Universitetet i Sørøst-Norge

www.usn.no

PRH612-1 20V Bacheloroppgave

Internet of Things-enhet for akustikkanalyse av pneumatisk transport

IA6-3-20

Fakultet for teknologi, naturvitenskap og maritime fag
Campus Porsgrunn

Universitetet i Sørøst-Norge

www.usn.no

Emne: PRH612-1 20V Bacheloroppgave

Tittel: Internet of Things-enhet for akustikkanalyse av pneumatisk transport

Denne rapporten utgjør en del av vurderingsgrunnlaget i emnet.

Prosjektgruppe: IA6-3-20 **Tilgjengelighet:** Åpen

Gruppedeltakere:
Gaute Eriksen
Henning Engesæt
Jarle Haugland Berge
William French

Veileder: Nils-Olav Skeie

Prosjektpartner: Inovyn Norge AS

Godkjent for arkivering: _____

Sammendrag:

Inovyn er en produsent av PVC-pulver. Dette pulveret transporteres i rør med pneumatisk transport. De har problemer med at sending av pulver stopper opp. Prosjektets hovedmål har vært å lage en enkel og billig prototype som kan detektere tre tilstander i pulversendingen: «Ingen pulver i rør» (av), «pulver i rør» (ok drift) og «kun luft i rør» (ikke ok drift).

Ved hjelp av en SoC og akselerometer måles vibrasjonene i røret. Tilstanden i røret blir avgjort ved å først ta en fouriertransformasjon av målesignalet fra akselerometeret. Dette lager et tilstandsbilde. Deretter blir avviket mellom nåværende tilstandsbilde og referansebilder for «pulver i rør», «luft i rør» og «av» beregnet. Det referansebildet som gir minst avvik til nåværende vibrasjonsbilde gir tilstanden. Signal om tilstand blir gjort tilgjengelig for Inovyn via MQTT over Wi-Fi. Prototypen er i stand til å skille mellom de tre tilstandene under relativt støyfrie forhold.

Gjennom prosjektets utførelse har vibrasjonsbildet i produksjonen vist seg å være mer kompleks enn først antatt. Prototypen tar ikke høyde for varierende vibrasjonsstøy fra omgivelsene. Videre utvikling må utføres for å få en tilstandsdeteksjon som Inovyn kan bruke.

Universitetet i Sørøst-Norge

www.usn.no

Course: PRH612-1 20V Bachelor's thesis

Title: Internet of Things device for acoustic analysis of pneumatic transport

This report forms part of the basis for assessing the student's performance on the course.

Project group: IA6-3-20 **Availability:** Open

Group participants: Gaute Eriksen
Henning Engesæt
Jarle Haugland Berge
William French

Supervisor: Nils-Olav Skeie

Project partner: Inovyn Norge AS

Approved for archiving: _____

Summary:

Inovyn is a manufacturer of PVC powder. This powder is transported in pipes with pneumatic transport. They have problems with the powder transport stopping. The project's main goal has been to create a cheap and simple prototype that can detect three states in the powder transmission: "No powder in pipe" (off), "powder in pipe" (ok operation) and "only air in pipe" (not ok operation).

Using a SoC and an accelerometer, the vibrations in the pipe are measured. The state is determined by first doing a Fourier transform of the signal from the accelerometer. This creates a state image. The difference between the current state image and reference images for "powder in pipe", "air in pipe" and "off" are calculated. The reference image has the least deviation from the current vibration image determines the condition. The condition is made available to Inovyn via MQTT over Wi-Fi. The prototype can distinguish between the three states under relatively noise-free conditions.

Through the project's execution, the vibrations in the production has proved to be more complex than first thought. The prototype does not take into account varying vibration noise from nearby sources. Further development must be carried out in order to provide a state detection that Inovyn can use.

Forord

Dette prosjektet tilhører en studentgruppe som går 6.semester på studieretningen Informatikk og Automatisering ved Universitetet i Sørøst-Norge, campus Porsgrunn.

Prosjektet er i samarbeid med Inovyn som er en produsent av PVC-pulver på Herøya i Porsgrunn. Deres periodiske utfordringer med transport av dette PVC-pulveret er grunnlaget for denne bacheloroppgaven.

I perioden denne oppgaven ble gjennomført brøt Covid-19 ut, og dette har påvirket arbeidet, spesielt med tanke på testing og utbedring av programvare. Det var ikke mulig for gruppen å få tilgang til produksjonslokalet hos Inovyn under testing, og testene ble utført av Bjørn Erik Larsen som rapporterte testresultatene tilbake til studentgruppa.

Takk til Dag Lønnerød (IT-sjef Inovyn) for muligheten til å skrive oppgaven hos Inovyn.

Takk til Rasmus Standal (Team leader Inovyn) og Bjørn Erik Larsen (Application manager Inovyn) for deres orientering og veiledning knyttet til problemstillingen og rutinene ute i felten, samt å gjøre tilgengelig nødvendig utstyr og hjelp under testing og fotografering.

Takk til Maths Halstensen (Førsteamanuensis Institutt for elektro, IT og kybernetikk Campus Porsgrunn) for råd om bruk av sensor, behandling av sensor data og tips om bruk av akselerometer.

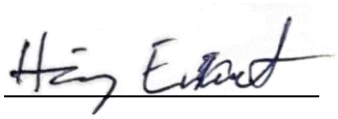
Takk til Frederik Hansen, Overingeniør Institutt for elektro, IT og kybernetikk Campus Porsgrunn for utlån av utstyr brukt i prosjektet.

Takk til Nils-Olav Skeie (Professor ved Institutt for elektro, IT og kybernetikk Campus Porsgrunn) for veiledning og rådføring av gruppa underveis i prosjektet.

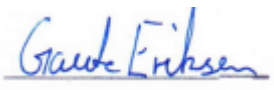
Forventet forkunnskap for lesing av rapport er bachelornivå i informatikk og automatisering, eller lignende.

Forsidebildet er av IoT-enheten montert på et rør i produksjonslokalet. Bildet er tatt av Bjørn Erik Larsen.

Jarle Haugland Berge



Henning Engesæt



Gaute Eriksen

William French

Nomenklaturliste

AMQP	Advanced Message Queuing Protocol
CoAP	Constrained Application Protocol
EEPROM	Electrically erasable programmable read-only memory
EMI	Electromagnetic Interference
FFT	Fast Fourier transform
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
I^2C	Inter-Integrated Circuit
IoT	Internet of Things
IP	Ingress Protection
IT	Informasjonsteknologi
MQTT	Message Queuing Telemetry Transport
OTA	Over-the-air programming
PVC	Polyvinyl chloride
SoC	System on Chip
SSID	Service set identifier
STOMP	Simple Text Orientated Messaging Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
USN	Universitetet i Sørøst-Norge

Innholdsfortegnelse

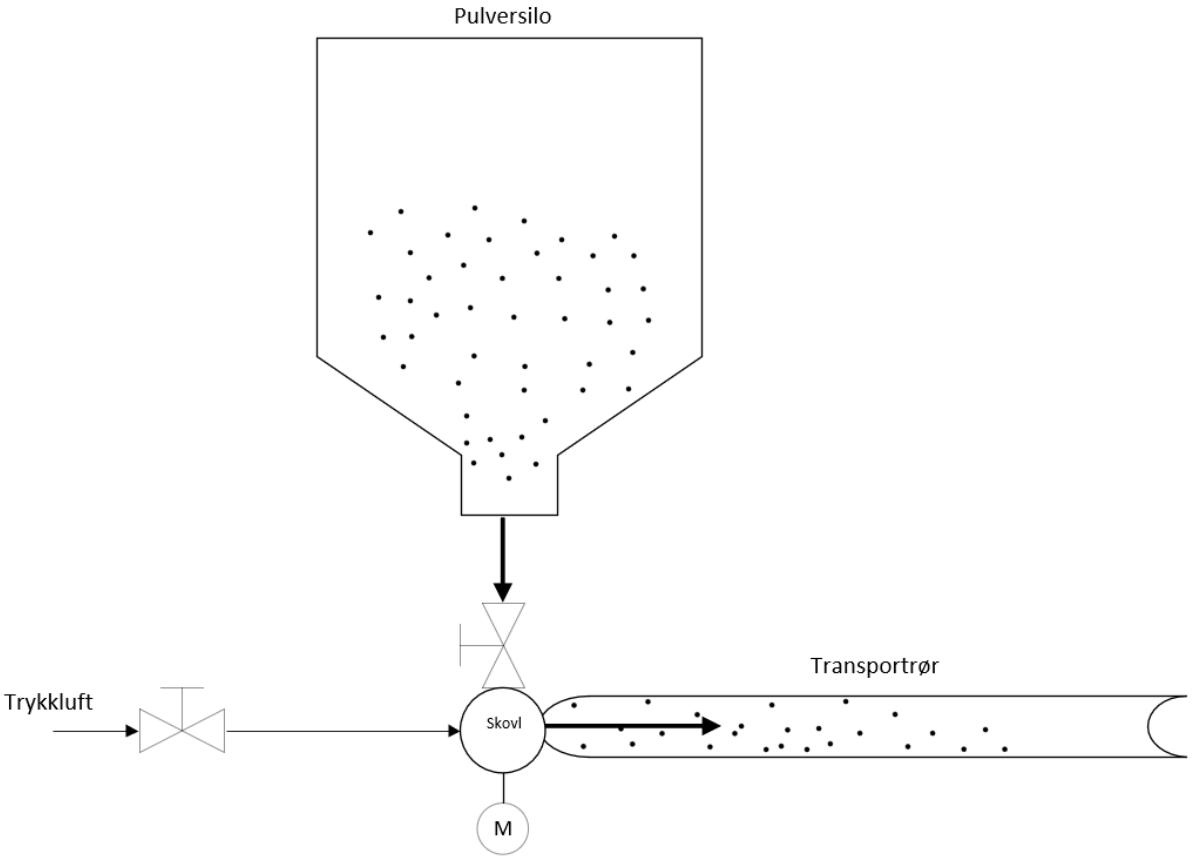
1 Innledning

Denne rapporten omhandler utvikling av en IoT-sensor som måler, analyserer og sammenlikner akustikkdata. Sensoren skal brukes til å detektere blokkering av PVC-pulver i et pneumatisk transportsystem.

