har resultert i en modulær kode for noden, som kan ta inn sensor verdier og sender dem videre til The Things Network. CubeCell koden finnes i vedlegg B.1.

4.2.2 Webhook tjeneren

Det har blitt laget en webhook tjener for å overføre data fra The Things Network til databasen. Kode til webtjerner finnes i vedlegg B.2.1 og B.2.2.

4.2.3 The Things Network formaterings skript

Inne i The Things Network har det blitt laget et formaterings skript for å formatere dataen brukt av webhook tjeneren. Formaterings skriptet finnes i vedlegg B.3.

4.2.4 Kalibrerings kode SCD30 sensor

For å kalibrere SCD30 sensoren brukes en modifisert versjon av kalibrerings koden laget av SEEED studios. Kalibrerings koden finnes i vedlegg B.4.

4.2.5 Kalibrerings kode SEN0193 sensor

Kalibrerings koden finnes i vedlegg B.5.

4.2.6 GitHub

All programvare og STL-filer for 3D-printing ble samlet på GitHub for lett tilgjengelighet. GitHub siden er delt inn i fem mapper en for CubeCell koden, en for kalibreringskode, en for STL-filene til 3D-printing, en for Things Stack formaterings skriptet og en for webtjeneren.

Link: https://github.com/IvarHansgard/BachelorOppgave.git

4.3 Klient del

4.3.1 database

Tabellen i databasen ble delt inn i følgende kolonner.

Kolonner med identifiserende informasjon:

- sensor_type (Sensoradressen i hexadecimal)
- sensorbox_connector (Pluggen sensoren er koblet til mest brukt til feilsøking)
- created_at (Tidsstempel for dataen)
- battery (batteri spenning)
- sensorbox_id (IDen til node fra The Things Network)

Kolonner med sensor verdier:

AS7340 sensoren har kolonner for hver av fargeområdene den kan måle og SCD30 sensoren har kolonner for hver av verdiene den kan måle temperatur, luftfuktighet og CO2 i lufta. Tabell 4.1 viser alle verdikolonnene i databasen.

Kolonner for AS7340 sensor	Kolonner for SCD30 sensor	Kolonne for sensorer med en verdi
nm435_455	temperature	value
nm405_425	humidity	
nm470_490	co2	
nm505_525		
nm545_565		
nm580_600		
nm620_640		
nm670_690		
Clear		
Nir		

Tabell 4.1: Kolonner for sensorverdier

4.3.2 Bruksanvisning/dokumentasjon

Bruksanvisningen som ble laget ble delt inn i tre ulike kategorier kode, hardware og backend.

Kode:

Kode delen av dokumentasjonen inneholder en detaljert gjennomgang av koden til Cubecellen og webhook tjeneren. Den inneholder link til koden på GitHub og alt av biblioteker man trenger for å sette opp CubeCellen samt hvordan å modifisere koden for flere sensorer. Det er også en bruksanvisning på hvordan man setter opp webhook tjeneren.

Kode delen er delt inn i følgende underelementer:

- Cubecell code (Inneholder alt av kode og bilioteker til cubecellen og hvordan man modifiserer den for å legge til nye sensorer.)
- Webhook server (Inneholder alt av kode til webhook tjeneren, hvordan man setter den opp og hvordan man modifiserer den for å legge til nye sensorer.)

Hardware:

Hardware delen av dokumentasjonen inneholder hvordan man setter sammen den fysiske delen av prosjektet dette er sensorene og noden. Denne delen inneholder også en detaljert gjennomgang av hvilke innstillinger og hvilken orientasjon man skal 3D-printe de forskjellige delene til nodene og sensorene.

Hardware delen er delt inn i følgende underelementer:

- Sensors (Inneholder BOM og hvordan man bygger alle sensorene)
- Nodes (Inneholder BOM og hvordan man bygger noder)
- 3D-printing (Inneholder link til STL-filer og guide på hvordan man printer de forskjellige delene til noden)

Backend:

Backend delen av dokumentasjonen inneholder alle delene som prosesserer dataen i bakgrunnen før den kommer fram til databasen.

Backend delen er delt inn i følgende underelementer:

- Database (Inneholder oppsett av databasen som hvilke tabeller og kolonner)
- Thingsstack (Inneholder bruksanvisning på hvordan man legger inn noder, setter opp payload format skriptet og webhook i The Things Network.
- Grafana (Inneholder bruksanvisning på hvordan man kobler opp grafana til databasen og hvordan man setter opp visualisering av data.

Link:

Bruksanvisningen er lagt ut på GitBook og kan finnes på denne siden: https://soldrevne-sensorbokser.gitbook.io/agrivoltaics-sensorboxes-documentation/

4.3.3 Grafana

Grafana er koblet til postgreSQL databasen og det har blitt satt opp ett testdashbord for å teste systemet og visualisere for oppdragsgiver hvordan det kan se ut når de setter de skal opp selv. Figur 4.13 viser dashboardet

Figur 4.13: Grafana dashbord

Kapittel 5

Diskusjon

5.1 Ferdig produkt

Ut ifra resultatene mener prosjektgruppen at de har levert et produkt som stemmer med de kravene som ble gitt i starten av prosjektet.

Det har blitt levert to noder med tilhørende sensorer som er vannmotstandige. Nodene har batteriløsning med solcellepanel for lading. Nodene er modulære ved at sensorer lett kan byttes ut og alle pluggene på noden kan brukes om hverandre. Nodene kommuniserer over LoRaWAN og dataen blir sendt til en database. Dataen kan bli hentet ut ved hjelp av en visuell platform som Grafana. Det er blitt laget en oversiktlig dokumentasjon med veiledning for at oppdragsgiver kan reprodusere systemet. Dokumentasjonen inneholder hvordan å lage nye noder og sensorer, hvordan å legge til nye sensorer, hvordan å sette opp LoRaWAN nettverket og kommunikasjonen med databasen.

5.2 Feilkilder

Feil kan bli synlig etter noden er plassert ute i felt ettersom produktet kun er testet i gunstige forhold.

5.2.1 vanntetting

Det har ikke blitt utført noen tester ute i felt med sensorene eller noden. Vi kan derfor ikke bekrefte at sensorene er vanntette. Nodene har blitt bygd inne i en IP65 sikker boks så skal være støvtett og spyling sikker, men vi har ikke verifisert dette. Vi har også drillet hull i boksen for pluggene som kan påvirke IP graden, men pluggene har pakninger og er IP67 sikre så i teorien skal boksen fortsatt være vanntett. Det er ikke implementert noen lufting. På grunn av dette kan det oppstå kondens inne i boksen, men dette har ikke blitt testet så prosjektgruppen vet ikke om dette blir et problem eller ikke.

5.2.2 Kalibrering av sensorer

SCD30

SCD30 har allerede et rammeverk fra produsenten for kalibrering. Det rammeverket ble brukt for å lage en kode slik at man kunne bruke noden for å kalibrere denne sensoren.

SEN0308

SEN0308 tar inn analoge verdier som ikke sier mye om fuktigheten i bakken. Derfor blir det laget en Kalibrerings kode som man må bruke for å finne verdier i luft, og verdier i vann. Når man kjører koden må man lese av verdiene. Deretter må man fylle disse verdiene inn i hovedkoden. Da returnerer funksjonen i hovedkoden en prosentverdi på hvor fuktig bakken er.

5.3 Forbedringer

Gruppen har funnet noen forbedringer som kan bli gjort på systemet.

5.3.1 Node

Skruterminaler

Akkurat nå kan det være litt vanskelig å lodde alle ledningene fra sensorpluggene til hovedkortet. For å gjøre det lettere å ta fra hverandre og sette sammen nodene kan man bruke flere skruterminaler til alle ledningene som skal inn og ut av hovedkortet. Dette vil gjøre monteringen av boksen helt loddefritt.

Batteri

Batteriene som ble valgt til systemet var litt for store for batteriholderen. Batteriholderen måtte derfor bli kuttet opp under bygging av batteriet. Derfor er en mulig forbedring å sørge for at batteriene som blir valgt passer i holderen sånn at byggingen av batteriet blir lettere. Bryteren på batteriet er også litt stor og kan erstattes med en liten glidebryter.

Produksjonsklar versjon

Sensorboksen som har blitt laget nå kan sees på som en prototype. En mulig produksjonsklar versjon av noden kan være ett spesiallaget kretskort. Dette kretskortet kan inneholde CubeCell chippen og ADCen med fastmonterte versjoner av pluggene på bunnen. Kortet må også ha plugger å koble til batteri og solcellepanel. Man kan også designe en egen boks for kretskortet som vil være mye mindre enn sensorboksen designet i dette prosjektet på grunn av det integrerte designet.

5.3.2 Sensor

En forbedring på sensorene kan være å bytte ut SCD30 med en Ip-sikret version siden denne er i en 3d printent etui nå. Det kan være denne etui-en tåler å stå ute, men ettersom det ikke er testet er ikke det garantert. Andre forbedringer på sensorer vil variere ut ifra hvilke data oppdragsgiver ønsker å samle.

5.3.3 Chirpstack

Istedenfor å bruke the things network hvor man har lite kontroll over dataen sin. Kan man sette opp en ChirpStack server som har samme funksjonalitet som på the things network, men man kan kjøre det selv.

5.3.4 Webhook tjener

En fremtidig forbedring som kan bli gjort på webhook tjeneren er å implementere float sensorverdier. Dette kan gjøres ved å for eksempel gange tallet man vil sende med 100 (for 2 desimaler) i CubeCell sendekoden og dele tallet med 100 når det kommer fram til webhook tjeneren.

Kapittel 6

Konklusjon

I dette prosjektet har det blitt designet og laget et system for trådløs overføring av sensordata over et LoRaWan nettverk. Systemets fysiske deler består av en node med sensorer koblet til og en gateway som sender data til The Things Network. Det er også laget en tjener som tar inn data fra The Things Network i form av webhooker og formaterer dem sånn at dataen kan bli lagt inn i SINTEF sin database. For å visualisere dataen har Grafana blitt valgt. Det har også blitt satt opp et testdashbord for demonstrere hvordan SINTEF kan sette opp grafana selv.

Hele systemet er dokumentert og det har blitt laget en guide som viser hvordan det kan replikeres. Den forklarer hvordan i detalj å bygge selve nodene og sensorene, hvordan man implementere nye sensorer i koden til systemet og hvordan man setter opp backend med webhook tjener, The Things Network og database. Guiden går også igjennom hvordan man setter opp Grafana og kobler seg opp mot databasen.

Den originale problemstillingen fra oppdragsgiver var at man skulle lage et dokumentert rammeverk/fremgangsmåte bestående av modulære soldrevne bokser for innsamling av sensordata tilknyttet et trådløst LoraWAN nettverk. Prosjektgruppa skulle sette opp LoraWAN nettverket og ordne tilknytning til SINTEF-database i sky og lage IP67/68 soldrevne data innsamlingsbokser til bruk i felt. Sensordataene skulle lagres og gjøres tilgjengelig for både SINTEF-forskere og eksterne kunder, gjerne i form av en nettside med presentasjon av dataene f.eks. Ved hjelp av grafana eller tilsvarende. Prosjektgruppa skulle fokusere på Skjetleincaset, men det var ønskelig at det skal være enkelt å bygge nye bokser og knytte til nye plasseringer i fremtidige prosjekter.

Prosjektgruppa mener at de spesifiserte kravene til systemet er oppfylt selv om det ikke ble tid til å teste systemet ute i felt før prosjektinnlevering. Gruppen er usikre på om kravet til vantetting har blitt møtt eller om nye feil vil oppstå ved lang brukstid. Allikevel mener prosjektgruppa at oppdragsgiver har fått et solid grunnlag til å bygge videre på. I tillegg har prosjektgruppa gitt forslag til ytterligere forbedringer på systemet.

Bibliografi

[1] (2023), hentet den 21.04.2023. adresse: https://www.regjeringen.no/no/dokumenter/nou-2023-3/id2961311/?ch=1.

[2] Wikipedia, *Maskinvare*, (2023), Hentet den: 08.05.2023. adresse: https://no.wikipedia.org/wiki/Maskinvare.

[3] Bjørn B. Larsen, *mikrokontroller*, (2023, hentet den 20.05.2023. adresse: https://snl.no/mikrokontroller.

[4] The network of things, *What are LoRa and LoRaWAN?* (u.å.), Hentet den: 08.05.2023. adresse: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/.

[5] inpixon, *What is Chirp (CSS) Technology?* (u.å.), Hentet den: 08.05.2023. adresse: https://www.inpixon.com/technology/standards/chirp-spread-spectrum.

[6] HELTEC AUTOMATION, *CubeCell – Dev-Board (V2)*, (u.å.), Hentet den: 08.05.2023. adresse: https://heltec.org/project/htcc-ab01-v2/.

[7] andprof, *What is arduino software (IDE), and how use it ?* (u.å.), Hentet den: 08.05.2023. adresse: https://andprof.com/tools/what-is-arduino-software-ide-and-how-use-it/.

[8] Gramstad,T, *åpen kildekode*, (2021), Hentet den: 08.05.2023. adresse: https://snl.no/C3%5C%A5pen_kildekode.

[9] wikipedia, *I^2C*, (2023), Hentet den: 08.05.2023. adresse: https://en.wikipedia.org/wiki/I%5C%C2%5C%B2C.

[10] wikipedia, *Byte*, (2023), Hentet den: 08.05.2023. adresse: https://en.wikipedia.org/wiki/Byte.

[11] wikipedia, *3d-utskrift*, (2022), Hentet den: 08.05.2023. adresse: https://no.wikipedia.org/wiki/3D-utskrift.

[12] Wikipedia, *Slicer*, (2023), Hentet den: 20.05.2023. adresse: https://no.wikipedia.org/wiki/Slicer.

[13] (2023), hentet den 21.04.2023. adresse: https://no.wikipedia.org/wiki/Gateway_(telekommunikasjon).

[14] Wikipedia, *Server*, (2023), Hentet den: 08.05.2023. adresse: https://no.wikipedia.org/wiki/Server.

[15] THE THINGS NETWORK, *thethingsnetwork*, (u.å.), Hentet den: 08.05.2023. adresse: https://www.thethingsnetwork.org/docs/quick-start/.

[16] wikipedia, *JavaScript*, (2023), Hentet den: 08.05.2023. adresse: https://no.wikipedia.org/wiki/JavaScript.

[17] wikipedia, *JSON*, (2020), Hentet den: 08.05.2023. adresse: https://no.wikipedia.org/wiki/JSON.

[18] Wikipedia, *Webhook*, (2023), Hentet den: 08.05.2023. adresse: https://en.wikipedia.org/wiki/Webhook.

[19] wikipedia, *Nodejs*, (2022), Hentet den: 08.05.2023. adresse: https://no.wikipedia.org/wiki/Node.js.

[20] Database, *Database*, (2023), Hentet den: 08.05.2023. adresse: https://no.wikipedia.org/wiki/Database.

[21] PostgreSQL, *PostgreSQL*, (2023), Hentet den: 08.05.2023. adresse: https://no.wikipedia.org/wiki/PostgreSQL.

[22] GitBook, *What is GitBook?* (u.å.), Hentet den: 08.05.2023. adresse: https://docs.gitbook.com/.

[23] wikipedia, *Grafana*, (2023), Hentet den: 08.05.2023. adresse: https://no.wikipedia.org/wiki/Grafana.

[24] (2023), hentet den 21.04.2023. adresse: https://grafana.com/products/cloud/?plcmt=footer.

[25] (2023), hentet den 20.04.2023. adresse: https://www.jcalc.net/battery-size-calculator.

[26] LoboCNC, *Stripsanker*, (2023), Hentet den: 14.05.2023. adresse: https://www.printables.com/model/132296-zip-tie-anchor/files.

[27] (2023), hentet den 21.04.2023. adresse: https://files.seeedstudio.com/wiki/Grove-CO2-Temperature-Humidity-Sensor-SCD30/res/CD_AN_SCD30_Design-In_Guidelines_D2.pdf.