Rapporten kan ha interesse for de som ønsker å utvikle sensorer basert på akustiske målinger, eller ønsker å ta i bruk rimelige SoC som er tilgjengelige på markedet.

1.1 Bakgrunn

Under Inovyn sin produksjon av plastpulver på Herøya i Porsgrunn blir pulveret fraktet gjennom fabrikken ved hjelp av pneumatiske transportsystemer. Det ferdigproduserte pulveret samles i pulversiloer før det blir transportert videre. Siloene er utstyrt med en matemekanisme i bunnen bestående av en skovl som drives av en elektrisk motor. Skovlen sørger for at en bestemt mengde pulver blir matet ut fra siloen og inn i et transportrør. På baksiden får skovlen et konstant lufttrykk som blåser pulveret ut i transportrøret. Figur 1-1 visualiserer hvordan pulveret transporteres. At pulver blir blåst gjennom transportrøret kalles pulversending av Inovyn, og blir omtalt slik i rapporten.



Figur 1-1 – Prinsippskisse av pulvertransport fra silo til rør

Sendesystemet kan ha tre tilstander. Tilstandene skal detekteres ved hjelp av et akselerometer som måler vibrasjoner i røret. Se kapittel 2.7 for forklaring av akselerometer. Med vibrasjoner produsert av systemet menes den ristingen som forekommer når skovlen går rundt og luft og pulver beveger seg gjennom røret. Forstyrrende vibrasjoner kan normalt komme fra andre transportlinjer, som er definert utenfor systemet. Dette er de tre tilstandene systemet kan ha:

- Systemet kan være av, hvor det er forventet at sendesystemet selv ikke skal produsere vibrasjoner. I rapporten er denne tilstanden kalt «av».
- Systemet kan være på og i normal sending. Da går skovlen rundt og mater pulver fra siloen inn i røret. Det står da på trykkluft som drar med seg pulveret i røret. I rapporten kalles denne tilstanden «pulver», «pulver i rør» eller lignende.
- Systemet kan også være på, men det blir ikke matet pulver ut i røret. Da går det kun luft i transportrøret. Tilstanden kalles i rapporten «luft i rør», «kun luft» eller lignende.

1.2 Problemstilling

Under drift oppstår det uforutsette tilfeller der matingen stopper opp. Dette resulterer i at det kun blåser luft gjennom transportrøret. En måte å fastslå om matingen har stoppet opp, er å høre på lyden fra transportrøret. Dersom tilstanden i pulversendingen endrer seg, er det tydelige forskjeller i lyd som mennesker kan høre dersom de oppholder seg i nærheten.

1.3 Hypotese

Hypotesen til gruppa er at ved å bruke et akselerometer til å samle vibrasjonsdata fra transportrøret kan man med fouriertransformasjon se forskjellen mellom de ulike tilstandene som skal overvåkes i transportsystemet.

1.4 Målformulering

Prosjektet skal ta for seg ulike metoder for å kunne detektere og analysere informasjon fra akustiske vibrasjonsbilder. Informasjonen skal brukes til å gjenkjenne tre ulike tilstander for transportrøret:

1. Ingenting i røret, transportsystemet er ikke i drift
2. Pulver og luft i røret, normal drift
3. Kun luft i røret, unormal drift

Ulike sensorer, kommunikasjonsprotokoller og maskinvare skal vurderes for å kunne lage en komplett prototype. Programvare skal lages. Prototypen skal ha et IoT-grensesnitt med MQTT inn mot Inovyn sitt produksjonslokale.

Rapporten skal legge til rette for videre arbeid til en eventuell etterkommende fase.

Se vedlegg 2 for prosjektbeskrivelse.

1.5 Avgrensinger

- Varighet til prosjektet er fra 01.01.2020 til 19.05.2020.
- Prototypen som utvikles under prosjektet skal ikke overstige total kostnad på 5 000 NOK.
- Prototypen skal ta seg av behandling av vibrasjonsdata. Med andre ord, kun diskrete signaler om hvilke tilstander prototypen detekterer skal sendes videre til overvåkning hos Inovyn.
- Overvåking og logging av informasjon fra prototypen gjøres av Inovyn.
- Prosjektet omhandler kun vibrasjonsanalyse for det spesifikke transportrøret som prototypen er montert på. Vibrasjon fra andre kilder ligger utenfor prosjektets ramme.

1.6 Leseveiledning

Kapittel 2 inneholder beskrivelser og valg knyttet til; bruk av fouriertransformasjon, kommunikasjonsprotokoll, systemstruktur, utviklingskort, sensor, programkode, testing, innkapsling og festeanordning.

Kapittel 3 inneholder resultater fra testene samt diskusjon av resultatene.

Kapittel 4 inneholder konklusjonen av prosjektet.

2 Metoder og materialer

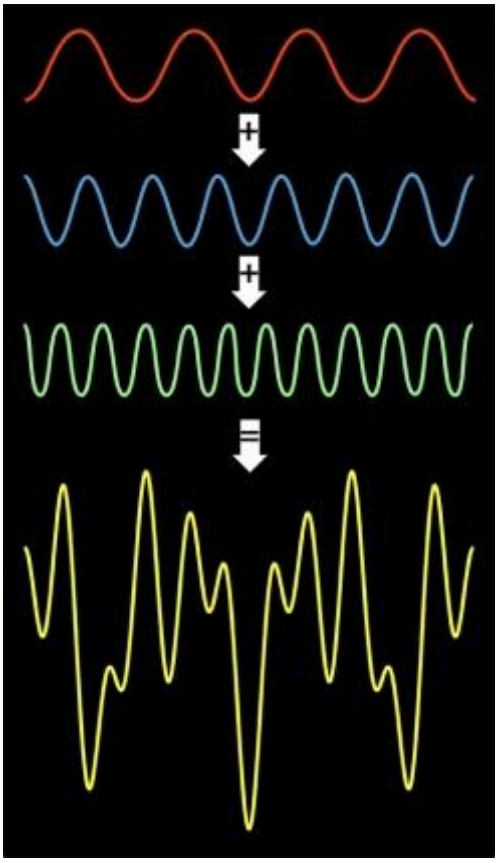
I dette kapittelet er designvalg, sentrale prinsipper i oppgaven, og funksjonalitet i programmet omtalt. I tillegg står det om planlegging og utførelse av test.

2.1 Alternative metoder for å løse problemet

Det er vurdert to prinsipper for å analysere signalet gitt fra sensoren. Det første og enkleste er å se på amplituden til målesignalet for å bestemme tilstand på pulversendingen alt etter hvor stor amplituden er. Den andre metoden er å gjøre en fouriertransformasjon på et sett av måleverdier, og bestemme tilstand på pulversendingen etter å ha sammenlignet nåværende frekvensbilde med tre referansebilder. Det ble valgt å bruke fouriertransformasjon. Dette er begrunnet i kapittel 2.3.

2.2 Hva er fouriertransformasjon?

Fouriertransformasjon baserer seg på prinsippet om at all lyd og vibrasjon kan brytes ned til en sammensetning av rene sinuskurver med forskjellige frekvenser og amplituder [1]. Dette er illustrert i Figur 2-1.

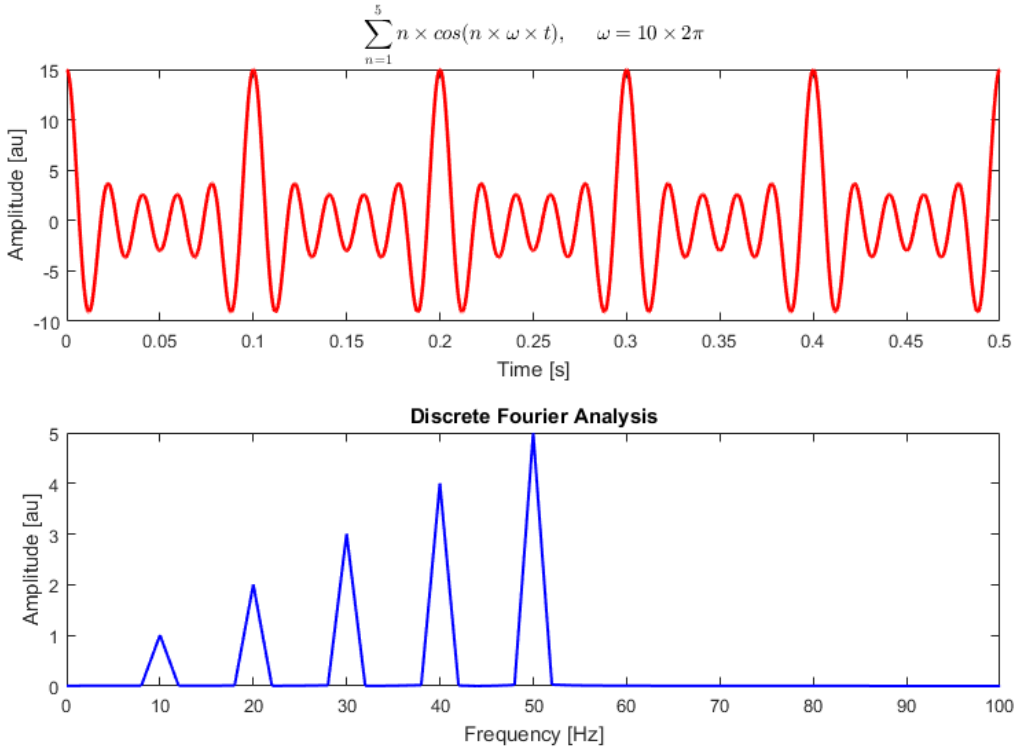


Figur 2-1 - Illustrasjon som viser at en sum av sinuskurver utgjør en kompleks lydkurve [2].

I Figur 2-1 er alle amplitudene like, mens frekvensene er forskjellig. Det gjør at noen steder forsterker signalene hverandre, mens andre steder opphever de hverandre [3]: Hadde amplitudene også vært forskjellige ville den, eller de frekvensene med høyest amplitude, hatt mer å si for hvordan det endelige signalet ser ut. Dette er fordi en høyere amplitude vil dominere hvordan det endelige signalet vil se ut.

Det man ønsker å oppnå med fouriertransformasjon er å gå motsatt vei [4]: For et sammensatt lyd- eller vibrasjonssignal, vil transformasjonen gi informasjon om hvilke frekvenskomponenter som har hvilken styrke i signalet.

Hvis man, som i Figur 2-2, legger rådata og fouriertransformert data i hver sin graf, vil man for rådataen ha amplitude på y-aksen og tid på x-aksen. For den transformerte dataen får man fortsatt amplitude på y-aksen, men frekvens på x-aksen. Det betyr i denne figuren at det opprinnelige signalet består av: en sinuskurve med amplitude 5 og frekvens 50 Hz, en sinuskurve med amplitude 4 og frekvens 40 Hz, en sinuskurve med amplitude 3 og frekvens 30 Hz, osv.

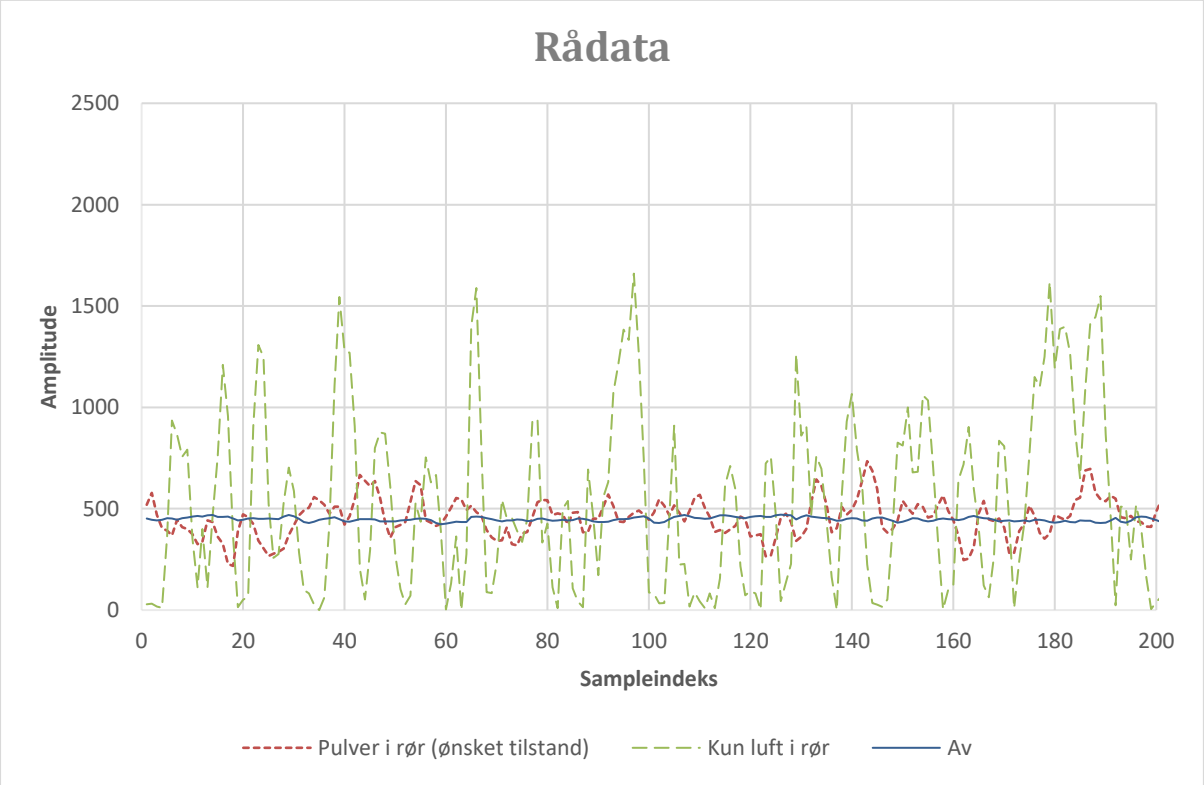


Figur 2-2 - Illustrasjon av diskret fouriertransformasjon, hvor punkter gitt av funksjon i toppen av figuren er brukt [5].

At en fouriertransformasjon er diskret betyr at den er utført med datapunkter istedenfor en funksjon [6]. Det finnes flere ulike metoder, med varierende grad av effektivitet og nøyaktighet, for å utføre diskret fouriertransformasjon. Det er oftest mest hensiktsmessig å utføre diskret fouriertransformering med en datamaskin. Metoden brukt i dette prosjektet er Fast Fourier Transform (FFT), som er en rask metode for å utføre diskret fouriertransformasjon i en datamaskin [7].

2.3 Hvorfor bruke fouriertransformasjon?

Måledata innhentet fra transportrøret hos Inovyn viser at forskjellen i amplitude på signalet fra akselerometeret er stor nok til at de forskjellige tilstandene i pulversendingen kan skilles fra hverandre ved å kun se på amplitude og spredningen av denne. Dette er vist i Figur 2-3.



Figur 2-3 - Utklipp av samplede data fra én akse fra akselerometer ved de tre forskjellige tilstandene i transportrør på silo L-3. Tilstandene er forklart i kapittel 1.1 Innhentet av studentene i prosjektet.

I produksjonslokalet er noen transportrør frittstående, mens noen er festet i samme konstruksjon, som illustrert i Figur 2-4.

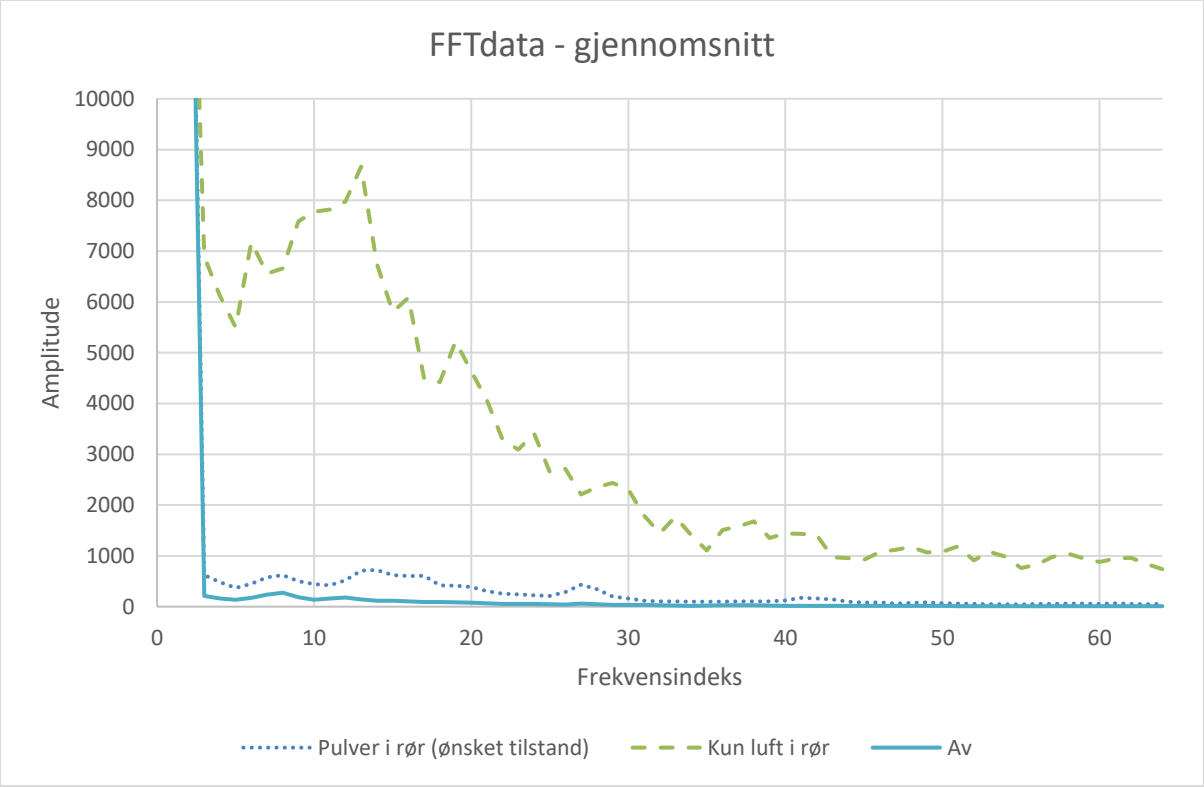
Figur 2-4 - Bilde av rørinnfestning

Der hvor rørene er festet i samme konstruksjon, som L2, L3 og L6, vil vibrasjoner fra rør i drift påvirke de andre rørene. I dette prosjektet er det ikke gjort målinger på rør som påvirker hverandre. Det antas at det blir vanskeligere å skille hvilket rør som rister ved kun måling og analysering av amplitude på vibreringen. Det er mulig at en fouriertransformasjon av signalet kan

gi data som bedre kan brukes til å skille ulike sendingstilstander i de forskjellige transportrørene. Fouriertransformasjon og analyse av ikke-sykliske signaler ble utført i «Sound Analysis to Recognize Different Animals» [8], som kan være en lignende situasjon når det gjelder kompleksitet i lyd-/vibrasjonsbildet. Ikke-syklisk er ment at signalet ikke har et repeterende mønster, slik som i for eksempel roterende utsyr. I tillegg var det et ønske fra Inovyn om å bruke fouriertransformasjon.

Som vist i Figur 2-5 er det klare forskjeller i frekvensbildet til vibrasjonene i transportrøret ved tilstand «pulver i rør» og «kun luft i rør», som er tilstandene det vil veksle mellom under drift. Det er merkbart mindre forskjell mellom tilstandene «pulver i rør» og «av». Dette er diskutert i kapittel 3.4.3.

I dette prosjektet blir det bare målt på rør som i liten grad eller ikke blir påvirket av vibrasjoner på andre rør i produksjonslokalet. Dette er videre diskutert i kapittel 3.4.2.

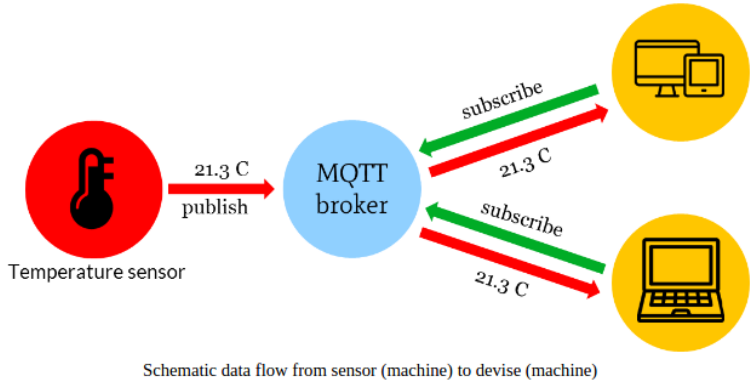


Figur 2-5 - Gjennomsnitt av fouriertransformert data fra test. Merk at på x-aksen er det indeks, ikke frekvens. Høy indeks er høy frekvens.