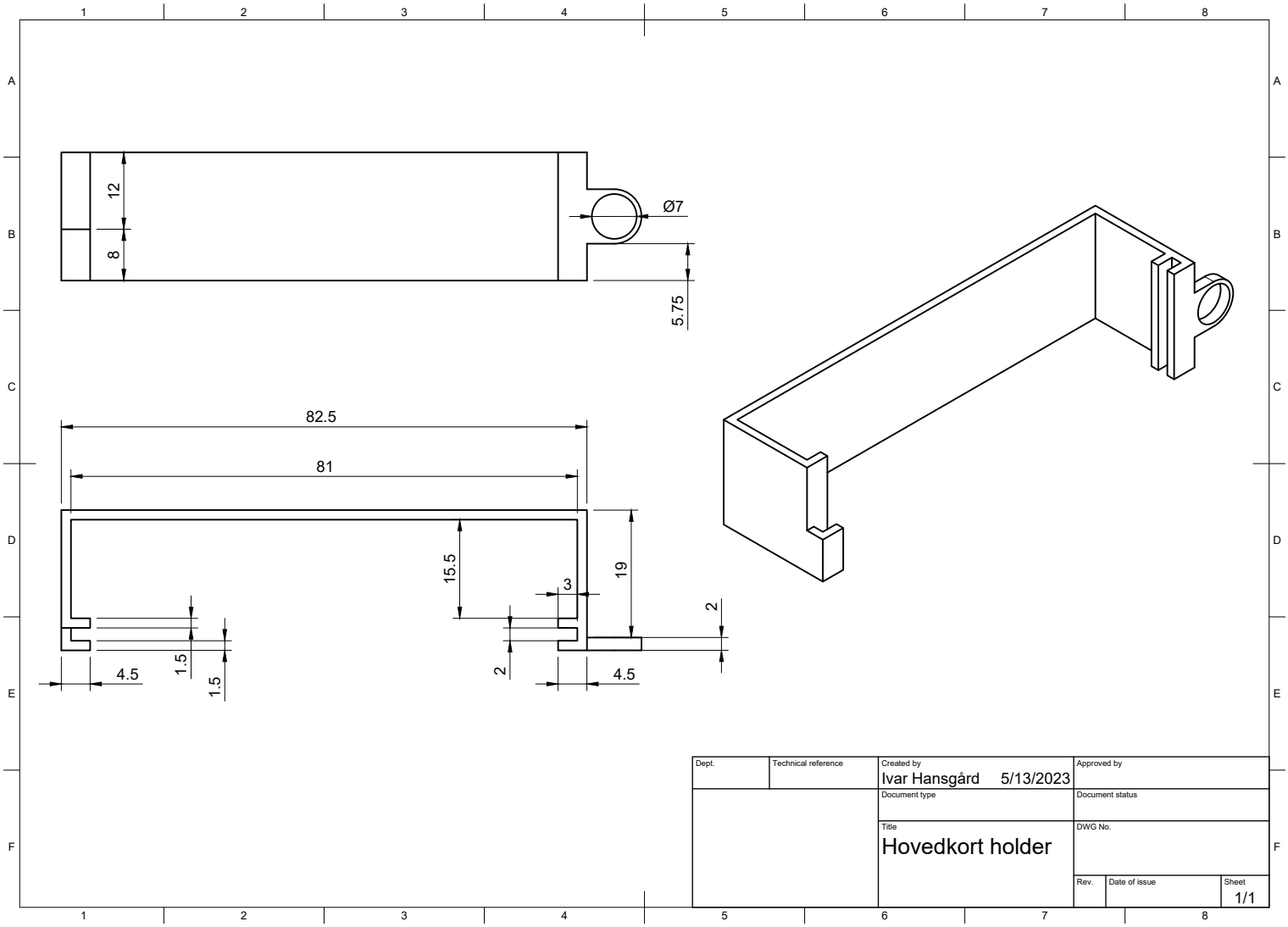
[28] Marco Suter, *Connecting to a PostgreSQL database using Node.js*, (2023), Hentet den: 14.05.2023. adresse: https://northflank.com/guides/connecting-to-a-postgresql-database-using-node-js.

[29] Bearer, *Consuming Webhooks with Node.js and Express*, (2023), Hentet den: 14.05.2023. adresse: https://medium.com/@BearerSH/consuming-webhooks-with-node-js-and-express-50e007fc7ae2.