2.4 Valg av kommunikasjonsprotokoll

For at prototypen skal kunne kommunisere mot et overordnet system, må det velges en protokoll for dette. Alternativer til IoT-grensesnitt vurderes mot MQTT-protokollen, som er foreslått av Inovyn. Av protokoller vil AMQP, STOMP og CoAP sammenlignes.

MQTT er en åpen standard laget av Dr. Andy Stanford-Clark fra IBM, og Arlen Nipper fra Arcom i 1999 [9]: Den har fokus på å være lettvekt og enkel, for å kunne brukes på enheter med ulike begrensninger. Enheter vil kunne koble seg til for å kommunisere via en server, også kalt MQTT-broker, som vist i strukturen i Figur 2-6. Protokollen bruker publisering og abonnering. Informasjon er gitt ved tekststrenger. Disse tekststrengene sorteres i en trestruktur, med sorteringsord kalt topics. Brokeren tar imot meldinger, og sender de videre til enheter som abonnerer på topicet meldingen er sendt under.



Figur 2-6 – Eksempel på MQTT-struktur [10].

AMQP er også en åpen standard [11]: Den har fokus på bruk i business og organisasjoner og sammenkobling av disse. Den har mulighet for kryptering av meldinger og gir en robust kommunikasjon mellom enheter. Krypteringen gjør imidlertid også standarden tyngre å prosessere for en mikrokontroller og er da i mindre grad brukt av utviklingskortsamfunnet.

STOMP er en enklere protokoll basert på HTTP [12]: Som med MQTT er den bygget for å være lettvekt, i motsetning til f.eks. AMQP, og enkel å implementere. Den er mest brukt i sammenheng med nettsider. Biblioteker er tilgjengelig for utviklingskort som Arduino og lignende.

CoAP er en spesialisert protokoll for IoT-bruk [13]: Som med STOMP er denne også basert på HTTP. Den bruker kommunikasjonsformen klient og server, på samme måte som webtjenester.

I Tabell 2-1 er hovedtrekkene ved protokollene oppsummert. Ved utgangspunkt i tilgjengelighet, kapabilitet, ferdigimplementering hos Inovyn og derav oppgavebeskrivelse, ble det valgt å bruke MQTT som kommunikasjonsprotokoll.

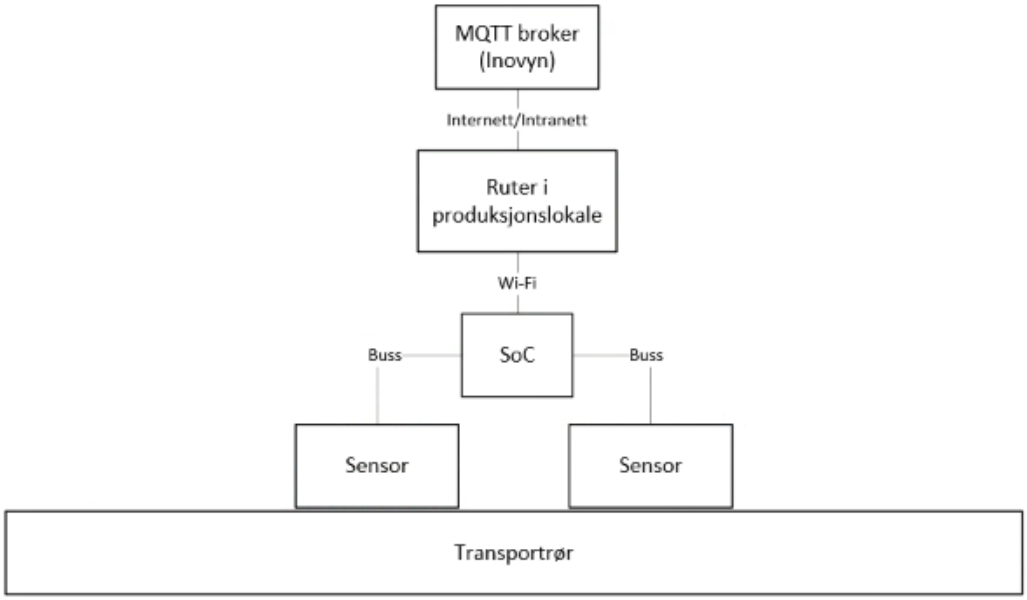
Tabell 2-1 - Sammenligning av kommunikasjonsprotokoller

Spesifikasjoner / Protokoll	Meldingsmodell	Standard	Adressering	Bibliotek for Arduino/ utviklingskort
MQTT	Publish/Subscribe	Ja (ISO/IEC 20922)	Ja (Topic)	Ja
AMQP	Publish/Subscribe	Ja (ISO/IEC 19464)	Ja	Nei
STOMP	Publish/Subscribe	Nei	Nei	Ja
CoAP	Request-Response, Publish-Subscribe	Nei	Ja (URL)	Ja

2.5 Valg av systemstruktur

Det har blitt vurdert tre systemstrukturer, og en blanding av to av disse ble valgt. Dette er beskrevet nedenfor. Valg av sensor er forklart i kapittel 2.7. SoC (System on Chip) er beskrevet i kapittel 2.6. MQTT broker er beskrevet i kapittel 2.4.

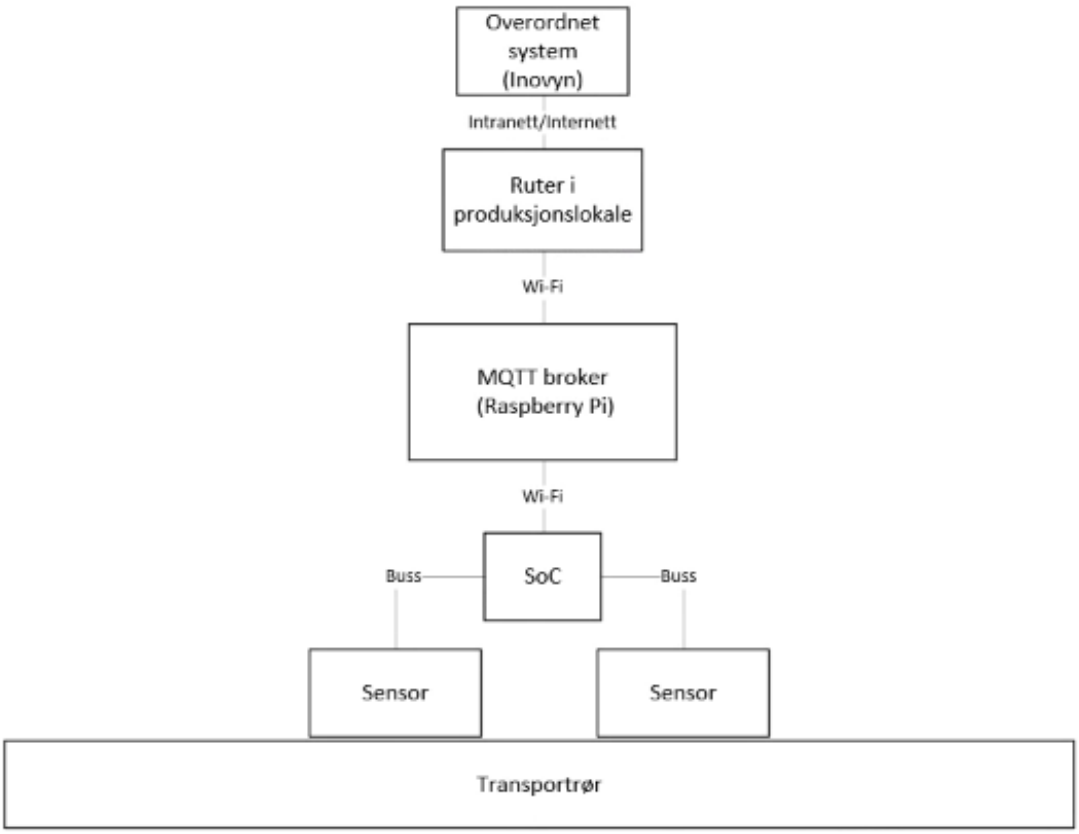
1. Første systemstruktur er vist i Figur 2-7.



Figur 2-7 - Én SoC med flere sensorer, hvor SoCen utfører all databehandling og kommunikasjon med overordnet system med MQTT.

I alternativet i Figur 2-7 ble det vurdert om flere sensorer kunne øke nøyaktigheten i avgjørelsene systemet tar, ved å ha mer data å støtte seg på. Det har ved testing vist seg at én sensor antakelig gir godt nok datagrunnlag i dette prosjektet, vist i kapittel 2.3 med Figur 2-5.

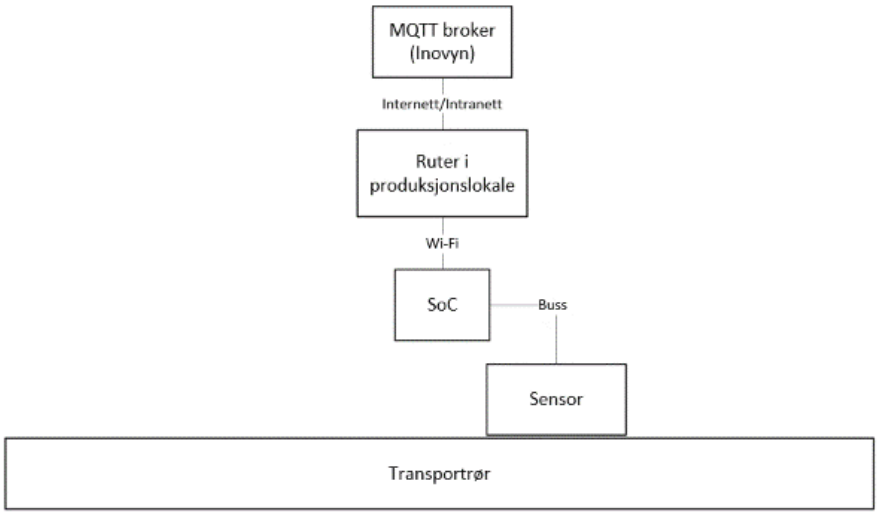
2. Andre systemstruktur er vist i Figur 2-8.