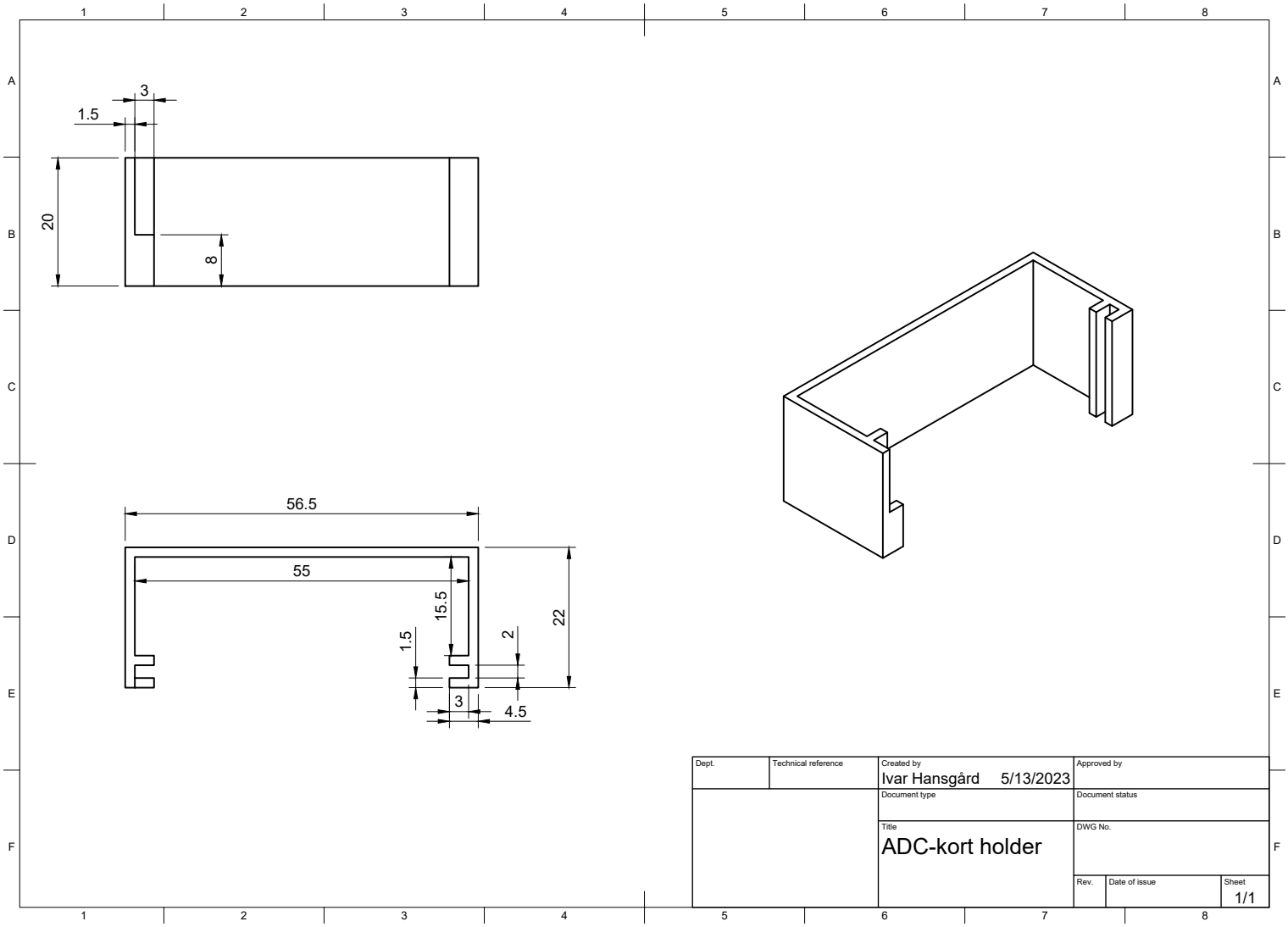
Vedlegg A

Tegninger

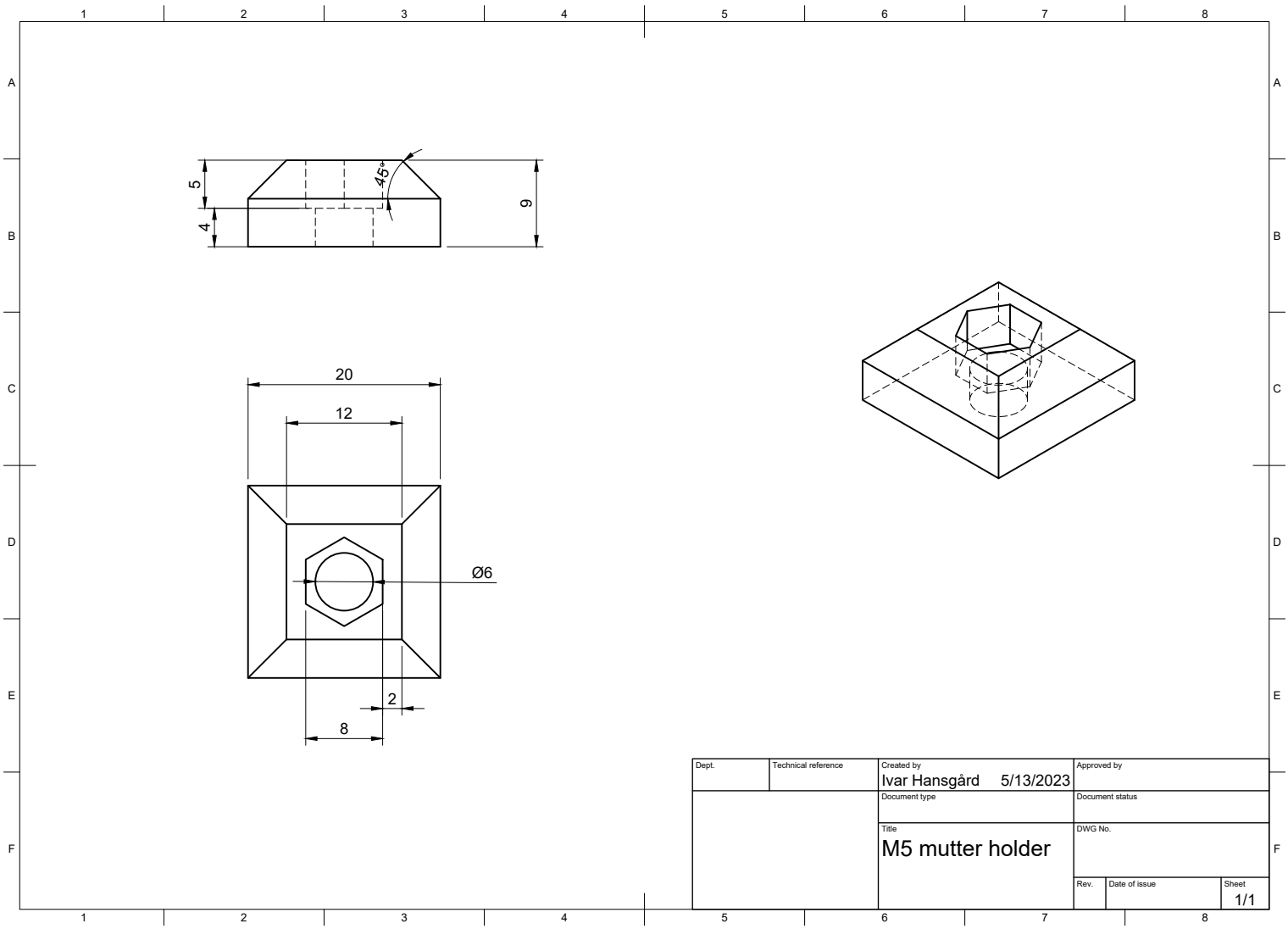
[A.1] Hovedkort holder



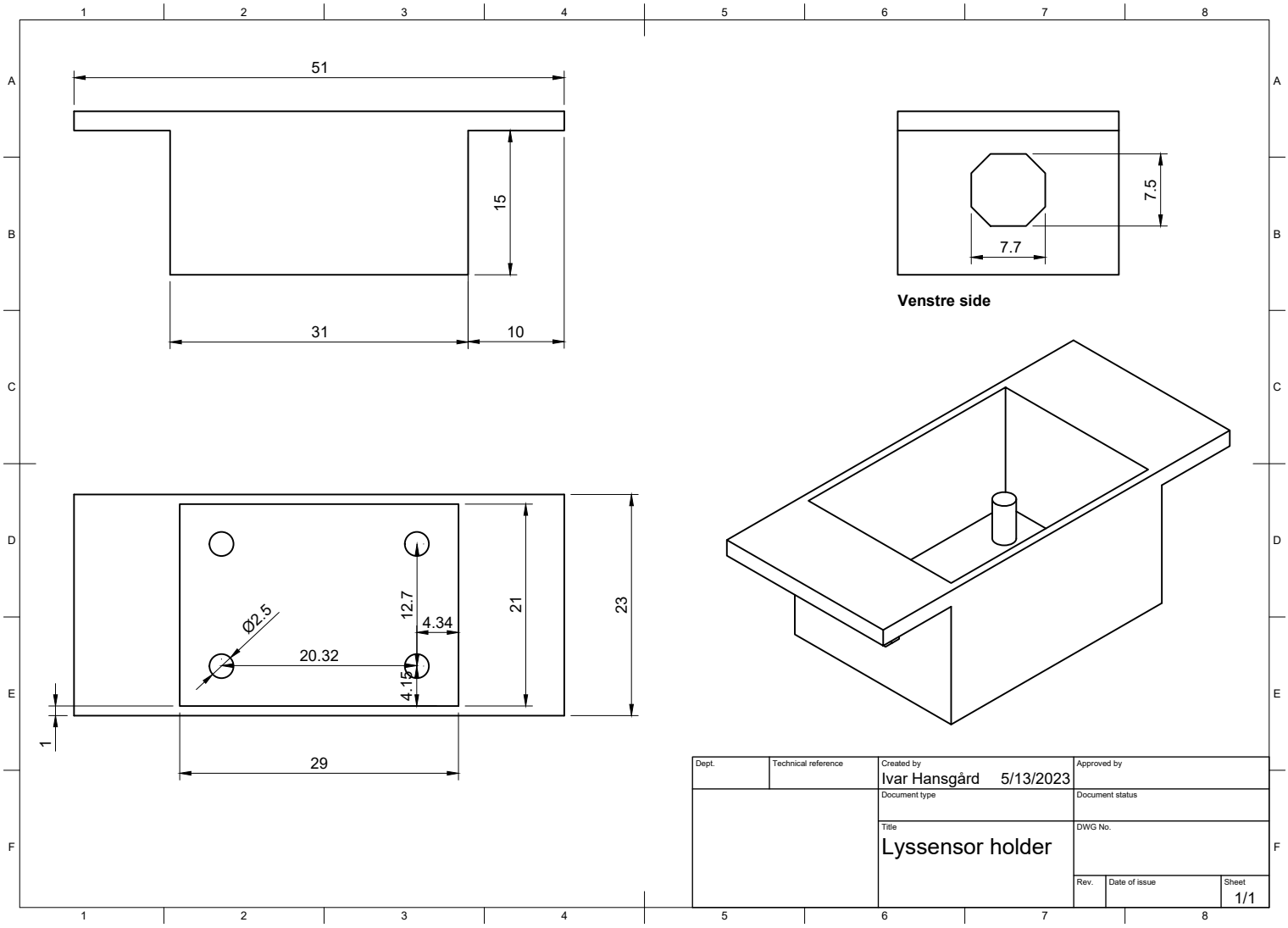
[A.2] ACD-kort holder



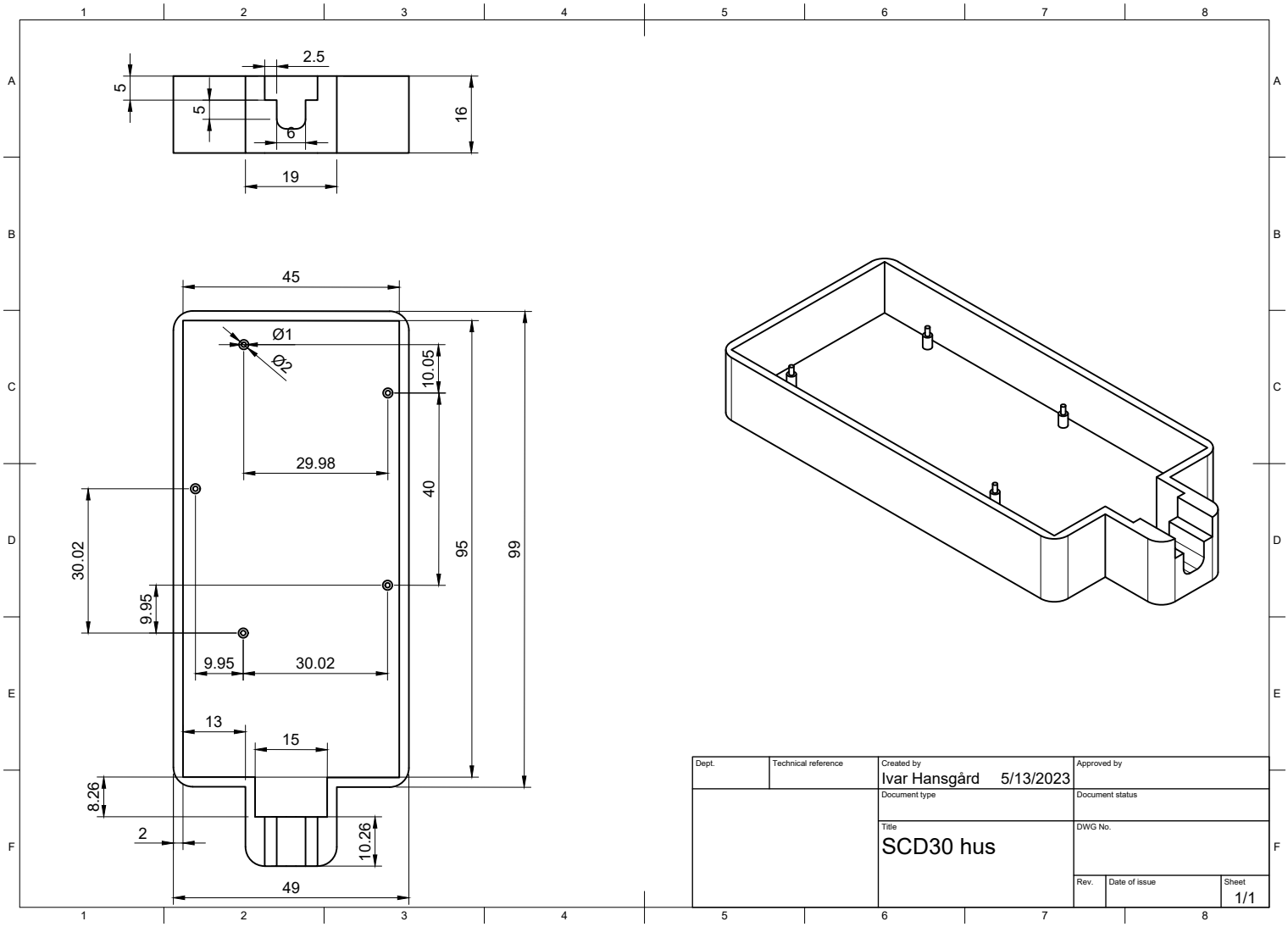
[A.3] M5 mutter holder



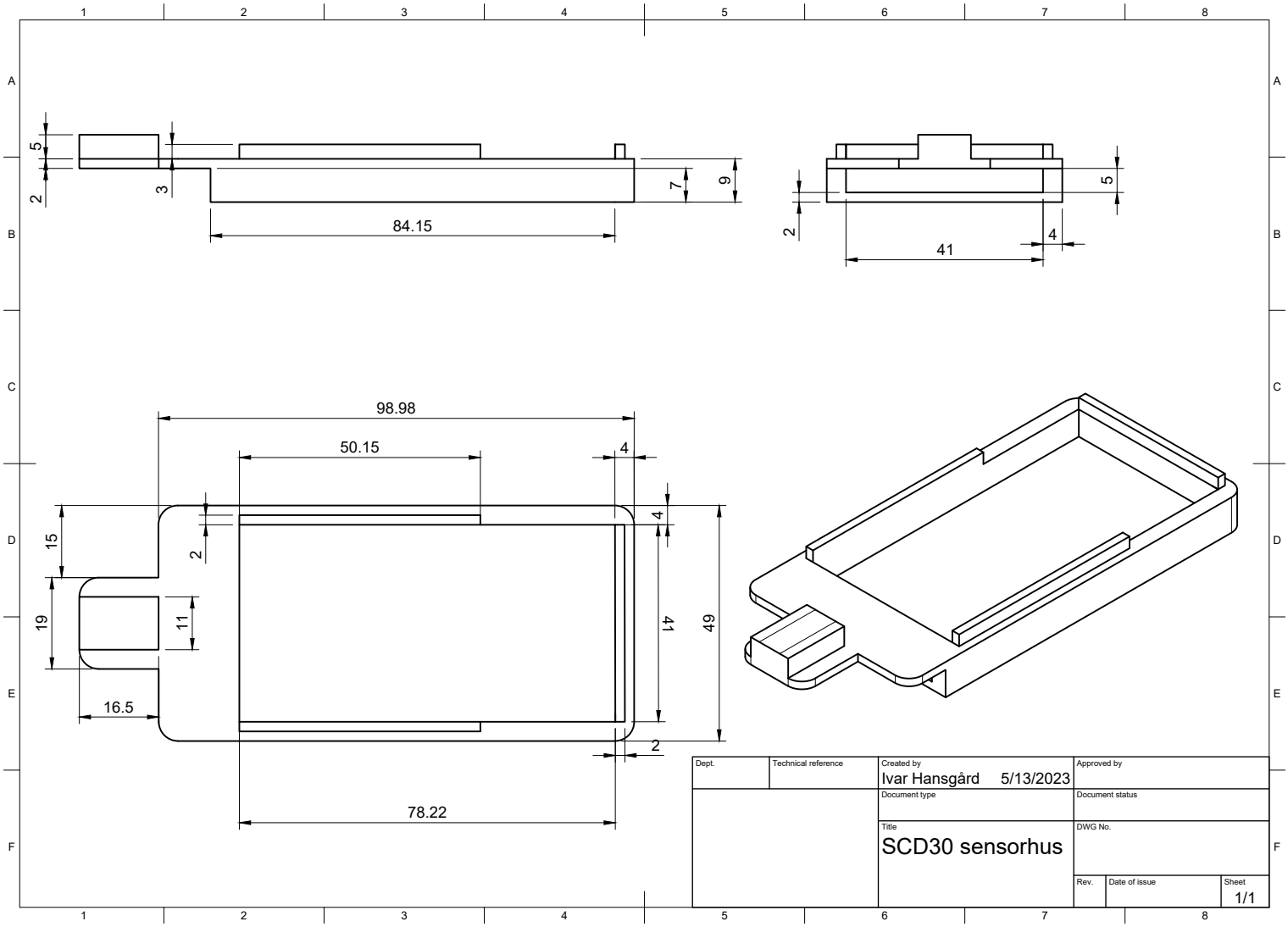
[A.4] AS7341 / BH1750 sensor hus



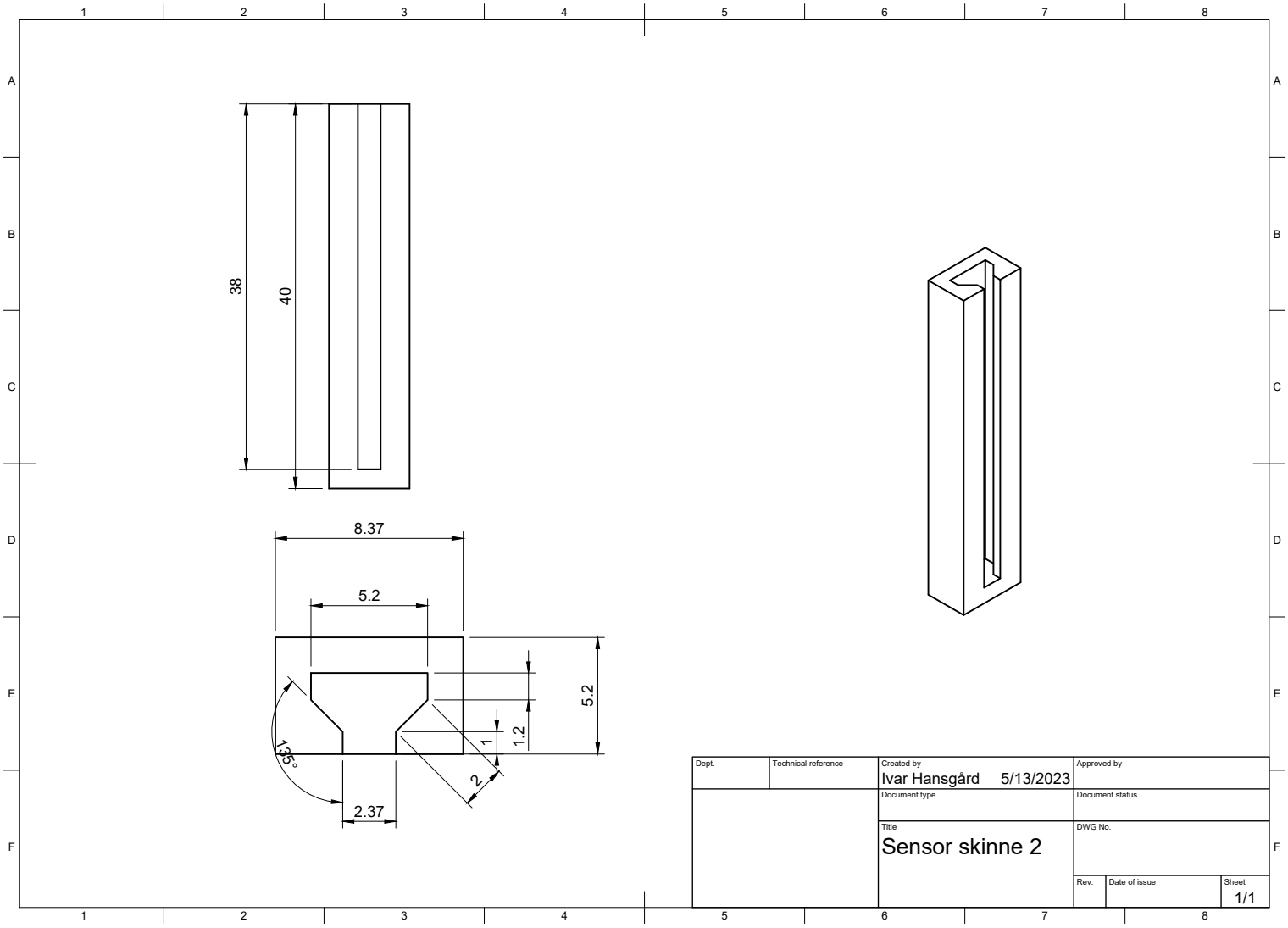
[A.5] SCD30 hus bun



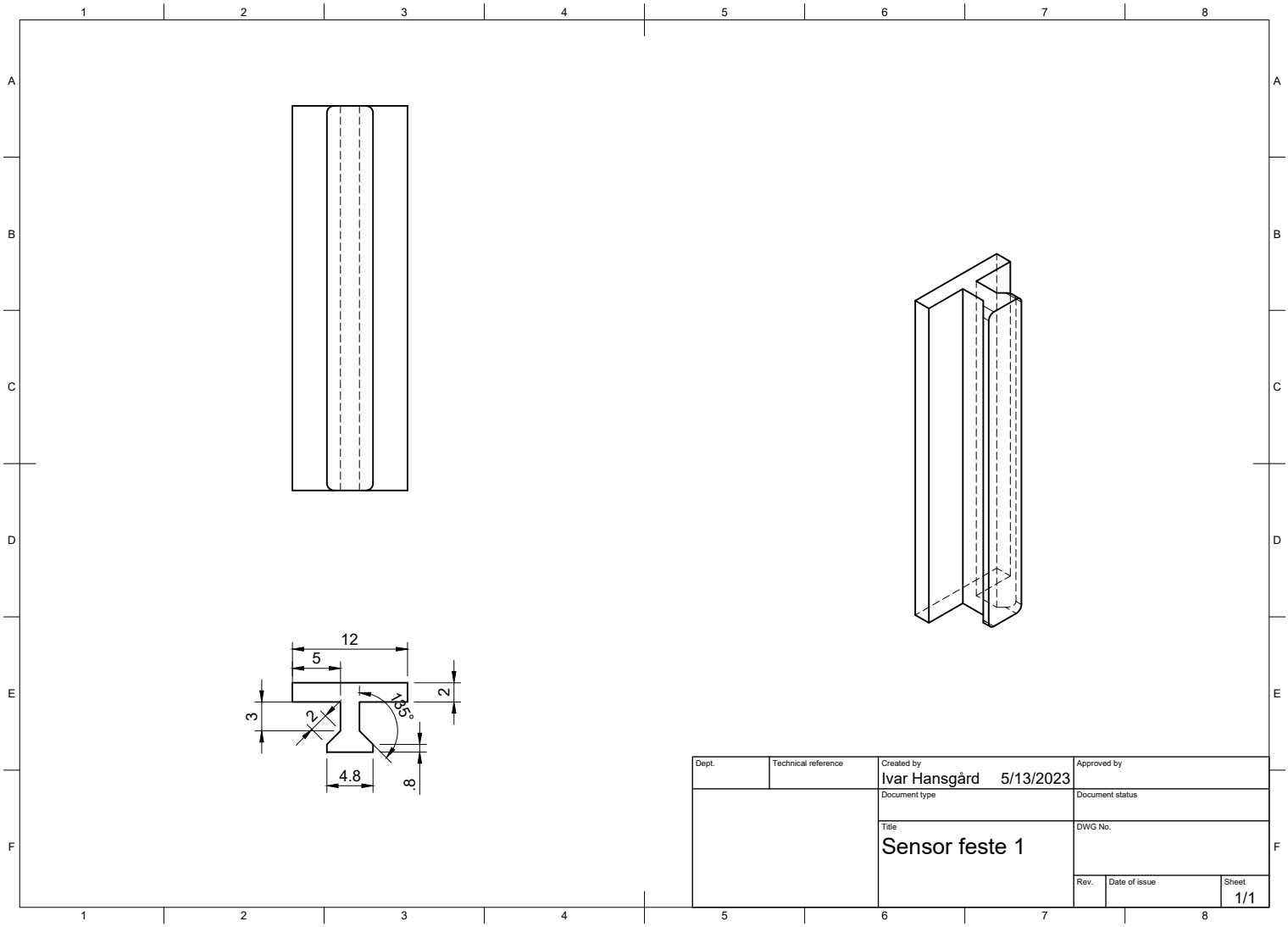
[A.6] SCD30 hus top



[A.7] Sensor feste



[A.8] Sensor skinne



Dept.	Technical reference	Created by Ivar Hansgård 5/13/2023	Approved by
		Document type	Document status
		Title: Sensor feste 1	DWG No.
			Rev. / Date of issue / Sheet 1/1