Figur 2-8 - Én SoC med en eller flere sensorer. SoCen utfører all databehandling og kommuniserer med en enhet som ikke er del av overordnet system, f.eks. en Raspberry Pi. Raspberry Pien utveksler informasjon med overordnet system med MQTT.

Alternativet i Figur 2-8 er vurdert i hovedsak med tanke på videre utviding. På et senere tidspunkt kan det være aktuelt å ha flere enheter som overvåker flere punkter. Da kan systemet deles opp slik at en gruppe SoCer med sensorer kommuniserer med en Raspberry Pi, som igjen kommuniserer med en MQTT broker gjennom en ruter.

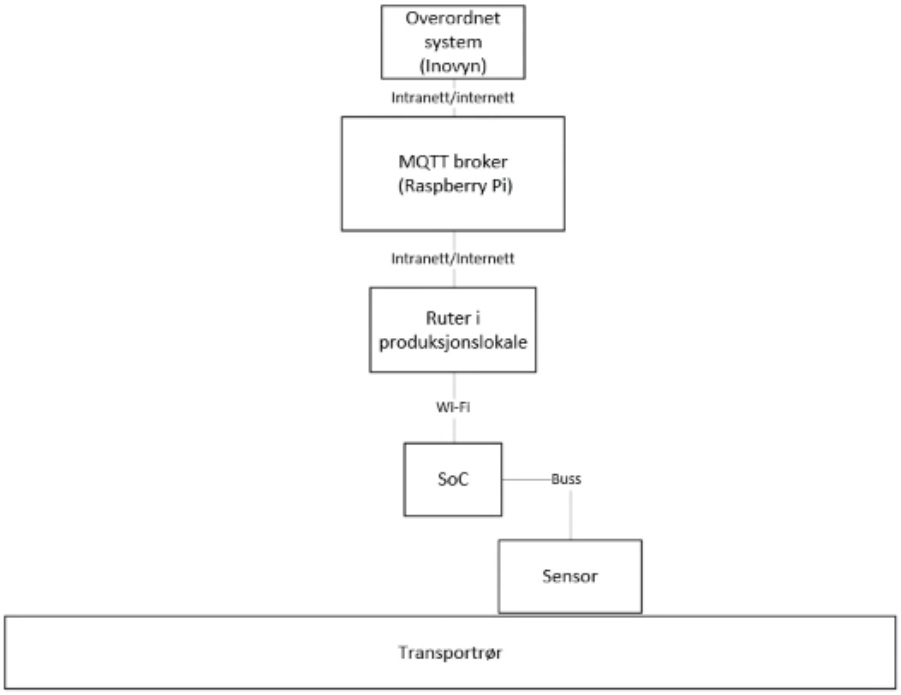
3. Tredje systemstruktur er vist i Figur 2-9.



Figur 2-9 - Én SoC med én sensor, hvor SoCen utfører all databehandling og kommunikasjon med overordnet system med MQTT.

Det var ønskelig å velge strukturen vist i Figur 2-9, som er det enkleste alternativet. Tester utført i produksjonslokalet viser at ett akselerometer med én akse antakelig gir nok informasjon til å skille tilstandene fra hverandre, vist i Figur 2-5. Valg av sensor er forklart i kapittel 2.7.

Grunnet Covid-19 ble endelig systemstruktur for dette prosjektet en blanding mellom struktur 2 og 3, vist i Figur 2-10. På denne måten kunne gruppa ha kontroll på MQTT-broker uten å trenge tilgang til datasystemene til Inovyn. Dette grunnet datasikkerhet.



Figur 2-10 - Blandingsstrukturen; én SoC med ett akselerometer, hvor SoCen utfører all databehandling. SoCen kommuniserer med en Raspberry Pi som ikke er del av overordnet system, som igjen utveksler informasjon med overordnet system med MQTT.

Databehandlingen blir gjort på SoCen. SoCen kommuniserer med MQTT-brokeren i Raspberry Pien. Den står på et eksternt nettverk. Systemstrukturen er vist i Figur 2-10 og koblingen mellom SoC og akselerometer er vist i Figur 2-13, som totalt sett gir signalgang fra sensor til mottaker hos Inovyn. Se Figur 2-22 for helhetlig illustrasjon av dataflyt i prototypesystemet.

2.6 Valg av utviklingskort

For valg av utviklingskort (kjent som SoC) ble det fokusert på: biblioteker for FFT, prosessorkraft, tilkobling for mikrofon/akselerometer og trådløs kommunikasjon/MQTT. En SoC er hele kortet og mikrokontrolleren er prosessoren på kortet.

Tidligere innføring i Arduino fra universitetet gav et naturlig utgangspunkt for søk etter kontroller. Etter noe undersøkelse ble det funnet eksempler på bruk av en alternativ kontroller, ESP8266, til å gjøre signalanalyser. En av disse er en artikkel om overvåking av helsen til en kompressor [14]: I denne artikkelen blir det brukt en ESP8266, akselerometeret GY-521 MPU-6050 [15] og fouriertransformering.

I tillegg til dette var det allerede en ESP8266 tilgjengelig innad i gruppen.

Det eksisterer en etterfølger av ESP8266, med navn ESP32, som er raskere og har flere funksjoner, blant annet Bluetooth. ESP32 er en chip som Inovyn har erfaring med, og som de bruker i samme system som prototypen skal kommunisere mot.

En kortfattet sammenligning av kontrollere er gjort under i Tabell 2-2.

Tabell 2-2 - Sammenligning av kontrollere/SoC [16] [17] [18].

Spesifikasjoner / SoC	Mikrokontroller	Maks Klokkefrekvens	Wi-Fi	Bluetooth	FFT-bibliotek	Ca. Pris (NOK)
Arduino Uno	ATmega328P	16 MHz	Nei	Nei	Ja	220,-
Arduino Uno m/ WiFi	ATmega328P	16 MHz	Ja	Nei	Ja	460,-
WeMos D1 Mini	ESP8266	160 MHz	Ja	Nei	Ja	60,-
Adafruit HUZZAH32	ESP32	240 MHz (dobbeltkjerne)	Ja	Ja	Ja	200,-

Kontrollere som bruker ESP8266 og ESP32 er gunstige valg da de er raskere og rimeligere enn Arduino Uno. Ved utgangspunkt i tilgjengelighet og god nok kapabilitet, blir det valgt å bruke en D1 Mini (ESP8266) som utviklingskort til prototypen.

2.7 Valg av sensor

Valg av sensor er et viktig element i løsningen til problemstillingen. I utgangspunktet er det forskjellen i lyd, som for prosessoperatørene er den største indikasjonen på om det går pulver i røret eller ikke. Mikrofoner ble derfor først vurdert som sensorer til prototypen før overgang til akselerometer.

2.7.1 Sensortyper

Mikrofoner vil fange opp enhver lyd som er i lokalet, hvorav det er flere. Vibrasjonene, eller lydbølgene i luften, vil være vanskeligere å garantere at kommer fra riktig prosessdel. Bruk av akselerometer vil i motsetning gi en mer direkte avlesning på det respektive røret som overvåkes. Vibrasjoner blir da fanget opp direkte fra prosessdelen som overvåkes. Akselerometer vil også være mottagelig for støy. Dette via vibrasjoner fra andre komponenter som forplanter seg gjennom produksjonslokalet.

2.7.2 Sensorgrensesnitt

For å kunne gjøre en god analyse av lydbildet som dannes av vibrasjonene, er samplingshastigheten på sensor viktig.

Som nevnt i kapittel 1.2, vil operatøren kunne høre tilstanden til transportrøret. Måleomfanget til sensoren må da være i stand til å fange de relevante frekvensene som videre kan brukes til å angi riktig tilstand på transportrøret. Studier viser at hørsel til en frisk ung person ligger i frekvensområdet 20 til 20 000 Hz [19].

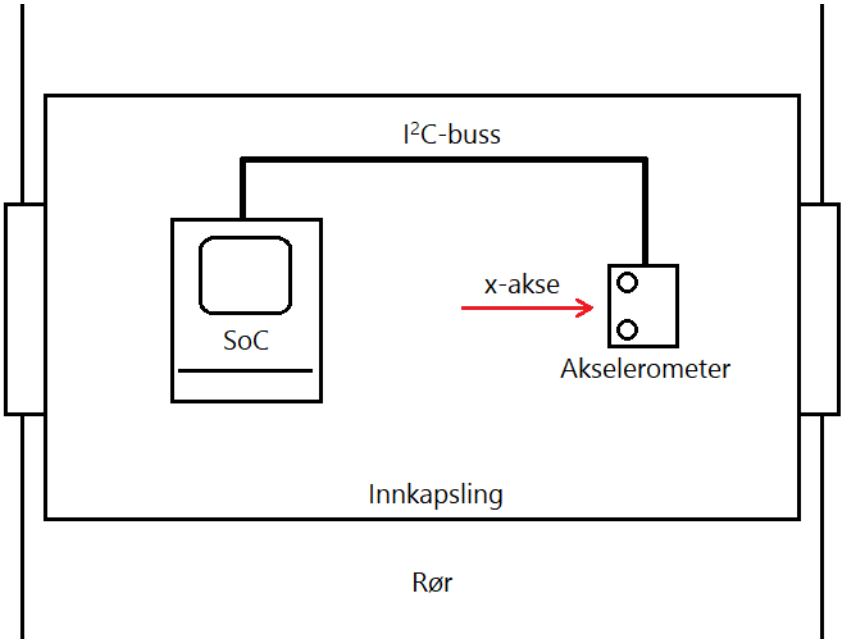
Digitale signaler mellom sensor og kontroller er hensiktsmessig å sende over en kommunikasjonsbuss. Valgt kontroller, D1 Mini (ESP8266), støtter både I^2C- og SPI-buss [17]. Valg av sensor ble derfor gjort til ett av disse grensesnittene.

2.7.3 Valgt sensor

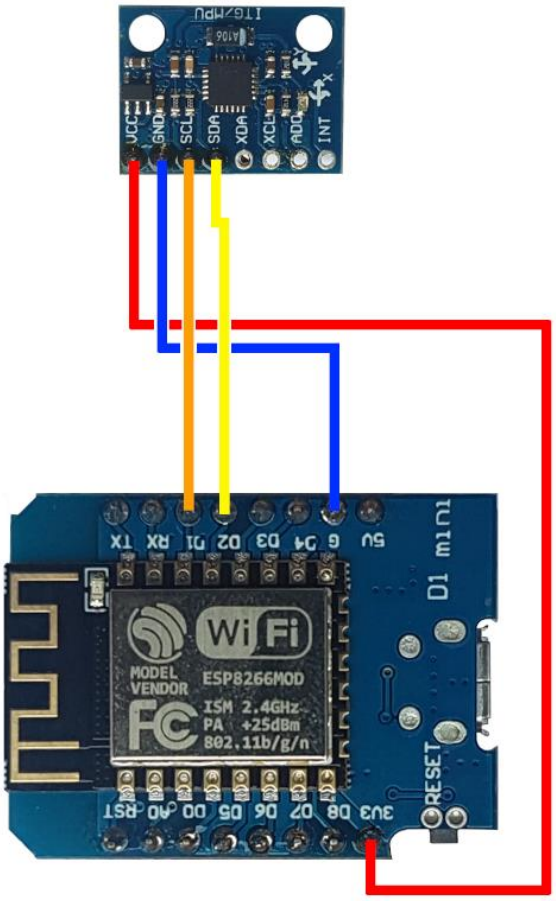
På samme grunnlag som valg av kontroller ble GY-521 breakout board (MPU-6050) valgt som sensor [15]. Den var tilgjengelig fra universitetet og kunne tas i bruk umiddelbart. Figur 2-11 viser sensoren.

Sensoren har både akselerometer og gyrometer innebygget i seg. Akselerometeret måler bevegelse i lengderetning av aksen og gyrometeret måler rotasjonsbevegelse om aksene. Hver av funksjonene fungerer på både x-, y- og z-aksen. Det er kun tatt i bruk akselerometer i x-akse på sensoren på tvers av røret, som vist i Figur 2-12. Sensoren har analog til digital konvertering og bruker I^2C-buss til å sende informasjon [20]. Bussen er støttet av valgt kontroller.

Figur 2-11 - Akselerometeret GY-521 MPU-6050 brukt i prototypen [15].



Figur 2-12 – Skisse på akseretning til akselerometer i forhold til rør



Figur 2-13 - Kobling av akselerometer til SoC.

2.8 Programkode

Bruken av valgt sensor, utviklingskort, systemdesign og fouriertransformasjon knyttes i koden på kontrolleren. I dette underkapittelet beskrives det hvordan programmet fungerer, uten å gå for dypt i kode eller syntaks. Programkoden finnes i vedlegg 3 og en illustrasjon av programflyt i vedlegg 4. Informasjon om biblioteker ligger i kapittel 2.8.4.

2.8.1 Oppstartfunksjoner i program

Funksjon for bruk av internminne blir kalt opp og referanseverdier blir hentet fra minnet. Dette blir gjort via EEPROM og bibliotek for dette. Tilknytning til Wi-Fi vil bli forsøkt frem til tilkobling er opprettet. SSID til ruter samt passord er lagt direkte i programmet via variabler. Adresse for MQTT-server blir også etablert her. Etter dette starter kommunikasjon mot akselerometeret over I^2C-buss. Underveis vil kontrolleren også kontrollere om kode skal lastes via OTA.

2.8.2 Teach-funksjon og MQTT-kommandoer

Kontrolleren vil kunne motta kommandoer for parametrisering via MQTT. Den abonnerer på fire topics; «FFT/ teach», «FFT/ axis», «FFT/restart» og «FFT/avgSamples». Topics er forklart i kapittel 2.4. Det er totalt ti meldinger som vil kunne utløse interne ferdiglagde funksjoner, der tre av dem kan lagres permanent. Meldingene er fordelt på fire topics og er forklart i Tabell 2-3 under:

Tabell 2-3 - Oversikt over MQTT-kommandoer kontroller kan motta fra bruker

Spesifikasjon / Kommando	Topic	Forklaring	Kan lagres permanent?
“powder”	“FFT/teach”	Setter nåværende vibrasjonsbilde som referansebilde for pulver i rør	Ja
“air”	“FFT/teach”	Setter nåværende vibrasjonsbilde som referansebilde for luft i rør	Ja
“off”	“FFT/teach”	Setter nåværende vibrasjonsbilde som referansebilde for ingenting i rør	Ja
“save”	“FFT/teach”	Lagrer referansebildene til minnet og overskriver tidligere bilder	Nei
“load”	“FFT/teach”	Henter og setter referansebilde fra minnet	Nei
“x”	“FFT/axis”	Setter x-akse som valgt akse fra sensor	Nei
“y”	“FFT/axis”	Setter y-akse som valgt akse fra sensor	Nei
“z”	“FFT/axis”	Setter z-akse som valgt akse fra sensor	Nei
“restart”	“FFT/restart”	Starter kontrolleren på nytt	Nei
Heltall	“FFT/avgSamples”	Setter heltallet som nytt valg av antall bilder for gjennomsnittbilde. Gjennomsnittbilde er forklart i kapittel 2.8.3.	Nei

Etter hver kjøring av programmet, ca. hvert 500. ms, vil programmet publisere status til server. Statusmelding blir sendt til topic «FFT/state» med tekst «powder, «air» eller «off» som vist i Tabell 2-4. Melding blir sendt uavhengig om referansebilder er laget eller ikke.

Tabell 2-4 - Oversikt over MQTT-kommandoer kontroller kan sende til bruker

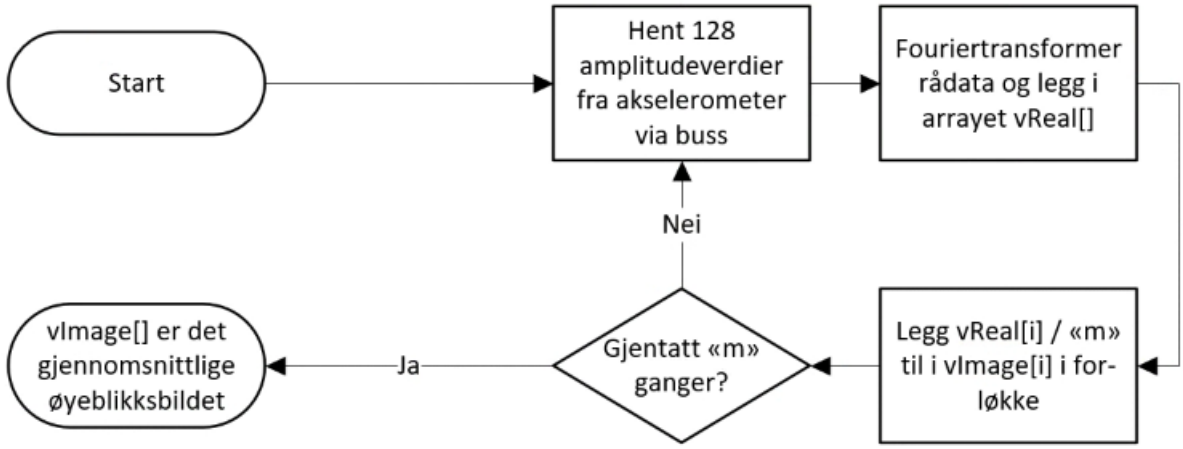
Melding	Topic	Tilstand
“powder”	“FFT/state”	Pulver i rør
“air”	“FFT/state”	Luft i rør
“off”	“FFT/state”	Av

2.8.3 FFT-analyse

Hvordan analysen utføres i programkoden er å finne etter linje 134 i vedlegg 3. Forenklet illustrasjon av analysen kan sees til venstre i vedlegg 4.

Hvert frekvensbilde tatt i samme pulversendingstilstand er ikke nødvendigvis helt lik, og derfor er referansebildene og øyeblikksbilde et gjennomsnitt av et antall frekvensbilder. Gjennomsnittet blir laget som vist i formel 2.1 og illustrert i Figur 2-14, og lagret i arrayet «vImage[]». I formel 2.1 er variabelen «n» størrelsen på arrayene, og variabelen «m» er antallet øyeblikksbilder som skal brukes til å lage gjennomsnittsbildet. «vReal[]» inneholder måleverdier.

$$vImage[i] = \sum_{i=0}^{n} \frac{vReal\_1[i]}{m} + \frac{vReal\_2[i]}{m} + \frac{vReal\_3[i]}{m} + \cdots + \frac{vReal\_m[i]}{m} \qquad (2.1)$$



Figur 2-14 Flytskjema som viser prinsipielt hvordan det gjennomsnittlige øyeblikksbildet lages i hver programsyklus. «m» er antallet øyeblikksbilder som blir brukt for å lage det gjennomsnittlige øyeblikksbildet. Formel 2.1 viser matematisk hva vImage[] består av.

Etter at FFT-analysen har blitt utført, er «vImage[]» et vibrasjonsbilde. Dette er et array med amplituder, hvor lav indeks er amplitude for lav frekvens og høyere indeks er amplitude for høyere frekvens. Øyeblikksbilde «vImage[]» blir tømt og fylt med nye verdier for hver programiterasjon.

Referansene er øyeblikksbilder, det vil si array som er lagret når tilstanden i pulversendingen stabilt er en av de tre som skal detekteres.

Ved vanlig programkjøring, det vil si når det ikke er mottatt meldinger via MQTT, vil det etter at «vImage[]» er fylt, bli beregnet avvik mellom nåværende lydbilde og de tre referansebildene. Absoluttverdien av avviket blir beregnet, indeks for indeks, og deretter summert, som i formel 2.2. Dette gjøres mot hver av de tre referansene. Referansen som gir minst avvik sammenlagt blir valgt som status til pulversendingssystemet.

$$deviation = \sum_{i=0}^{n} |vImage[i] - vRef[i]| \tag{2.2}$$

Melding med status blir så publisert under MQTT-topic «FFT/state», i henhold til Tabell 2-4.