Vedlegg B

Kode

[B.1] CubeCell kode

```
/*
 * This program is created for CubeCell - Dev-Board(V2) HTCC-AB01(V2)
 * Created by Ivar Hansgård and Marius Marthinsen
 * The purpose of the program is to collect data from diffrent sensors and send it to thingstack thru LoRaWAN.
 * The program can be modified to include more sensors or to change out sensors, see guide -->
 * https://soldrevne-sensorbokser.gitbook.io/agrivoltaics-sensorboxes-documentation/?fbclid=IwAR3VUQUcgZCHTAPZnGT2KMgJ8DO0GakS-EMYXsXnoaGo5td8hl_pqnvlNjA
 */

#include "Arduino.h"
#include "LoRaWan_APP.h" //Lorawan, communication library.
#include <Wire.h>        //I2c communcation library.


// include
#include <BH1750.h>            // Lightsensor bh1750 library.
// BH1750 library by Christopher Laws. Guide: https://randomnerdtutorials.com/arduino-bh1750-ambient-light-sensor/

#include <Adafruit_AS7341.h>   // Multispecter sensor AS7341 library.
//Adafruit AS7341 library. Guide: https://learn.adafruit.com/adafruit-as7341-10-channel-light-color-sensor-breakout/arduino

#include "ADS1X15.h"           // Analog to digital converter  ads1115 library. ADC is connected to SEN0193 soil moisture sensor.
// Ads1X15-WE library by Rob tillaart. Guide: https://github.com/RobTillaart/ADS1X15

#include "SCD30.h"             // Co2, temperature and humidity sensor SCD30 library.
// Seeed SCD30 Library. Guide: https://github.com/Seeed-Studio/Seeed_SCD30

#include "Adafruit_MCP9600.h"  // mcp9600 termocouple i2c amplifier.
// Adafruit MCP9600 library. Guide https://learn.adafruit.com/adafruit-mcp9600-i2c-thermocouple-amplifier


// i2c addresses
#define TCAADDR 0x70    // I2c multiplexer address 0x70 in hex, 112 in decimal.
BH1750 lightMeter;      // using library, standard address is 0x23 in hex, 35 in decimal.
Adafruit_AS7341 as7341; // Using library, standard address is 0x39 in hex, 57 in   decimal.
ADS1115 ADS(0x48);      // Analog to digital converter  ads1115, address 0x48 in hex, 72 in decimal.
#define MCP9600 (0x60)  // mcp9600, using library, standard address is 0x60 in hex, 96 in decimal
Adafruit_MCP9600 mcp;
//the address for scd30, Co2, temperature and humidity sensor is set at 0x61 in hex, 97 in decimal


// Array to place data in a specified order before sending.
// Element [0] is port number, element [1] is sensor address. The rest of the elements is sensor data.

int32_t data_array_0[12];
int32_t data_array_1[12];
int32_t data_array_2[12];
int32_t data_array_3[12];
int32_t data_array_4[12];
int32_t data_array_5[12];
int32_t data_array_6[12];
int32_t data_array_7[12];



// declaring sensor arrays and variables for use:
//Arrays
int32_t multispecter_sensor_value[10];  // Data element order,wavelength nm:415,445,480,515,555,590,630,680,clear,nir. Multispecter sensor sensor AS7341.
float   scd30_sensor_value[3];          // Data element order: co2 in ppm, temperature in celsius, humidity in percentage. Co2, temperature and humidity sensor SCD30.
//variables
int32_t sensor_value_bh1750;            // Variable to store sensor values from lightsensor BH1750.
int32_t sensor_value_SEN0193;           // variable to store sensor values from Analog to digial converter ADS1115. ADC is connected to SEN0193 soil moisture sensor.
int32_t sensor_value_mcp9600;           // variable to store snesor values from mcp9600 connected to a thermocouple J.


uint8_t packageNum = 0;



//LoraWAN (find these in The Things Stack when creating a new node)
/* OTAA para*/
uint8_t devEui[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
uint8_t appEui[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
uint8_t appKey[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

/* ABP para*/
uint8_t nwkSKey[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
uint8_t appSKey[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
uint32_t devAddr =  ( uint32_t )0x00000000;

/*LoraWan channelsmask, default channels 0-7*/
uint16_t userChannelsMask[6] = { 0x00FF, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000 };

/*LoraWan region, select in arduino IDE tools*/
LoRaMacRegion_t loraWanRegion = ACTIVE_REGION;

/*LoraWan Class, Class A and Class C are supported*/
DeviceClass_t  loraWanClass = LORAWAN_CLASS;

/*the application data transmission duty cycle.  value in [ms].*/
uint32_t appTxDutyCycle = 15000;

/*OTAA or ABP*/
bool overTheAirActivation = LORAWAN_NETMODE;

/*ADR enable*/
bool loraWanAdr = LORAWAN_ADR;

/* set LORAWAN_Net_Reserve ON, the node could save the network info to flash, when node reset not need to join again */
bool keepNet = LORAWAN_NET_RESERVE;

/* Indicates if the node is sending confirmed or unconfirmed messages */
bool isTxConfirmed = LORAWAN_UPLINKMODE;

/* Application port */
uint8_t appPort = 2;

uint8_t confirmedNbTrials = 4;

//function for preparing sending of data
static void prepareTxFrame( uint8_t port, uint8_t sending_data)
{
  //variables
  uint8_t sensor;
  int32_t data[22];
  uint16_t batteryVoltage = getBatteryVoltage();
```

[B.1] CubeCell kode

```
Serial.print(sending_data);
//check wich data_array to send from with the sending_data variable and fill the data[] array with values from that array
switch (sending_data) {
  //send data from connector 1 / multiplexer pin 1
  case 0:
    for (uint8_t i = 0; i < 12; i++)
    {
      data[i] = data_array_0[i];
    }
    break;
  //send data from connector 2 / multiplexer pin 2
  case 1:
    for (uint8_t i = 0; i < 12; i++)
    {
      data[i] = data_array_1[i];
    }
    break;
  //send data from connector 3 / multiplexer pin 3
  case 2:
    for (uint8_t i = 0; i < 12; i++)
    {
      data[i] = data_array_2[i];
    }
    break;
  //send data from connector 4 / multiplexer pin 4
  case 3:
    for (uint8_t i = 0; i < 12; i++)
    {
      data[i] = data_array_3[i];
    }
    break;
  //send data from lightsensor / multiplexer pin 5
  case 4:
    for (uint8_t i = 0; i < 12; i++)
    {
      data[i] = data_array_4[i];
    }
    break;
  //send data from the spectrometer / multiplexer pin 6
  case 5:
    for (uint8_t i = 0; i < 12; i++)
    {
      data[i] = data_array_5[i];
    }
    break;

  case 6: //send battery
    data[1] = 0xff;
    break;

  //cases for two extra connectors
  /*
    case 7: //sensor name / multiplexer 7
    for (uint8_t i = 0; i < 12; i++)
    {
      data[i] = data_array_6[i];
    }
    break;

    case 8: //sensor name / multiplexer pin 8
    for (uint8_t i = 0; i < 12; i++)
    {
      data[i] = data_array_7[i];
    }
    break;
  */

  //should not enter this case somthing is wrong if you do
  default:
    Serial.println("Entered default case sending_data");
    break;
}

Serial.print(":" );
Serial.println(data[0]);

//the first value in the array is the sensor adress, this switch checks which sensor values are present and formats the appdata correctly for sending
switch (data[1]) {
  case 0x39:  //multispecter sensor

    appDataSize = 23;
    //
    appData[0] = data[0]; //connector
    appData[1] = data[1]; //sensor
    //
    appData[3] = (uint8_t)(data[2] >> 8); //405_425nm
    appData[4] = (uint8_t)(data[2]);
    //
    appData[5] = (uint8_t)(data[3] >> 8); //435_455nm
    appData[6] = (uint8_t)(data[3]);
    //
    appData[7] = (uint8_t)(data[4] >> 8); //nm470_490nm
    appData[8] = (uint8_t)(data[4]);
    //
    appData[9] = (uint8_t)(data[5] >> 8); //505_525nm
    appData[10] = (uint8_t)(data[5]);
    //
    appData[11] = (uint8_t)(data[6] >> 8); //545_565nm
    appData[12] = (uint8_t)(data[6]);
    //
    appData[13] = (uint8_t)(data[7] >> 8); //580_600nm
    appData[14] = (uint8_t)(data[7]);
    //
    appData[15] = (uint8_t)(data[8] >> 8); //620_640nm
    appData[16] = (uint8_t)(data[8]);
    //
    appData[17] = (uint8_t)(data[9] >> 8); //670_690nm
    appData[18] = (uint8_t)(data[9]);
    //
    appData[19] = (uint8_t)(data[10] >> 8); //Clear
    appData[20] = (uint8_t)(data[10]);
    //
    appData[21] = (uint8_t)(data[11] >> 8); //NIR
```

[B.1] CubeCell kode

```
      appData[22] = (uint8_t)(data[11]);
      //
      break;

    case 0x61:  //CO2 - Temp - Humidity sensor
      appDataSize = 10;
      //
      appData[0] = data[0]; //connector
      appData[1] = data[1]; //sensor
      //
      appData[2] = (uint8_t)(data[2] >> 8); //CO2 first byte
      appData[3] = (uint8_t)(data[2]);  //CO2 second byte
      //
      appData[4] = (uint8_t)(data[3] >> 24); //4 bytes of temperature
      appData[5] = (uint8_t)(data[3]) >> 16; //We need to send 4 bytes of the temp value because this value can be negative,
      appData[6] = (uint8_t)(data[3] >> 8); //it will be converted in the webserver to either negative or posititve
      appData[7] = (uint8_t)(data[3]); // see the hexToPosOrNegNum function
      //
      appData[8] = (uint8_t)(data[4] >> 8); //humidity first byte
      appData[9] = (uint8_t)(data[4]); //humidity second byte

      break;

    case 0x48:  //soil moisture sensor
      appDataSize = 4;
      //
      appData[0] = data[0]; //connector
      appData[1] = data[1]; //sensor
      //
      appData[2] = (uint8_t)(data[2] >> 8); //soil moisture first byte
      appData[3] = (uint8_t)(data[2]);     //soil moisture second byte
      break;

    case 0x23:  //light sensor
      appDataSize = 4;
      //
      appData[0] = data[0]; //connector
      appData[1] = data[1]; //sensor
      //
      appData[2] = (uint8_t)(data[2] >> 8); //lux value first byte
      appData[3] = (uint8_t)(data[2]); //lux value second byte

      break;


    case 0x60:  //mcp9600
      appDataSize = 6;
      //
      appData[0] = data[0]; //connector
      appData[1] = data[1]; //sensor
      //
      appData[2] = (uint8_t)(data[2]  >> 24); //4 bytes of temperature
      appData[3] = (uint8_t)(data[2]) >> 16;  //We need to send 4 bytes of the temp value because this value can be negative,
      appData[4] = (uint8_t)(data[2]  >> 8);  //it will be converted in the webserver to either negative or posititve
      appData[5] = (uint8_t)(data[2]);        //see the hexToPosOrNegNum function.
      break;

    case 0xff: //battery
      appDataSize = 6;
      //
      appData[0] = 9; //connector
      appData[1] = data[1]; //sensor
      //
      appData[2] = (uint8_t)(batteryVoltage >> 8); //battery voltage first byte
      appData[3] = (uint8_t)(batteryVoltage); //battery voltage second byte
      //
      appData[4] = 0; //send 0 because no data
      appData[5] = 0; //send 0 because no data
      break;

    //add more cases here if you add more sensors

    default:
      Serial.println("Entered default case sensor sending code ");
      appDataSize = 4;
      //
      appData[0] = data[0]; //connector
      appData[1] = data[1]; //sensor
      //
      appData[2] = 0; //send 0 because no data
      appData[3] = 0; //send 0 because no data
      break;

  }
}

// ----------------------------

// Functions for i2c multiplexer TCA9548
// Both functions was created by using resources from https://learn.adafruit.com/adafruit-tca9548a-1-to-8-i2c-multiplexer-breakout/arduino-wiring-and-test.

// Function to choose which scl and sda pins to read from on i2c multiplexer TCA9548. where "i" is the choosen scl and sda pins to read from.
void tcaselect(int i) {
  if (i > 7) return;

  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}

// Function to read from selected "t" which sensor connected, returns a decimal address value
int check_sensor_address(int t)
{
  int addr = 0;
  tcaselect(t);
  for (addr; addr <= 127; addr++)
  {
    if (addr == TCAADDR) continue;
    Wire.beginTransmission(addr);
    if (!Wire.endTransmission() && addr != 0)
    {
      return (addr);
    }
  }
  return 0;
```

[B.1] CubeCell kode

```
}

// ----------------------

// Functions to collecting data from sensors.

// Function for lightsensor bh1750, reads the sensor and returns a lux value
// Created by using resources from https://randommerdtutorials.com/arduino-bh1750-ambient-light-sensor/
void lightsensor_BH1750()
{
  lightMeter.begin();
  for (int k = 0; k < 2; k++)
  {
    delay(500);
    uint16_t lux = lightMeter.readLightLevel();
    delay(100);
    lux = lightMeter.readLightLevel();
    sensor_value_bh1750 = lux;
  }
}


// Function to collect data from multispecter sensor AS7341
// Function uses the function from as7341 library and stores the values inn the array multispecter_sensor_value.
// The function can be adjusted, the adjustments will change the amount of time it uses to read the light and how much gain the output will have.
// Created by using the resources from https://learn.adafruit.com/adafruit-as7341-10-channel-light-color-sensor-breakout/arduino.

void multispecter_as7341()
{
  if (as7341.begin())
  {
    as7341.setATIME(100); // integrator time
    as7341.setASTEP(999); // integrator time
    as7341.setGain(AS7341_GAIN_128X);   //gain, x0.5, x1,x2, x4, x8, x18, x32, x64, x128, x256, x512

    multispecter_sensor_value[0] = as7341.getChannel(AS7341_CHANNEL_415nm_F1); //F1(405-425nm
    multispecter_sensor_value[1] = as7341.getChannel(AS7341_CHANNEL_445nm_F2); //F2(435-455nm
    multispecter_sensor_value[2] = as7341.getChannel(AS7341_CHANNEL_480nm_F3); //F3(470-490nm
    multispecter_sensor_value[3] = as7341.getChannel(AS7341_CHANNEL_515nm_F4); //F4(505-525nm
    multispecter_sensor_value[4] = as7341.getChannel(AS7341_CHANNEL_555nm_F5); //F5(545-565nm
    multispecter_sensor_value[5] = as7341.getChannel(AS7341_CHANNEL_590nm_F6); //F6(580-600nm
    multispecter_sensor_value[6] = as7341.getChannel(AS7341_CHANNEL_630nm_F7); //F7(620-640nm
    multispecter_sensor_value[7] = as7341.getChannel(AS7341_CHANNEL_680nm_F8); //F8(670-690nm
    multispecter_sensor_value[8] = as7341.getChannel(AS7341_CHANNEL_CLEAR);    // Clear
    multispecter_sensor_value[9] = as7341.getChannel(AS7341_CHANNEL_NIR);     // NIR
  }
  if (!as7341.readAllChannels())
  {
    Serial.println("Error reading all channels!");
  }
}

// function to fill the variable sensor_value_sen0193, uses the 0 pin on ADC, values is a % from 0 to 100.
// function is created by using resources from https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193

void SEN0193()
{ for (int k = 0; k < 2; k++)
  {
    int sensor_raw_value        = 0;
    int sensor_prosentage_value = 0;
    // change the value on air_value and water_value from results from calibration
    const int air_value   = 8700;    //example values from my readings  : 8700
    const int water_value = 192;     //example values from my readings : 192

    sensor_raw_value = ADS.readADC(0); //reads the adc value from pin 0 on the adc
    sensor_prosentage_value = map(sensor_raw_value, air_value, water_value, 0, 100);

    if (sensor_prosentage_value >= 100)
    {
      sensor_value_SEN0193 = 100;
    }
    else if (sensor_prosentage_value <= 0)
    {
      sensor_value_SEN0193 = 0;
    }
    else if (sensor_prosentage_value > 0 && sensor_prosentage_value < 100)
    {
      sensor_value_SEN0193 = sensor_prosentage_value;
    }
  }
}


// function to fill variable sensor_value_mcp9600 with temperature from thermocouple sensor type J. values can be negative and positive.
// function is created by using resources from https://learn.adafruit.com/adafruit-mcp9600-i2c-thermocouple-amplifier/arduino and example code
void mcp9600()
{
  mcp.begin(MCP9600);
  delay(500);
  if (! mcp.begin(MCP9600)) {
    Serial.println("Sensor not found. Check wiring!");

  }
  else {
    mcp.setADCresolution(MCP9600_ADCRESOLUTION_18);
    mcp.setThermocoupleType(MCP9600_TYPE_J);
    mcp.setFilterCoefficient(3);
    mcp.enable(true);
    for (int k = 0; k < 2; k++)
    {
      sensor_value_mcp9600 = round(mcp.readThermocouple());
    }
  }
}


// Function to turn on or off the gpio1, by choosing "x" to be either "0"(off) or "1"(on).
// Created by using resources from https://docs.heltec.org/en/node/cubecell/frequently_asked_questions.html?fbclid=IwAR3SiM3gACF898N_-fueX9G3EJ4GC8tko5Bf0S78U1mE5DZAPlOEK2IdbHo
void Turn_on_off_sensor(int x)
{
  pinMode(GPIO1, OUTPUT);

  if (x == 0)
  {
```

```
    digitalWrite(GPIO1, LOW);
  }
  if (x == 1)
  {
    digitalWrite(GPIO1, HIGH);
  }

}
```

[B.1] CubeCell kode

```
// i is connector in use, sensor_array is the array you want to fill.
// the function goes thru connector to see what sensor connected and then fills the chosen array with those values

void fill_data_array(int i, int32_t sensor_array[12])
{
  uint8_t sensoradress = check_sensor_address(i);

  tcaselect(i);

  sensor_array[0] = i;
  sensor_array[1] = sensoradress;
  switch (sensoradress)
  {
    case 0x23:   //lightsensor BH1750.
      lightsensor_BH1750();
      sensor_array[2] = sensor_value_bh1750;

      break;

    case 0x48:   //Analog to digital converter ads1115, connected to SEN0193 soil moisture sensor.
      SEN0193();
      sensor_array[2] = sensor_value_SEN0193;
      break;

    case 0x61:   //Co2, temperature and humidity sensor SCD30.

      scd30.initialize();
      delay(500);

      if (scd30.isAvailable())
      {

        scd30.getCarbonDioxideConcentration(scd30_sensor_value); //fills the array with new values.

        uint16_t(scd30_sensor_value);

      }
      for (int i = 0; i < 3; i++)
      {
        sensor_array[i + 2] = scd30_sensor_value[i];
      }
      break;

    case 0x39:   //multispecter sensor AS7341.
      for (int i = 0; i < 2; i++) // has to go thru a loop,needs time to collect data, brute force solution
      {

        multispecter_as7341(); //fills the array with new values.

        for (int i = 0; i < 11; i++)
        {
          sensor_array[i + 2]  = multispecter_sensor_value[i];
        }
      }
      break;

    case 0x60:   //mcp9600.
      mcp9600();
      sensor_array[2] = sensor_value_mcp9600;
      break;
  }

}
void get_sensor_data()  //lage funksjon, av det nedenfor.
{
  // Data collecting part of code.
  // Starts of by turning on sensors and resetting all data arrays to 0.
  // Then reads and stores sensor address values to each individual connector variable called sensoradress.
  // The for loop goes thru each connector and is using the sensor address to sort out which sensor is in use in that port.
  // Then it uses the sensor address to choose which function to use on that connector thereafter it fills the variable or array with values.
  // the array is formated as data_array_ [connector, sensor, data values, ...]

  Turn_on_off_sensor(true); //turns on or off gpio0, true = on, false = off
  delay(2000); //adjust this according to delay you want

  //reset data_array_ to zero
  memset(data_array_0, 0, sizeof(data_array_0)); //etter array er brukt sett til 0
  memset(data_array_1, 0, sizeof(data_array_1));
  memset(data_array_2, 0, sizeof(data_array_2));
  memset(data_array_3, 0, sizeof(data_array_3));
  memset(data_array_4, 0, sizeof(data_array_4));
  memset(data_array_5, 0, sizeof(data_array_5));
  memset(data_array_6, 0, sizeof(data_array_6));
  memset(data_array_7, 0, sizeof(data_array_7));

  for (int i = 0; i < 8; i++)  //a loop to go thru connector 0 to 7, on the i2c multiplexer.
  {
    switch (i)
    {
      case 0: //Connector 0 on i2c multiplexer
        fill_data_array(0, data_array_0);
        break;

      case 1: //connector 1
        fill_data_array(1, data_array_1);
        break;

      case 2: //connector 2
        fill_data_array(2, data_array_2);
        break;

      case 3: //connector 3
        fill_data_array(3, data_array_3);
        break;
```

[B.1] CubeCell kode

```
      case 4: //connector 4
        fill_data_array(4, data_array_4);
        break;

      case 5: //connector 5
        fill_data_array(5, data_array_5);

        break;

      case 6: //connector 6
        fill_data_array(6, data_array_6);
        break;

      case 7: //connector 7
        fill_data_array(7, data_array_7);
        break;
    }
  }

  delay(2000);
  Turn_on_off_sensor(false);
}

void setup() {
  Serial.begin(115200);
  Serial.println("Setup start");  //to mark start for testing.
  Wire.begin();


#if(AT_SUPPORT)
  enableAt();
#endif
  deviceState = DEVICE_STATE_INIT;
  LoRaWAN.ifskipjoin();
}


void loop()
{
  switch ( deviceState )
  {
    case DEVICE_STATE_INIT:
      {
#if(AT_SUPPORT)
        getDevParam();
#endif
        printDevParam();
        LoRaWAN.init(loraWanClass, loraWanRegion);
        deviceState = DEVICE_STATE_JOIN;
        break;
      }
    case DEVICE_STATE_JOIN:
      {
        LoRaWAN.join();
        break;
      }
    case DEVICE_STATE_SEND:
      {
        if (packageNum == 0 )
        {
          get_sensor_data();
        }

        delay(500);
        prepareTxFrame(appPort, packageNum);
        LoRaWAN.send();
        packageNum++;
        if (packageNum == 7) //change this to 9 if you want to add two more connectors
        {
          packageNum = 0;
          //put to sleep here -- deepsleep/ sleep ? --> watchdog ?
        }
        deviceState = DEVICE_STATE_CYCLE;
        break;
      }

    case DEVICE_STATE_CYCLE:
      {
        // Schedule next packet transmission
        txDutyCycleTime = appTxDutyCycle + randr( 0, APP_TX_DUTYCYCLE_RND );
        LoRaWAN.cycle(txDutyCycleTime);
        deviceState = DEVICE_STATE_SLEEP;
        break;
      }
    case DEVICE_STATE_SLEEP:
      {
        LoRaWAN.sleep();
        break;
      }
    default:
      {
        deviceState = DEVICE_STATE_INIT;
        break;
      }
  }



}
```

[B.2.1] Webhook tjener hovedkode

```
//SOURCES
/* This code is based on these guides
Connecting and inserting data to a PostgreSQL database
- https://northflank.com/guides/connecting-to-a-postgresql-database-using-node-js
Consuming and formating JSON webhooks
- https://medium.com/@BearerSH/consuming-webhooks-with-node-js-and-express-50e007fc7ae2
*/

//js file for connecting to database
const { getClient } = require('./get-client');

//require express for webhook functionality, body-parser for json parsing and float-array-to-string for converting hex array to float
const express = require("express")
const bodyParser = require("body-parser")
const floatConverter = require("float-array-to-string") //not used but can be used in further itterations

//function to convert hex string to deximal
const hexToDecimal = hex => parseInt(hex, 16);
const binToDecimal = bin => parseInt(bin, 2);

//function to convert signed 32 bit ints to decimal usefull for sensors that have negative values
//pass it a 4byte hex number eks (0x00FF00FF) and it will convert it to a decimal value
hexToNegPosNum = (hexNum) => {
    let binString = '';

    if(parseInt(hexNum,16).toString(2).length < 32){
        //if the number is less than 32 bits long this padds start of the string with zeros to make it 32 bits long
        //converts it also to string of binary numbers
        binString =  hexToDecimal(hexNum).toString(2).padStart(32,'0')
    }
    else{
        //converts hexadecimal number to binary string
        binString = hexToDecimal(hexNum).toString(2)
    }

    if(binString[0] == '1'){
        // 1 as msb means that number is negative, flipp the bits in the number (~) add 1 and convert it to negative decimal
        return (-(~(binToDecimal(binString))+1))
    }
    else{
        //0 as msb means the number is positive and we can convert it directly to decimal
        return (binToDecimal(binString))
    }
}

//function for formating data from the JSON file into an usable array that can be inserted into the database
function formatData(data){
    //create variables
    let returnArray
    let sensorData = []

    sensorData = data.uplink_message.decoded_payload.dataArray //Data from thingsstack in hex format
    batteryValue = data.uplink_message.decoded_payload.battery //battery value from thingsstack in decimal value
    sensorType =  parseInt(data.uplink_message.decoded_payload.sensor) //Sensor adress in hex format
    sensorboxConnector = parseInt(data.uplink_message.decoded_payload.connector) //Connector number
    sensorboxID = data.end_device_ids.device_id //Device id from thingsstack

    switch(sensorType){
        case 23: //Light meter
            sensorData = hexToDecimal(sensorData[0]+sensorData[1]) //takes the hex values from sensordata combines them and turns them into decimal
            returnArray = [sensorType,sensorboxID,sensorboxConnector,sensorData]
            break;
        case 48: //Soil humidity / ADC
            sensorData = hexToDecimal(sensorData[0]+sensorData[1]) //takes the hex values from sensordata combines them and turns them into decimal
            returnArray = [sensorType,sensorboxID,sensorboxConnector,sensorData]
            break;
        case 39: //Spectrometer
            //Takes two and two hex values from the sensor data array combines them, turns them into decimal and puts them back into sensorData
            colorArray = []
            for(let i = 0; i < sensorData.length; i+=2){
                colorArray.push(hexToDecimal(sensorData[i]+sensorData[i+1]))
            }
            //
            returnArray = [sensorType,sensorboxID,sensorboxConnector,colorArray]
            break;
        case 61: //SCD30 Sensor
            //diffrent sensor values
            var co2Sensor = hexToDecimal(sensorData[0]+sensorData[1]) //takes the hex values from sensordata [0] and [1] combines them and turns them into decimal
            var tempSensor = hexToNegPosNum(sensorData[2]+sensorData[3]+sensorData[4]+sensorData[5]) //takes hex values at place [2],[3],[4] and [5] from sensorData and decodes them t
            var humiditySensor = hexToDecimal(sensorData[6]+sensorData[7])  //takes the hex values from sensordata [6] and [7] combines them and turns them into decimal

            sensorData = [co2Sensor,tempSensor,humiditySensor] //places co2, temp and humidity data into sensordata
            returnArray = [sensorType,sensorboxID,sensorboxConnector,sensorData] //returns [sensorType, sensorboxID,sensorboxConnector, sensorData[int,pos or neg int,int],]
            break;
        case 60: //Soil temperature
        sensorData = hexToNegPosNum(sensorData[0]+sensorData[1]+sensorData[2]+sensorData[3]) //takes hex values at place from sensorData and decodes them to a negative or positive val
        returnArray = [sensorType,sensorboxID,sensorboxConnector,sensorData]
        break;

        case 255: //battery
        returnArray = [sensorType,sensorboxID,sensorboxConnector,batteryValue]
        break;
        //insert new case her to add another sensor

        default:
            console.log(`Default case...`);
            returnArray = [null,sensorboxID,null,null,null] //returns 0 if there is no data
            break;
    }
    return returnArray
}

async function addData(data){
    //data[0] = sensorType
    //data[1] = sensorboxID
    //data[2] = sensorboxConnector
    //data[3] = sensorData[x] - size depends on amount of data

    const client = await getClient();
    switch(data[0]){
        /* not in use anymore but can be used for somthing else
        case null: //If the first number is null there is nothing connected to the box
            insertRow = await client.query(`INSERT INTO "Skjetleingard_test"(sensor_type,sensorbox_id,sensorbox_connector,value) VALUES(${data[0]},'${data[1]}',${data[2]},${data[3]});`
            console.log(`Inserted ${insertRow} row`);
            break;
        */
        case 255: //battery //inserts data into the database and prints it to the console
            insertRow = await client.query(`INSERT INTO "Skjetleingard_test"(sensorbox_id,sensorbox_connector,battery) VALUES('${data[1]}',${data[2]},${data[3]});`)
            console.log(`Inserted ${insertRow} row`);
        break;
```

[B.2.1] Webhook tjener hovedkode

```
        case 23: //Soil moisture / ADC //inserts data into the database and prints it to the console
            insertRow = await client.query(`INSERT INTO "Skjetleingard_test"(sensor_type,sensorbox_id,sensorbox_connector,value) VALUES(${data[0]},'${data[1]}',${data[2]},${data[3]});
            console.log(`Inserted ${insertRow} row`);
            break;
        case 48: //Light meter //inserts data into the database and prints it to the console
            insertRow = await client.query(`INSERT INTO "Skjetleingard_test"(sensor_type,sensorbox_id,sensorbox_connector,value) VALUES(${data[0]},'${data[1]}',${data[2]},${data[3]});
            console.log(`Inserted ${insertRow} row`);
            break;
        case 60: //Soil temperature / Soil Electrical Conductivity //inserts data into the database and prints it to the console
            insertRow = await client.query(`INSERT INTO "Skjetleingard_test"(sensor_type,sensorbox_id,sensorbox_connector,value) VALUES(${data[0]},'${data[1]}',${data[2]},${data[3]});
            console.log(`Inserted ${insertRow} row`);
        break;
        case 39: //Spectrometer //inserts data into the database and prints it to the console
            insertRow = await client.query(`INSERT INTO "Skjetleingard_test"(sensor_type,sensorbox_id,sensorbox_connector,nm405_425,nm435_455,nm470_490,nm505_525,nm545_565,nm580_600,nm
            console.log(`Inserted ${insertRow} row`);
            break;
        case 61: //SCD30 sensor CO2,Air Temperature, Humidity //inserts data into the database and prints it to the console
            insertRow = await client.query(`INSERT INTO "Skjetleingard_test"(sensor_type,sensorbox_id,sensorbox_connector,co2,temperature,humidity) VALUES(${data[0]}, '${data[1]}', ${
            console.log(`Inserted ${insertRow} row`);
            break;


        //insert new case her to add another sensor

        default: //if you end up here there is somthing wrong with the formated data or the modified code
            console.log(`Default case... does not insert anything in database`);
            break;
    }

    await client.end();
}

// Initialize express and define a port
const app = express()
//Port to run server on
const PORT = 3000

// Tell express to use body-parser's JSON parsing
app.use(bodyParser.json())

// Start express on the defined port
app.listen(PORT, () => console.log(`🚀 Server running on port ${PORT}`))

app.use(bodyParser.json())
app.post("/hook", (req, res) => {
    //debugging code, can be seen in the console
        console.log("----------------")
        console.log("Raw data: ",req.body.uplink_message.decoded_payload.rawInput)
        console.log("Data array: ", req.body.uplink_message.decoded_payload.dataArray)
        console.log("Connector: ",req.body.uplink_message.decoded_payload.connector)
        console.log("Sensor: 0x"+req.body.uplink_message.decoded_payload.sensor, " Decimal: ",req.body.uplink_message.decoded_payload.sensor)
        console.log("Formated data: ", formatData(req.body))
    //run the addData function with the formatData command on the JSON file sent by thingsstack
    addData(formatData(req.body))
    console.log("----------------")

    res.status(200).end() // Responding is important
})
//...
```

[B.2.2] Webhook tjener kode for å koble til database

```
const { Client } = require('pg'); //require pg for connecting to postgres
require('dotenv').config(); //require dotenv for extracting data from the .env file

//function for connecting to the postgres database and exporting the connected client
module.exports.getClient = async () => {
    const client = new Client({

        //should work with the dotenv package but doesnt can try to fix later
        /*
        host: process.env.PG_HOST,
        port: process.env.PG_PORT,
        user: process.env.PG_USER,
        password: process.env.PG_PASSWORD,
        database: process.env.PG_DATABASE,
        */

        //Enter database details
        ///*
        host: 'database url',
        port: 'port',
        user: 'username',
        password: 'password',
        database: 'database name',
        //*/
        ssl: true,
      });

  await client.connect();
  return client;
};
```

[B.3] The Things Stack formaterings skript

```
//Main function for converting data from CubeCell
function decodeUplink(input) {
  //create an array
  let byteArr = [];
  let batteryValue = 0;
  //puts all the bytes in the array from the input into byteArr
  for(let i = 0; i < input.bytes.length; i++){
    byteArr.push(input.bytes[i].toString(16));
  }

  //puts first value in byteArr into connector and deletes it
  let connector = byteArr.shift()
  //puts first value in byteArr into sensor and deletes it
  let sensor = byteArr.shift()

  //if connector = 9 then we are sending the battery value of the box
  if(connector == 9){
    //puts first value in battery_byte1 into connector and deletes it
    let battery_byte1 = byteArr.shift()
    //puts first value in battery_byte2 into connector and deletes it
    let battery_byte2 = byteArr.shift()
    //Takes battery_byte 1 and 2 combines them and turns the string into decimal
    batteryValue = parseInt((battery_byte1+battery_byte2),16)
    sensor = 255
  }

  //return a JSON file with all the variables
  return {
    data: {
      rawInput: input, //Raw input bytes (only used for debugging in the webserver)
      battery: batteryValue, //Battery value as a decimal number
      connector: connector, //The connector number in hex
      sensor: sensor, //The sensor adress in hex
      dataArray: byteArr //The rest of byteArr which should be the sensor values
    },
    warnings: [], // optional
    errors: [] // optional (if set, the decoding failed)
  };
}
```