2.8.4 Utviklingsverktøy og feilsøking

Programkoden er utviklet ved hjelp av Arduino IDE som utviklingsverktøy. I tillegg trengs biblioteker for å støtte ESP8266, FFT og MQTT. For installering av biblioteker vises det til Arduino sine sider for FFT og MQTT [21], og GitHub for ESP8266 [22]. Under ligger liste over brukte programmer, biblioteker og deres versjoner:

- Arduino IDE for Windows (ver. 1.8.10, utviklingsprogram)
- esp8266 av ESP8266 Community (ver. 2.6.3, bibliotek)
- arduinoFFT av Enrique Condes (ver. 1.5.5, bibliotek)
- PubSubClient av Nick O'Leary (ver. 2.7.0, bibliotek)

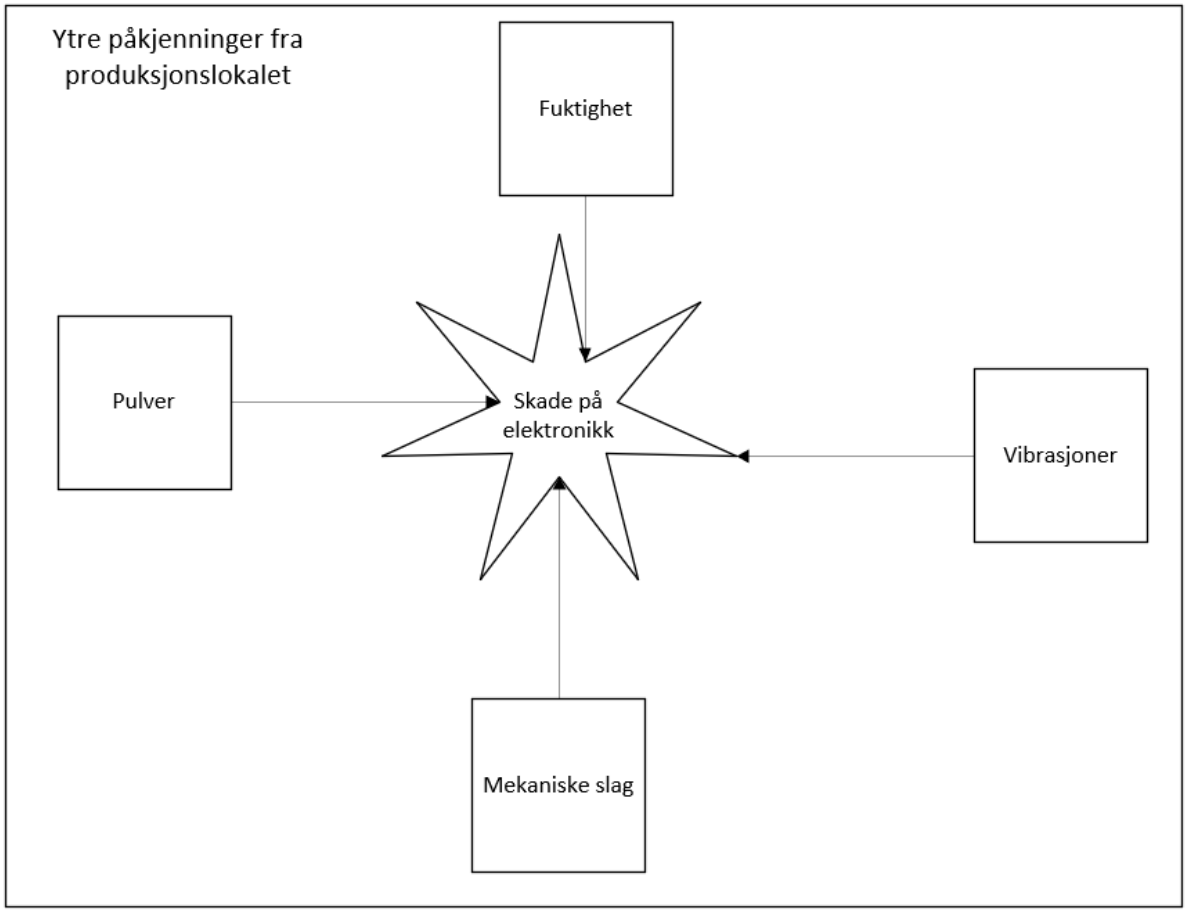
For feilsøking vil kontrolleren starte kommunikasjon over USB på 115200 baudrate når den kobles til en PC. Dette vil i all hovedsak bli brukt til å lese ut programstatus serielt og er ikke nødvendig for hovedfunksjonen til systemet.

2.9 Innkapsling og festeanordning

For at prototypen skal kunne monteres i produksjonslokalet må det gjøres valg for innkapsling og festeanordning. Fra erfaringer etter å ha vært ute i produksjonslokalet må følgende ytre påvirkninger gjøres tiltak mot for å beskytte elektronikken (SoC, akselerometer og batteri):

- pulverpartikler og fuktighet som kan gjøre skade på elektronikk
- mekaniske slag og vibrasjoner som kan riste løs eller skade elektronikk

Det må velges en innkapsling som beskytter mot påkjenningene som vises i Figur 2-15. I tillegg må det velges en festeanordning som gjør det mulig å montere innkapslingen på transportrøret.

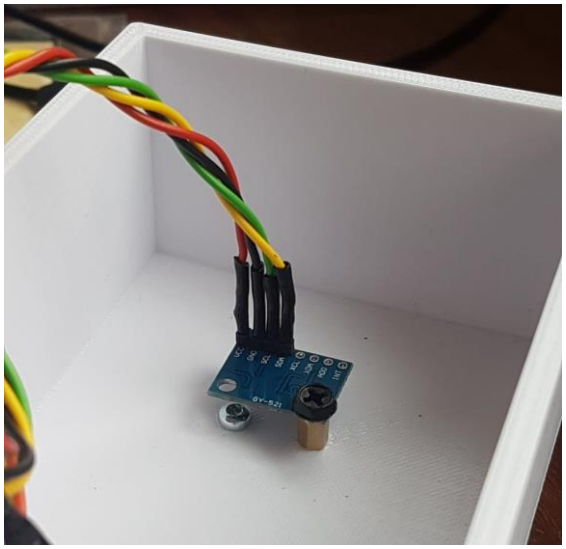


Figur 2-15 - Blokkskjema som viser ytre påkjenninger fra produksjonslokalet som kan skade prototypen

2.9.1 Innkapsling

Figur 2-16 – 3D modell av innkapslingen som består av lokk, kabinett og festeanordning. Modellen er tegnet av Inovyn med Fusion360 og printet av Inovyn.

Inne i boksen er det laget til en monteringsskrue som passer til akselerometeret, se Figur 2-17. Resten av utstyret ble ikke prioritert å montere fast til kabinettet for denne prototypen. Batteri, ledninger og SoC ligger løst i boksen som vist på Figur 2-18.



Figur 2-17 – Akselerometer montert i boks med monteringsskrue.

Figur 2-18 –Bildet viser hvordan boks ser ut med innhold

Festeanordningen er laget slik at braketten monteres på røret og festes med strips, som vist på Figur 2-19. Siden hele innkapslingen er laget av hardplast fra en 3D-printer hos Inovyn, er det usikkert hvor bra den beskytter mot de ytre påvirkningene som ble nevnt tidligere i kapittelet. Det er derfor utredet et annet forslag til permanent montering i kapittel 3.4.8.

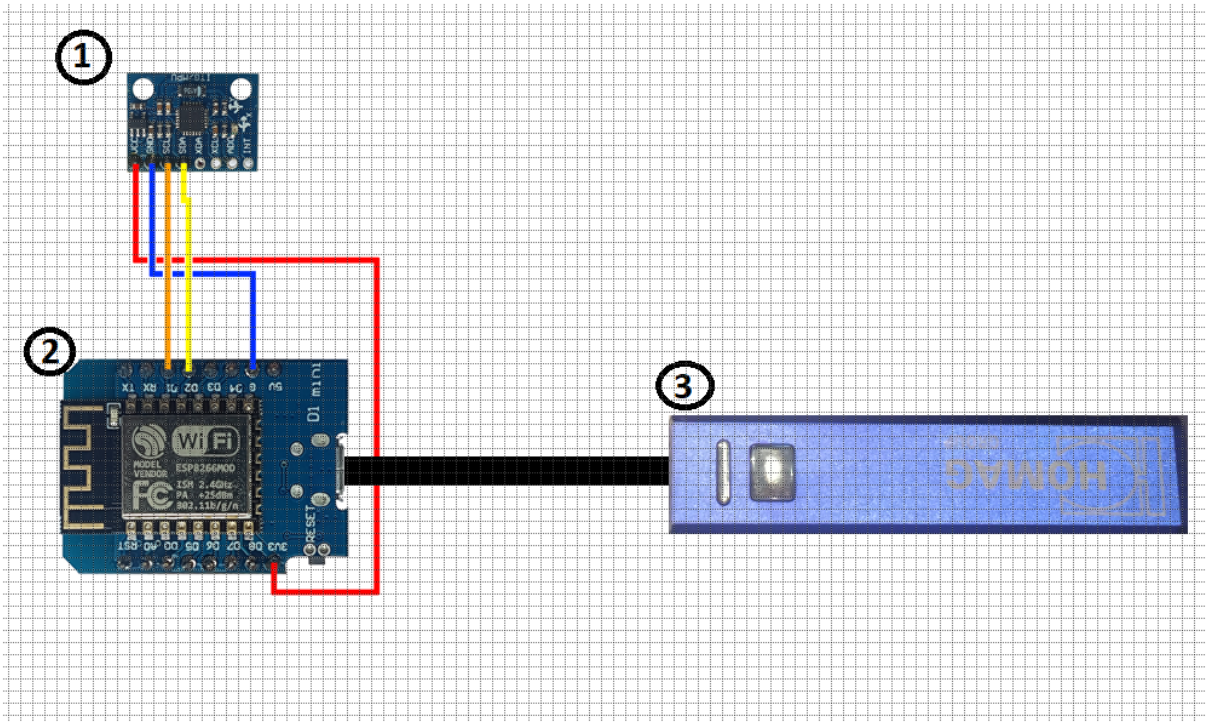
Figur 2-19 - Feste på rør med strips

2.9.2 Utstyrsliste for prototype brukt i prosjektet

Tabell 2-5 viser en oversikt over utstyr som er brukt til å lage prototypen, og Figur 2-20 viser hvordan prototypen er koblet. Prisene i tabellen viser at prototypen er innenfor budsjett nevnt i kapittel 1.5. Innkapsling og festeanordning er supplert av Inovyn, og derfor ikke prissatt.