[B.4] SCD30 kalibrerings kode

```
//This is a modified version of the original code for calibrating the scd30
// the original can be found https://github.com/Seeed-Studio/Seeed_SCD30/blob/master/examples/SCD30_auto_calibration/SCD30_auto_calibration.ino
// note must be running for 7 days, and spend 1 hour outside every day
// the sensor has to be plugged inn plugg 0


#include "Arduino.h"
#include <Wire.h>          //I2C communication library
#include "SCD30.h"
#define TCAADDR 0x70             // I2c multiplexer address 0x70 in hex, 112 in decimal.


#if defined(ARDUINO_ARCH_AVR)
#pragma message("Defined architecture for ARDUINO_ARCH_AVR.")
#define SERIAL Serial
#elif defined(ARDUINO_ARCH_SAM)
#pragma message("Defined architecture for ARDUINO_ARCH_SAM.")
#define SERIAL SerialUSB
#elif defined(ARDUINO_ARCH_SAMD)
#pragma message("Defined architecture for ARDUINO_ARCH_SAMD.")
#define SERIAL SerialUSB
#elif defined(ARDUINO_ARCH_STM32F4)
#pragma message("Defined architecture for ARDUINO_ARCH_STM32F4.")
#define SERIAL SerialUSB
#else
#pragma message("Not found any architecture.")
#define SERIAL Serial
#endif

// Functions for i2c multiplexer TCA9548
// Both functions was created by using resources from https://learn.adafruit.com/adafruit-tca9548a-1-to-8-i2c-multiplexer-breakout/arduino-wiring-and-test.
// Function to choose which scl and sda pins to read from on i2c multiplexer TCA9548. where "i" is the choosen scl and sda pins to read from.
void tcaselect(int i) {
  if (i > 7) return;

  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}


void Turn_on_off_sensor(int x)
{
  pinMode(GPIO1, OUTPUT);
  if (x == 0)
  {
    digitalWrite(GPIO1, LOW);
  }
  if (x == 1)
  {
    digitalWrite(GPIO1, HIGH);
  }
}

void setup() {
  Turn_on_off_sensor(true);
  delay(1000);
  tcaselect(0);
  Turn_on_off_sensor(true);
  Wire.begin();
  SERIAL.begin(115200);
  SERIAL.println("SCD30 Raw Data");
  scd30.initialize();
  //Calibration for minimum 7 days,after this ,close auto self calibration operation.
  scd30.setAutoSelfCalibration(1);
}

void loop() {
  float result[3] = {0};

  if (scd30.isAvailable()) {
    scd30.getCarbonDioxideConcentration(result);
    SERIAL.print("Carbon Dioxide Concentration is: ");
    SERIAL.print(result[0]);
    SERIAL.println(" ppm");
    SERIAL.println(" ");
    SERIAL.print("Temperature = ");
    SERIAL.print(result[1]);
    SERIAL.println(" â  ");
    SERIAL.println(" ");
    SERIAL.print("Humidity = ");
    SERIAL.print(result[2]);
    SERIAL.println(" %");
    SERIAL.println(" ");
    SERIAL.println(" ");
    SERIAL.println(" ");
  }

  delay(2000);
}
```

[B.5] SEN0193 kalibrerings kode

```
// created by using resources form https://how2electronics.com/interface-capacitive-soil-moisture-sensor-arduino/#Capacitive_Soil_Moisture_Sensor_Calibration
// and https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193 '
// use this calibration to get airvalue and watervalue to create a lowest value, highest value.
// the function will use these values to create a % of how wet the ground is.

#include "Arduino.h"
#include <Wire.h>                 //I2c communcation library.
#include "ADS1X15.h"              // Library for ADS1115 Analog to digital converter library. ADC is connected to SEN0193 soil moisture sensor.
#define TCAADDR 0x70              // I2c multiplexer address 0x70 in hex, 112 in decimal.
ADS1115 ADS(0x48);                // Analog to digital converter  ads1115, address 0x48 in hex, 72 in decimal.

int32_t sensor_value_SEN0193;   // variable to store sensor values from Analog to digial converter ADS1115. ADC is connected to SEN0193 soil moisture sensor.

// to calibrate, read of the raw values from air and write it down, insert it in air_value
// then put the sensor in water and read of the value from water and write it down

void SEN0193_calibration()
{
  int sensor_raw_value        = 0;
  int sensor_prosentage_value = 0;
  // change the value on air_value and water_value from results from
  const int air_value   = 8700;   //example values from my readings  : 8700
  const int water_value = 192;     //example values from my readings : 192

  sensor_raw_value = ADS.readADC(0); //reads the adc value from pin 0 on the adc
  sensor_prosentage_value = map(sensor_raw_value, air_value, water_value, 0, 100);
  Serial.print("raw data: "); Serial.println(sensor_raw_value);

  if (sensor_prosentage_value >= 100)
  {
    sensor_value_SEN0193 = 100;
  }
  else if (sensor_prosentage_value <= 0)
  {
    sensor_value_SEN0193 = 0;
  }
  else if (sensor_prosentage_value > 0 && sensor_prosentage_value < 100)
  {
    sensor_value_SEN0193 = sensor_prosentage_value;
  }
}
// Functions for i2c multiplexer TCA9548
// Both functions was created by using resources from https://learn.adafruit.com/adafruit-tca9548a-1-to-8-i2c-multiplexer-breakout/arduino-wiring-and-test.

// Function to choose which scl and sda pins to read from on i2c multiplexer TCA9548. where "i" is the choosen scl and sda pins to read from.
void tcaselect(int i) {
  if (i > 7) return;

  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}

// Function to read from selected "t" which sensor connected, returns a decimal address value
int check_sensor_address(int t)
{
  int addr = 0;
  tcaselect(t);
  for (addr; addr <= 127; addr++)
  {
    if (addr == TCAADDR) continue;
    Wire.beginTransmission(addr);
    if (!Wire.endTransmission() && addr != 0)
    {
      return (addr);
    }
  }
  return 0;
}
void Turn_on_off_sensor(int x)
{
  pinMode(GPIO1, OUTPUT);

  if (x == 0)
  {
    digitalWrite(GPIO1, LOW);
  }
  if (x == 1)
  {
    digitalWrite(GPIO1, HIGH);
  }
}

void check_connector_sensor(int i)
{
  uint8_t sensoradress = check_sensor_address(i);
  tcaselect(i);
  switch (sensoradress)
  {
    case 0x48:   //Analog to digital converter ads1115, connected to SEN0193 soil moisture sensor.
      SEN0193_calibration();
      Serial.print("value_sen; "); Serial.print(sensor_value_SEN0193); Serial.println(" %");
      delay(1000);
      break;
  }
}

void setup()
{
  Serial.begin(115200);
  Wire.begin();
  Turn_on_off_sensor(true);
```

```
}

void loop()
{
  for (int i = 0; i < 6; i++)  //a loop to go thru connector 0 to 5, on the i2c multiplexer.
  {
    check_connector_sensor(i);
    delay(100);
  }
}
```

[B.5] SEN0193 kalibrerings kode

Vedlegg C

BOM

[C.1] BOM full system

BOM for a full system with one node

Name	Type	Quantity	Price each	Total	Link	notes
Node						
CubeCell - Dev-Board (V2)	Microcontroller	1	kr 153,03	kr 153,03	https://heltec.org/project/htcc-ab01-v2/	It is inclued antenna.
TCA9548A I2C Multiplexer	I2C Multiplexer	1	kr 76,51	kr 76,51	https://www.adafruit.com/product/2717	
IP68 Box	Box, case for node	1	kr 73,30	kr 73,30	https://www.elfadistrelec.no/no/koblingsboks-med-blankt-lokk-120x160x70mm-termoplast-wiska-	
TE Connectivity Circular Connector, 4 Contacts, Cable Mount, M12 Series	Male connector	4	kr 63,12	kr 252,48	https://no.rs-online.com/web/p/industrial-circular-connectors/2373904	
TE Connectivity Circular Connector, 4 Contacts, Panel Mount Female M12 Series	Female connector	4	kr 63,00	kr 252,00	https://no.rs-online.com/web/p/industrial-circular-connectors/1345766	
0.5W Solar Panel 55x70	Solar panel	2	kr 33,86	kr 67,72	https://www.elfadistrelec.no/en/5w-solar-panel-5v-100ma-55-70mm-seeed-studio-	
Rechargeable lithium-ion battery 3.6V 2600m	Lithium-ion battery	2	kr 216,63	kr 433,26	https://www.elfadistrelec.no/en/rechargeable-battery-li-ion-18650-6v-6ah-ansmann-1307-	
1049 - Battery Holder 2x 18650 Through Hole, Keystone	Battery holder	1	kr 63,70	kr 63,70	https://www.elfadistrelec.no/en/battery-holder-2x-18650-through-hole-keystone-	
M5 Nut	M5 Nut	1	kr 45,00	kr 45,00	https://www.elfadistrelec.no/no/sekskantmutre-rustfrie-a2-m5-4mm-rustfritt-stal-bossard-bn-	
Adafruit Perma-Proto Quarter-sized Breadboard PCB - Single	PCB	1	kr 32,81	kr 32,81	https://www.adafruit.com/product/1608	
Adafruit Perma-Proto Half-sized Breadboard PCB - Single	PCB	1	kr 49,54	kr 49,54	https://www.adafruit.com/product/1609	
ADS1115 16-Bit ADC - 4 Channel with Programmable Gain Amplifier	Analog to digital converter	1	kr 164,59	kr 164,59	https://www.adafruit.com/product/1085	
bd679	Transistor	1	kr 4,61	kr 4,61	https://www.elfadistrelec.no/en/darlington-transistor-npn-80v-4a-sot-32-st-	
resistor 10kohm	Resistor	1	kr 2,20	kr 2,20	https://www.elfadistrelec.no/en/resistor-600mw-10kohm-vishay-	
RE520-HP, Single-Sided Stripboard	Board to use with battery holder	1	kr 47,51	kr 47,51	https://no.rs-online.com/web/p/stripboards/2065841?cm_mmc=NO-PLA-DS3A-_-google-_-	
Sensors						
Adafruit AS7341 10-Channel Light / Color Sensor	Lightsensor with spectrometer	1	kr 175,60	kr 175,60	https://www.adafruit.com/product/4698	
Adafruit BH1750 Light Sensor	Light sensor	1	kr 49,54	kr 49,54	https://www.adafruit.com/product/4681	
SCD30 CO2, Temperature and Humidity Sensor, Seeed Studio	CO2, Temperature and Humidity Sensor	1	kr 462,15	kr 462,15	https://www.elfadistrelec.no/en/scd30-co2-temperature-and-humidity-sensor-seeed-studio-	
DFRobot SEN0308	Analog Waterproof Capacitive Soil Moisture	1	kr 185,23	kr 185,23	https://no.rs-online.com/web/p/sensor-development-tools/2049905?cm_mmc=NO-PLA-DS3A-_-	
Alpha Wire Cat5 Ethernet Cable, S/FTP Shield, Grey PVC Sheath, 30m	Cabel for sensor	1	kr 1 153,24	kr 1 153,24	https://no.rs-online.com/web/p/ethernet-cable/1681385	This one has eight wires inside, could be
RS PRO Type J Thermocouple 150mm Length, 4.5mm Diameter	Thermocouple	1	kr 272,56	kr 272,56	https://no.rs-online.com/web/p/thermocouples/2194725	
Adafruit MCP9600 I2C Thermocouple Amplifier	I2c thermocouple amplifier board	1	kr 175,60	kr 175,60	https://www.adafruit.com/product/4101	
Slide Switch	Slide Switch	1	42,01	kr 42,01	toOYoGunpxaYlac6gHb2GFwi93rLN_EHNKcBCQ	switch can be changed out
Screw terminal	Screw terminal	1	kr 4,60	kr 4,60	https://no.rs-online.com/web/p/pcb-terminal-blocks/8971332	
JST PH 2mm 4-Pin to Male Header Cable - I2C STEMMA Cable - 200mm	Cable for BH1750 and AS7341 sensor	2	kr 16,51	kr 33,03	https://www.adafruit.com/product/3955	
Gateway						
HT-M02 Edge LoRa Gateway (V2)	Gateway	1,00	kr 2 752,33	kr 2 752,33	https://heltec.org/project/ht-m02-v2/	price may vary depending on which
				kr 0,00		
various materials						
HellermannTyton Heat Shrink Tubing, 3:1 Ratio, ShrinKit 321 Basic Series	Heatshrink kit	1	kr 358,65	kr 358,65	data=7365617263685F6361736361646555F6F726465723D312673656172636385F696E74657266616	
Assorted wire 7.6m, SparkFun Electronics	Assorted wire	1	kr 234,70	kr 234,70	electronics-prt-11367/p/30145492	
Epoxy lim	Epoxy lim	2	kr 717,27	kr 1 434,54	https://uk.rs-online.com/web/p/glues/9184804	
heatshrink	Heatshrink for amplifier board	1	kr 91,20	kr 91,20	https://no.rs-online.com/web/p/heat-shrink-tubing/7004668	
TOTAL				**kr 7 024,14**		

[C.2] BOM node

Parts for one node						
Name	**Type**	**Quantity**	**Price each**	**Total**	**Link**	**notes**
Node						
CubeCell - Dev-Board (V2)	Microcontroller	1	kr 153,03	kr 153,03	https://heltec.org/project/htcc-ab01-v2/	It is inclued antenna.
TCA9548A I2C Multiplexer	I2C Multiplexer	1	kr 76,51	kr 76,51	https://www.adafruit.com/product/2717	
IP68 Box	Box, case for node	1	kr 73,30	kr 73,30	https://www.elfadistrelec.no/no/koblingsboks-med-	
RS PRO Circular Connector, 4 Contacts, Cable Mount, Plug, Male, IP67	Male connector	4	kr 63,12	kr 252,48	https://no.rs-online.com/web/p/industrial-circular-	
RS PRO Circular Connector, 4 Contacts, Panel Mount, Socket, Female, IP67	Female connector	4	kr 63,00	kr 252,00	https://no.rs-online.com/web/p/industrial-circular-	
0.5W Solar Panel 55x70	Solar panel	2	kr 33,86	kr 67,72	https://www.elfadistrelec.no/en/5w-solar-panel-5v-	
Rechargeable lithium-ion battery 3.6V 2600m	Lithium-ion battery	2	kr 216,63	kr 433,26	https://www.elfadistrelec.no/en/rechargeable-battery-	
1049 - Battery Holder 2x 18650 Through Hole, Keystone	Battery holder	1	kr 63,70	kr 63,70	https://www.elfadistrelec.no/en/battery-holder-2x-	
M5 Thread insert	M5 Thread insert	1	kr 45,00	kr 45,00	https://www.elfadistrelec.no/no/sekskantmutre-	
Adafruit Perma-Proto Half-sized Breadboard PCB - Single	PCB	1	kr 49,54	kr 49,54	https://www.adafruit.com/product/1609	
ADS1115 16-Bit ADC - 4 Channel with Programmable Gain Amplifier	Analog to digital converter	1	kr 164,59	kr 164,59	https://www.adafruit.com/product/1085	
bd679	Transistor	1	kr 4,61	kr 4,61	https://www.elfadistrelec.no/en/darlington-transistor-	
resistor 10kohm	Resistor	1	kr 2,20	kr 2,20	https://www.elfadistrelec.no/en/resistor-600mw-	
RE520-HP, Single-Sided Stripboard	Board to use with battery holder	1	kr 47,51	kr 47,51	https://no.rs-online.com/web/p/stripboards/2065841?c	
Sensors						
Adafruit AS7341 10-Channel Light / Color Sensor	Lightsensor with spectrometer	1	kr 175,60	kr 175,60	https://www.adafruit.com/product/4698	
Adafruit BH1750 Light Sensor	Light sensor	1	kr 49,54	kr 49,54	https://www.adafruit.com/product/4681	
SCD30 CO2, Temperature and Humidity Sensor, Seeed Studio	CO2, Temperature and Humidity Sensor	1	kr 462,15	kr 462,15	https://www.elfadistrelec.no/en/scd30-co2-	
DFRobot SEN0308	Analog Waterproof Capacitive Soil Moisture	1	kr 185,23	kr 185,23	https://no.rs-online.com/web/p/sensor-development-	
RS PRO Type J Thermocouple 150mm Length, 4.5mm Diameter	Thermocouple	1	kr 272,56	kr 272,56	https://no.rs-online.com/web/p/thermocouples/2104735	
Adafruit MCP9600 I2C Thermocouple Amplifier	I2c thermocouple amplifier board	1	kr 175,60	kr 175,60	https://www.adafruit.com/product/4101	
Slide Switch	Slide Switch	1	kr 42,01	kr 42,01	switches/7347292?fbclid=IwAR01vgrYvqDapXZt7OWN-	The switch can be changed out.
Screw terminal	Screw terminal	1	kr 4,60	kr 4,60	blocks/8971332	
JST PH 2mm 4-Pin to Male Header Cable - I2C STEMMA Cable - 200mm	Cable for BH1750 and AS7341 sensor	2	kr 16,51	kr 33,03	https://www.adafruit.com/product/3955	
TOTAL				**kr 3 006,13**		

Vedlegg D

Strømbudsjett

[D.1] Strømbudsjett fra målinger

POWER BUDGET CREATED FROM MEASURING					
COMPONENT	**CONDITION**	**TYP VOLTAGE (V)**	**TYP CURRENT (A)**	**ACTIVE POWER (W)**	**COMMENTS**
Test 1: Test without sensors connected to connectors					
Whole Node	Data sending	3,6	0,01	0,036	
Whole Node	Data gathering	3,6	0,016	0,0576	
Test 2: Test with sensors connected to connectors					
Whole Node	Data sending	3,6	0,01	0,036	
Whole Node	Data gathering	3,6	0,031	0,1116	
Test 3: test of each sensor					
SCD30	Data gathering	3,6	0,045	0,162	
MCP9600/Earth temperature	Data gathering	3,6	0,0064	0,02304	
SEN0193/Earth moisture	Data gathering	3,6	0,003	0,0108	
Input power					
Solar panels	2 in paralell			-1	"-" in this case means that it does not draw power
Total test 3+1 datagathering:				0,25344	
Total test 3+1 - solar:				-0,74656	We should be charging

[D.2] Strømbudsjett fra datablad

POWER BUDGET CREATED FROM DATASHEETS					
COMPONENT	**CONDITION**	**TYP VOLTAGE (V)**	**TYP CURRENT (A)**	**ACTIVE POWER (W)**	**COMMENTS**
Node					
CubeCell	Transmitt mode	3,6	0,185	0,666	
TCA9548A I2C multiplexer	fscl = 400khz	3,6	0,00002	0,000072	
Sensor					
ADS1115		3,6	0,01	0,036	
AS7341		3,6	0,00021	0,000756	
BH1750		3,6	0,00012	0,000432	
SEN0193		3,6	0,005	0,018	
MCP9600		3,6	0,01	0,036	
SCD30		3,6	0,019	0,0684	
Input power					
Solar panels	two in parallel, assuming tyical power generation			-1	"-" in this case means that it does not draw power
TOTAL				0,82566	0,22935
TOTAL + 10%				0,908226	
TOTAL + 10% - Solar power				-0,091774	We should be charging

Vedlegg E

Testrapporter

TESTRAPPORT SOLDREVNE LoRaWAN NODER

[E.1] Testrapport for test av systemet

Ivar Hansgård og Marius Marthinsen

Table of Contents

[E.1] Testrapport for test av systemet

1. Sammendrag

For å validere at sensornodene vi har laget funker har vi utført tre tester per boks hvor vi har testet om data fra sensorene samt batterispenning blir sent hele veien fra sensornoden til Sintef databasen. Ut ifra testene fant vi ut at både boks 1 og boks 2 samt alle de tilhørende sensorene funket uten problemer og data kom fram til databasen.

2. Test prosedyre

2.1. Fremgangsmåte

Vi hadde to sett med sensorer og to bokser.

Vi utførte tre tester per boks for å finne ut om vi fikk data fra alle sensorene helt til databasen.

Test nummer en hvor vi testet i2c pluggene og sensorene om de funket sammen.

Test nummer to hvor vi testet ADC pluggen og de fasttilkoblete lyssensorene.

Test nummer tre hvor vi testet alle sensorene sammen.

2.2. Testmatrise

Testmatrisene ble satt opp med følgende parameter som varierte:

- Plugg
- Sensor

Testene er bestått eller ikke bestått avhengig av om dataen kom fram til The Things Stack eller helt til databasen

X betyr at testen har bestått og O betyr at testen er ikke bestått.

Test 1.

Boks: x							
Sensor	Plugg 0		Plugg 1		Plugg 2		Kommentar
	Things Stack	Database	Things Stack	Database	Things Stack	Database	
SCD30							
Thermocouple							

Test 2.

Boks: x			
Sensor	Plugg 4 / Fast tilkoblet		Kommentar
	Things Stack	Database	
SEN0193			
AS7341			
BH1750			

Batteri spenning			

Test 3.

[E.1] Testrapport for test av systemet

Boks: x			
Sensor			Kommentar
	Things Stack	Database	
SCD30			
Thermocouple			
SEN0193			
AS7341			
BH1750			
Batteri spenning			

3. Resultater

3.1. Boks 1

Test 1.

I2C sensorer plugget inn en etter en i hver av pluggene

Boks: 1							
Sensor	Plugg 0		Plugg 1		Plugg 2		Kommentar
	Things Stack	Database	Things Stack	Database	Things Stack	Database	
SCD30	X	X	X	X	X	X	
Thermocouple	X	X	X	X	X	X	

Test 2.

Sjekket om analog pluggen funker og om vi får data fra de fast loddete sensorene samt batteri spenningen.

Boks: 1			
Sensor	Plugg 4 / Fast tilkoblet		Kommentar
	Things Stack	Database	
SEN0193	X	X	
AS7341	X	X	
BH1750	X	X	
Batteri spenning	X	X	

Test 3.

Alle sensorene plugget inn samtidig.

Boks: 1		
Sensor		Kommentar

[E.1] Testrapport for test av systemet

	Things Stack	Database	
SCD30	X	X	
Thermocouple	X	X	
SEN0193	X	X	
AS7341	X	X	
BH1750	X	X	
Batteri spenning	X	X	

3.2. Boks 2

Test 1.

I2C sensorer plugget inn en etter en i hver av pluggene

Boks: 2							
Sensor	Plugg 0		Plugg 1		Plugg 2		Kommentar
	Things Stack	Database	Things Stack	Database	Things Stack	Database	
SCD30	X	X	X	X	X	X	
Thermocouple	X	X	X	X	X	X	

Test 2.

Sjekket om analog pluggen funker og om vi får data fra de fast loddete sensorene samt batteri spenningen.

Boks: 2			
Sensor	Plugg 4 / Fast tilkoblet		Kommentar
	Things Stack	Database	
SEN0193	X	X	
AS7341	X	X	
BH1750	X	X	
Batteri spenning	X	X	

Test 3.

Alle sensorene plugget inn samtidig.

Boks: 2			
Sensor			Kommentar
	Things Stack	Database	

SCD30	X	X	
Thermocouple	X	X	
SEN0193	X	X	
AS7341	X	X	
BH1750	X	X	
Batteri spenning	X	X	

[E.1] Testrapport for test av systemet

4. Konklusjon

4.1. Boks 1

Etter en gjennomgang av alle testene kan vi konkludere med alle sensorene til boks 1 funket og data kommer helt frem til databasen. Thermocouple og SCD30 har heller ingen problemer med å plugges inn i forskjellige plugger. Vi får også batteri spenning helt frem til databasen.

4.2. Boks 2

Etter en gjennomgang av alle testene kan vi konkludere med alle sensorene til boks 2 funket og data kommer helt frem til databasen. Thermocouple og SCD30 har heller ingen problemer med å plugges inn i forskjellige plugger. Vi får også batteri spenning helt frem til databasen.

TESTRAPPORT SOLDREVNE LoRaWAN NODER

[E.2] Testrapport for strømforbruk

STRØMFORBRUK

Ivar Hansgård og Marius Marthinsen

Table of Contents

1. Sammendrag

For finne ut hvor mye strøm hver av sensorene trekker og, sjekke om batteriet kan holde live i sensorboksen i lang nok tid har ble det utført tre tester. Ut ifra testene viser resultatet at hele boksen trekker maks 0.08mA ved sending av data og maks 0.8mA ved datainnsamling. Når man bruker formelen (100 – Q) * B / (100*i) = t, viser det at boksen kan samle inn data konstant i 134t med et batteri på 5.2Ah ved typisk strømtrekk.

Vi kan få til enda lengere batteritid ved å bare samle inn data en gang i timen i tillegg til å ha solpanel på boksen.

[F.2] Testrapport for strømforbruk

[E.2] Testrapport for strømforbruk

2. Test prosedyre

2.1. Fremgangsmåte

Vi utførte testen på en av boksene og ett sett med sensorer

Det ble utførte tre tester.

Test nummer en hvor det ble målt strømforbruk fra boksen uten sensorer koblet på pluggene.

Test nummer to hvor det ble målte strømforbruk fra boksen med alle sensorene koblet til.

Test nummer tre hvor det ble målte strømforbruk fra hver av sensorene som skulle kobles på pluggene.

Testen ble utført med ett UNI-T UT58B Digitalt multimeter

2.2. Testmatrise

Testmatrisene ble satt opp med følgende parameter

- CubeCell i sendemodus
- CubeCell i datainnsamlings modus

Siste matrise hadde også en parameter per sensor som skulle inn i pluggene

- SCD30
- MCP9600/Jordtemperatur
- SEN0308/Jordfuktighet

Test 1.

Modus	**Maks strømforbruk (mA)**	**Kommentar**
Sendemodus		
Datainnsamling		

Test 2.

Modus	**Min strømforbruk (mA)**	**Typ strømforbruk (mA)**	**Maks strømforbruk (mA)**
Sendemodus			
Datainnsamling			

Test 3.

Datainnsamling		
Sensor	**Maks strømforbruk (mA)**	**Kommentar**
SCD30		
MCP9600 / Jordtemp		
SEN0193 / Jordfuktighet		

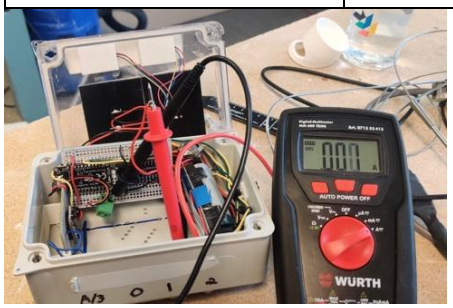
3. Resultater

Test 1.

Test uten noen sensorer koblet til pluggene

[E.2] Testrapport for strømforbruk

Modus	Maks strømforbruk (mA)	Kommentar
Sendemodus	10	
Datainnsamlings modus	16	Varierte fra 10mA til 16mA med ett typisk strømforbruk på 12mA.



Test 2.

Test av strømforbruk med alle sensorene koblet til ble utført tre ganger fordi resultatene varierer.

Runde 1.

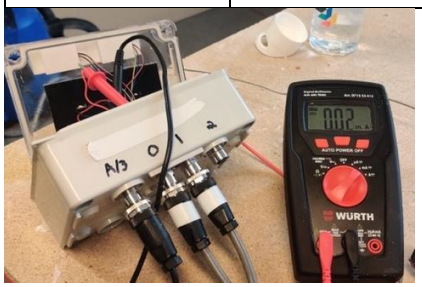
Modus	Min strømforbruk (mA)	Typ strømforbruk (mA)	Maks strømforbruk (mA)
Sendemodus	5	10	35
Datainnsamlings modus	27	30	80

Runde 2.

Modus	Min strømforbruk (mA)	Typ strømforbruk (mA)	Maks strømforbruk (mA)
Sendemodus	3	10	30
Datainnsamlings modus	25	33	88

Runde 3.

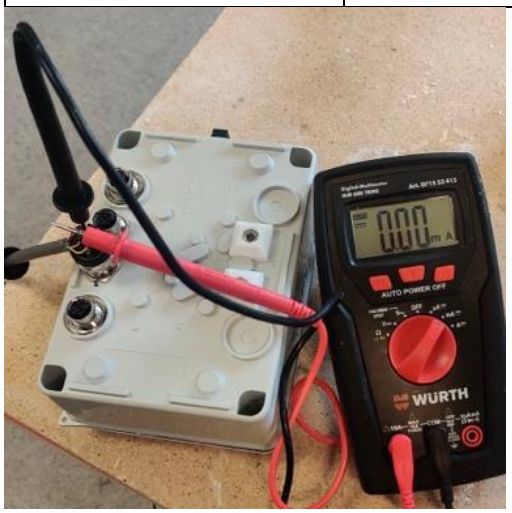
Modus	Min strømforbruk (mA)	Typ strømforbruk (mA)	Maks strømforbruk (mA)
Sendemodus	5	10	11
Datainnsamlings modus	25	30	80



Test 3.
Test av hver enkel sensor koblet til pluggene

[E.2] Testrapport for strømforbruk

Datainnsamling		
Sensor	**Maks strømforbruk (mA)**	**Kommentar**
SCD30	45	
MCP9600 / Jordtemp	6.4	
SEN0193 / Jordfuktighet	3	



4. Konklusjon

[E.2] Testrapport for strømforbruk

Ut ifra testene ser vi at SCD30 sensorenner den som trekker mest strøm noe som ikke er overaskende da den egentlig er tre sensorer i en. Vi kan også se at strømforbruket varier når man har i alle sensorene, men resultatene viser gjennomsnittlig ett strømforbruk på

Sending:

Min: 4mA

Typ: 10mA

Maks 25mA

Data innsamling:

Min: 26mA

Typ: 31mA

Maks 83mA

I beregningene kan man bruke typisk strømtrekk fordi strømtrekket var bare på maks i under maks 2 sekunder.

Ut ifra test to kan man også sjekke hvor lenge batteriet kan holde konstant datainnsamling som vil være en ekstrem case.

Batteriet er på 2600Ah*2 (parallell) = 5200Ah

Ved å bruke formelen:

i = strømtrekk

B = batteri størrelse

Q = strøm som skal være igjen i batteriet i % (burde ikke gå under 20%)

(100 – Q) * B / (100*i) = t

(100 – 20) * 5.2/ 100*0.031 = 134t

Kan også bruke maks trekk

(100 – 20) * 5200 / 100*0.088= 47t

For å utvide denne tiden vil ikke noden samle inn data konstant, men heller samle inn data en gang pr time. Det er og solpanel på boksen som lader opp imellom hver sending.

Vedlegg F

Poster

SOLDREVNE SENSORBOKSER MED TILKOBLING TIL ETT TRÅDLØST LoRaWAN NETTVERK

BAKGRUNN

I fremtiden vil vi ha et behov for høyere energiproduksjon og bedre effektivisering av arealbruk. For at Norge skal kunne bli med på det grønne skifte, er det nødvendig med utbygging av grønn energi som vindkraft og solceller.

I den sammenheng har SINTEF et prosjekt hvor de ser på virkningen av solenergi i landbruket. Er det mulig å kombinere elektrisitetsproduksjon med daglig gårdsdrift?

Som en del av prosjektet har en plassert vertikale solcellepaneler på Skjetlein gård utenfor Trondheim. Forskningen ser på hvilken virkning solcellepanelenes lys/skygge og refleksjoner har på planteveksten.

Dette har allerede blitt simulert digitalt, men for å verifisere resultatene er det bruk for et modulært system for innhenting av sensordata.

THE THINGS STACK

The Things Stack (TTS) er en LoRaWAN nettverks tjener utviklet av The Things Industries.

The Things Stack inneholder ett komplett sett med tjenester for å sette opp ett eget LoRaWAN nettverk.

The Things Network bruker TTS og er den tjenesten vi bruker for å sende data fra nodene til databasen.

THE THINGS STACK | THE THINGS NETWORK

MASKINVARE

Batteriet er på 5.2Ah og lades i parallell av to 0.5W solcellepaneler som befinner seg i lokket av boksen. Dette gir en total strømgenerering på 1W (1).

Det er 4 plugger på sensorboksen. Disse er koblet til en I2C multiplexer, som gjør samtlige plugger modulerbare, med unntak av en plugg som er analog.

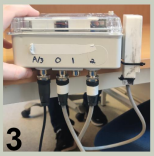
Dette gjør det mulig for brukeren å koble ønsket sensor til hvilken som helst av de tre pluggene, ut fra behov (3).
I tillegg er det to lyssensorer festet til lokket av boksen.

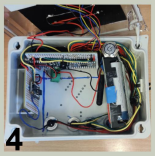

1


2

Sensorene man kan koble til boksen er:
- SCD30 for luftfuktighet, lufttemperatur og CO2(2)
- J-Type Thermoelement for jord temperatur (5)
- SEN0308 for jord fuktighet (6)

Sensorer i lokket til boksen er:
- AS7341 spektrometer (måler fargespekter)(1)
- BH1750 lyssensor (måler lys i lux)(1)


3


4

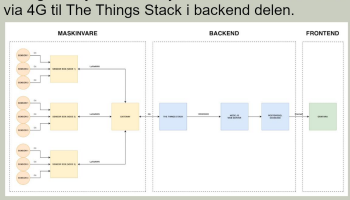

5


6

OPPSETT

Systemet er delt inn i tre deler maskinvare, backend og frontend. Maskinvare delen inneholder alt det fysiske i systemet, sensorene, sensornodene og gatewayen.

Sensorene kommuniserer med sensornodene via i2c og sensornodene bruker LoRaWAN for å kommunisere med gatewayen. Gatewayen sender så dataen videre via 4G til The Things Stack i backend delen.



Backend delen inneholder alle delene som prosserer dataen i bakgrunen før den kommer fram til databasen.

The Things Stack bruker en webhook for å kommunisere med NodeJS webserveren som formaterer dataen og sender den videre til SINTEF sin postgreSQL database.

Deretter kobles Grafana til databasen for å visualisere dataen

VISUALISERING

Som visualiseringsløsning ble det valgt Grafana.

Grafana er en bra løsning fordi det kan kjøres selv på egen tjener eller man kan velge å betale for Grafana Cloud.

Grafana er også en god løsning på grunn av at det støtter tilkobling til mange forskjellige typer databaser inkludert postgreSQL og har bra med tilpasningsmuligheter for dataen

RESULTAT

Prosjektet har resultert i to sensorbokser og to sett med sensorer i tilegg til en grundig dokumentasjon.

Dokumentasjonen tar for seg hvordan man setter opp systemets maskinvare og produksjon av flere bokser og sensorer. Den inneholder også hvordan man setter opp backend med the things stack og webserver i tillegg til en frontend med Grafana.

Dokumentasjonen sammen med de to ferdigproduserte sensorboksene gir et solid grunnlag for SINTEF til å fortsette utviklingen av systemet.

GitBook

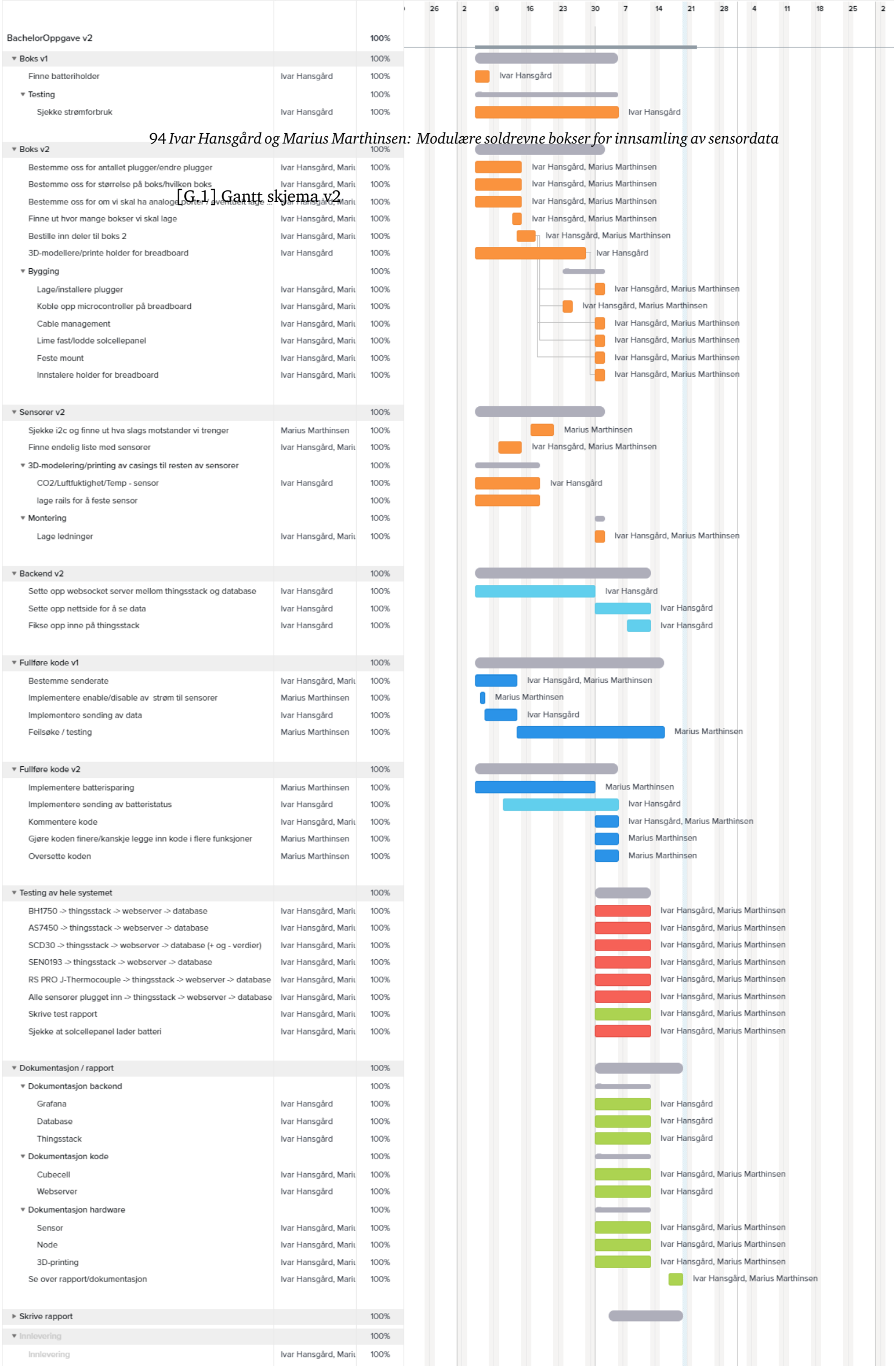
NTNU

SKREVET AV: IVAR HANSGÅRD OG MARIUS MARTHINSEN

SINTEF

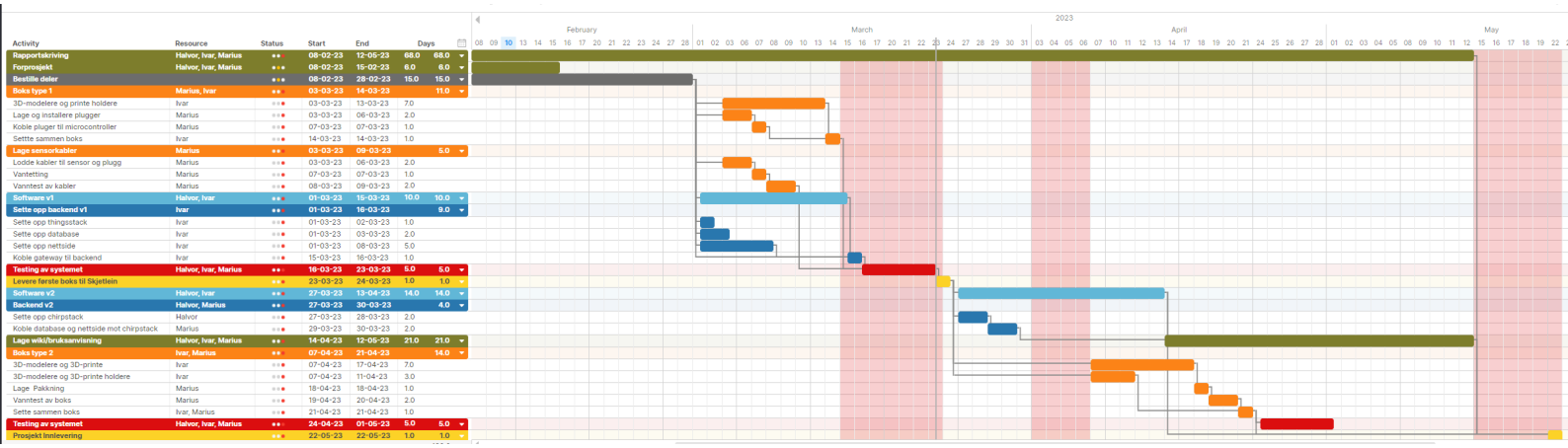
Vedlegg G

Gantt skjema

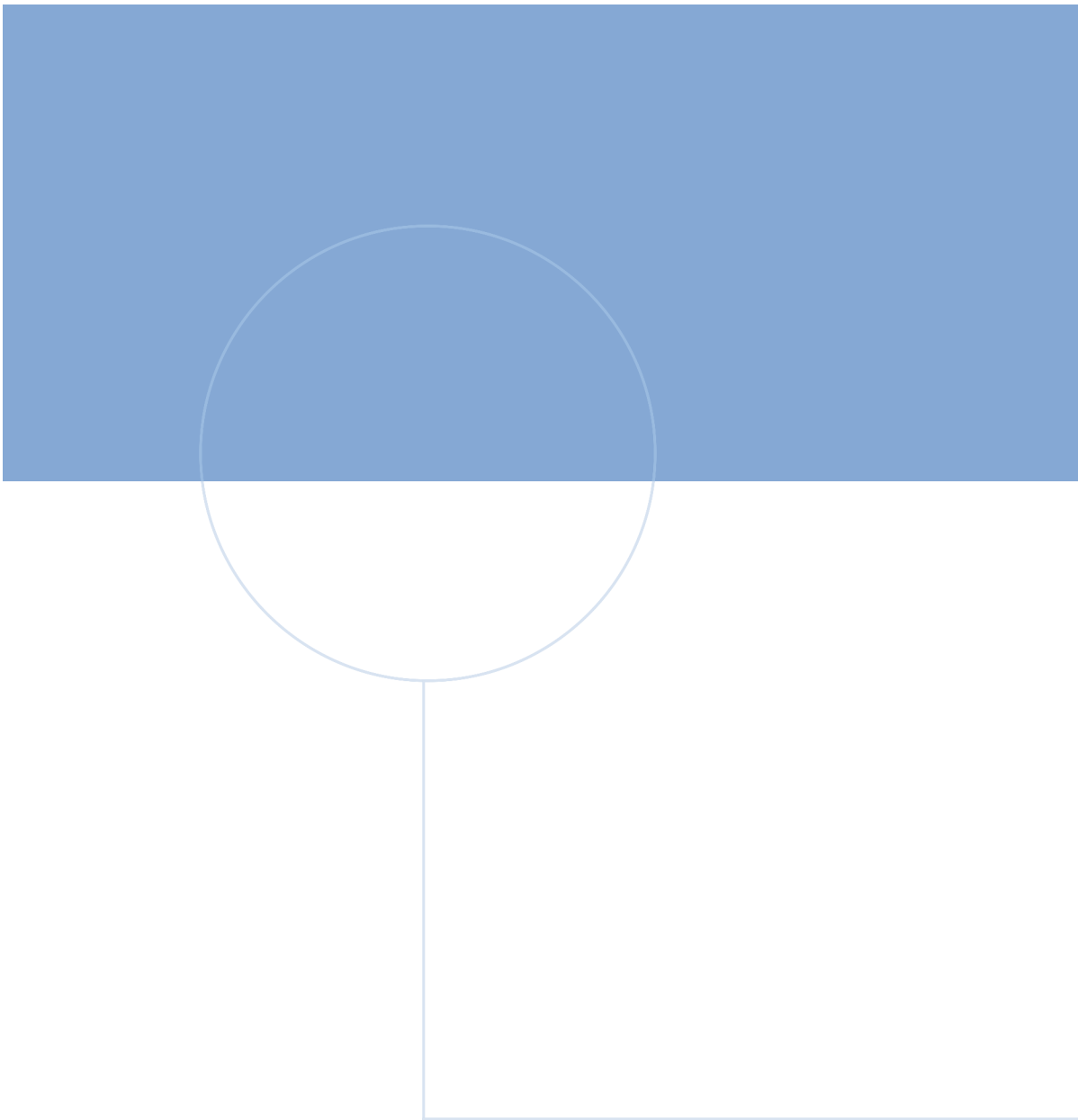


[G.1] Gantt skjema v2

[G.2] Gantt skjema v1



Activity	Resource	Status	Start	End	Days	
Rapportskriving	**Halvor, Ivar, Marius**		**08-02-23**	**12-05-23**	**68.0**	**68.0**
Forprosjekt	**Halvor, Ivar, Marius**		**08-02-23**	**15-02-23**	**6.0**	**6.0**
Bestille deler			**08-02-23**	**28-02-23**	**15.0**	**15.0**
Boks type 1	**Marius, Ivar**		**03-03-23**	**14-03-23**		**11.0**
3D-modelere og printe holdere	Ivar		03-03-23	13-03-23	7.0	
Lage og installere plugger	Marius		03-03-23	06-03-23	2.0	
Koble pluger til microcontroller	Marius		07-03-23	07-03-23	1.0	
Settte sammen boks	Ivar		14-03-23	14-03-23	1.0	
Lage sensorkabler	**Marius**		**03-03-23**	**09-03-23**		**5.0**
Lodde kabler til sensor og plugg	Marius		03-03-23	06-03-23	2.0	
Vantetting	Marius		07-03-23	07-03-23	1.0	
Vanntest av kabler	Marius		08-03-23	09-03-23	2.0	
Software v1	**Halvor, Ivar**		**01-03-23**	**15-03-23**	**10.0**	**10.0**
Sette opp backend v1	**Ivar**		**01-03-23**	**16-03-23**		**9.0**
Sette opp thingsstack	Ivar		01-03-23	02-03-23	1.0	
Sette opp database	Ivar		01-03-23	03-03-23	2.0	
Sette opp nettside	Ivar		01-03-23	08-03-23	5.0	
Koble gateway til backend	Ivar		15-03-23	16-03-23	1.0	
Testing av systemet	**Halvor, Ivar, Marius**		**16-03-23**	**23-03-23**	**5.0**	**5.0**
Levere første boks til Skjetlein			**23-03-23**	**24-03-23**	**1.0**	**1.0**
Software v2	**Halvor, Ivar**		**27-03-23**	**13-04-23**	**14.0**	**14.0**
Backend v2	**Halvor, Marius**		**27-03-23**	**30-03-23**		**4.0**
Sette opp chirpstack	Halvor		27-03-23	28-03-23	2.0	
Koble database og nettside mot chirpstack	Marius		29-03-23	30-03-23	2.0	
Lage wiki/bruksanvisning	**Halvor, Ivar, Marius**		**14-04-23**	**12-05-23**	**21.0**	**21.0**
Boks type 2	**Ivar, Marius**		**07-04-23**	**21-04-23**		**14.0**
3D-modelere og 3D-printe	Ivar		07-04-23	17-04-23	7.0	
3D-modelere og 3D-printe holdere	Ivar		07-04-23	11-04-23	3.0	
Lage Pakkning	Marius		18-04-23	18-04-23	1.0	
Vanntest av boks	Marius		19-04-23	20-04-23	2.0	
Sette sammen boks	Ivar, Marius		21-04-23	21-04-23	1.0	
Testing av systemet	**Halvor, Ivar, Marius**		**24-04-23**	**01-05-23**	**5.0**	**5.0**
Prosjekt innlevering			**22-05-23**	**22-05-23**	**1.0**	**1.0**



NTNU

Kunnskap for en bedre verden