Tabell 2-5 – Valg av utstyr for prototypen

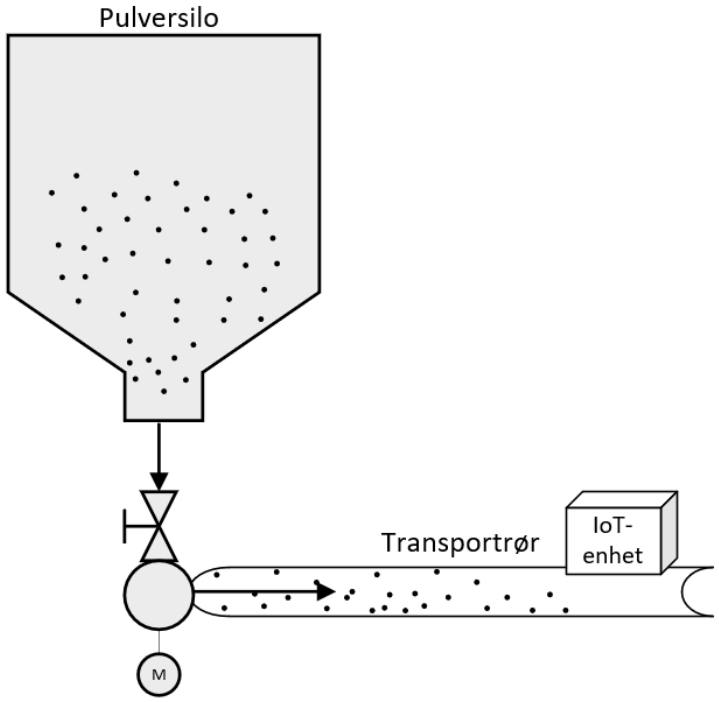
Komponent	Varenavn	Fabrikat	LxBxH (mm)	Antall	Pris (NOK)
SoC	WeMos D1 Mini	LOLIN	35x25x10	1	Ca. 50,- [23] [24]
Akselerometer	MPU GY-521	-	20x15x4	1	89,- [25]
Batteri	Powerbank Square 5V 2600mAh USB	Zogi	95x22x21	1	99,- [26]
Innkapsling	Koblingsboks 3D-printet	Inovyn	150x84x50	1	-
Festeanordning	Strips	-	-	2	-



Figur 2-20 – Koblingsskjema som viser hvordan akselerometeret (1) og batteriet (3) er koblet til utviklingskortet (2). 5 V fra batteriet kobles til utviklingskortet med USB-kabel.

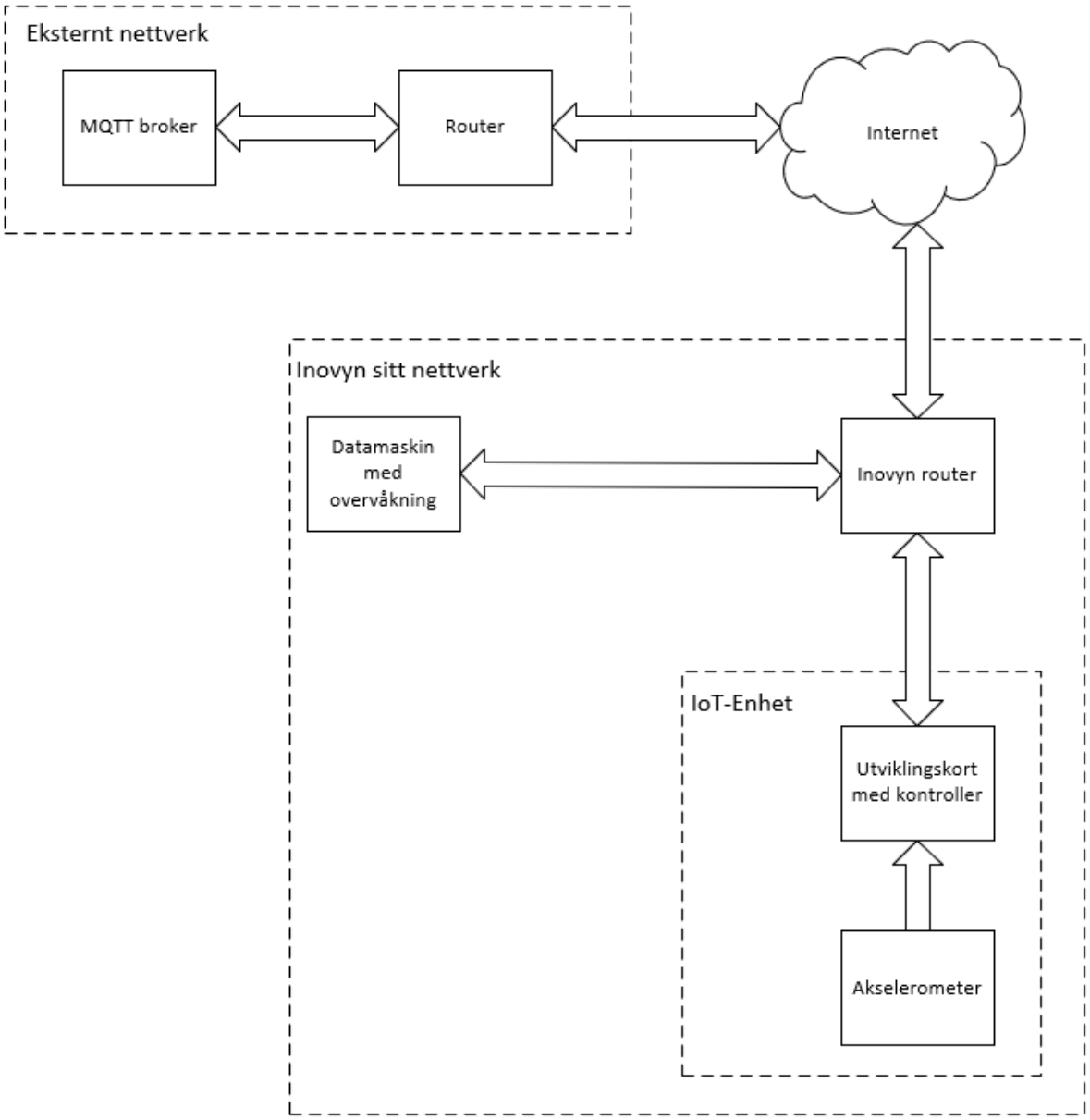
2.10 Funksjonstesting av IoT-enhet

For å kunne bekrefte at funksjonene til IoT-enheten (prototypen) virket, ble det utført test ute i produksjonslokalet. Enheten ble montert på transportrøret som utførte pulversending, se Figur 2-21. Vedlegg 1 viser en sjekkliste med definerte kontrollpunkter som kontrolleres under testingen, forutsatt at; systemets dataflyt er satt opp på riktig måte slik som systemskissen på Figur 2-22 illustrerer, og er montert og koblet opp som på figur Figur 2-12 og Figur 2-20.



Figur 2-21 – Oversiktsbilde av pulvertransportsystemet med montert IoT-enhet under testing.

På grunn av Covid-19 utbruddet ble det ikke mulig for gruppemedlemmene å utføre testing i produksjonslokalet hos Inovyn. Som et alternativ ble MQTT-brokeren satt opp på et eksternt nettverk, og Inovyn monterte IoT-enheten i produksjonslokalet.



Figur 2-22 – Systemskisse som viser hvordan dataflyten mellom enhetene er satt opp under testingen. IoT-enheten, som er montert i produksjonslokalet til Inovyn, utveksler informasjon med MQTT-brokeren gjennom routeren hos Inovyn og routeren til det eksterne nettverket. På samme måte blir informasjon utvekslet mellom Inovyn sin datamaskin og MQTT-brokeren.

2.10.1 Lesing av testresultater

Inovyn har overvåking av trykket i røret. Trykket korrelerer med pulvermengden i røret. Det ble foreslått at tilstandsstatus fra IoT-enheten ble sammenstilt med trykkmålingene i samme plot. Dette for å gi en indikasjon på om IoT-enheten sine deteksjoner er ok. Grafene i Figur 2-23 viser hvordan målingene sammenstilles. Alle grafene som er brukt til å sammenstille måledataene er utklipp fra Inovyn sine overvåkningssystemer, og er ikke laget av student-gruppa. Dette gjelder for Figur 2-23 og alle grafene i kapittel 3.

Figur 2-23 – Sammenstilling av måledata. Den øverste grafen viser rådata fra Inovyn sin trykksensor i transportrøret (Pressure sensor RAW readout). Langs y-aksen er amplitude og langs x-aksen er klokkeslett. Den nederste grafen viser tilstandene fra IoT-enheten langs y-aksen (USN pipe state). Langs y-aksen er tilstand, i henhold til tabell Tabell 2-6. Langs x-aksen er klokkeslett.

I overvåkningssystemet til Inovyn er IoT-enheten navngitt som «USN pipe state». Tilstandene «av», «luft» og «pulver» blir representert i grafen som henholdsvis «0», «1» og «2».

Inovyn foreslår at ved å ta i bruk avlesningen som ble illustrert på Figur 2-23 blir det mulig å konkludere om prototypen responderer på de riktige tilstandene. Trykket vil ikke være konstant ved de ulike tilstandene, hvor noe av grunnen er forklart i kapittel 3.4.5. Inovyn foreslår at trykkavlesningen kan tolkes slik:

- Dersom trykket er «lavt» antas det ingenting i røret (av).
- Dersom trykket er «høyt» antas det pulver i røret (ok drift).
- Dersom det blir et «fall» i trykket antas det kun luft i røret (ikke ok drift).

2.10.2 Forventa testresultater

En kan se i Tabell 2-6 hva som er forventet å lese av under testing. Tilstandene er beskrevet detaljert i kapittel 1.1.

Tabell 2-6 – Forventet resultat fra test.

Tilstand til transportrøret	Forventet «USN pipe state» (y-akse i graf)	Forventet «Pressure sensor RAW readout» (y-akse i graf)
Ingenting i røret: transportsystemet er av	0	«lave» verdier
Kun luft i røret: uønsket	1	«fall» i verdiene
Pulver røret: normal drift	2	«høye» verdier

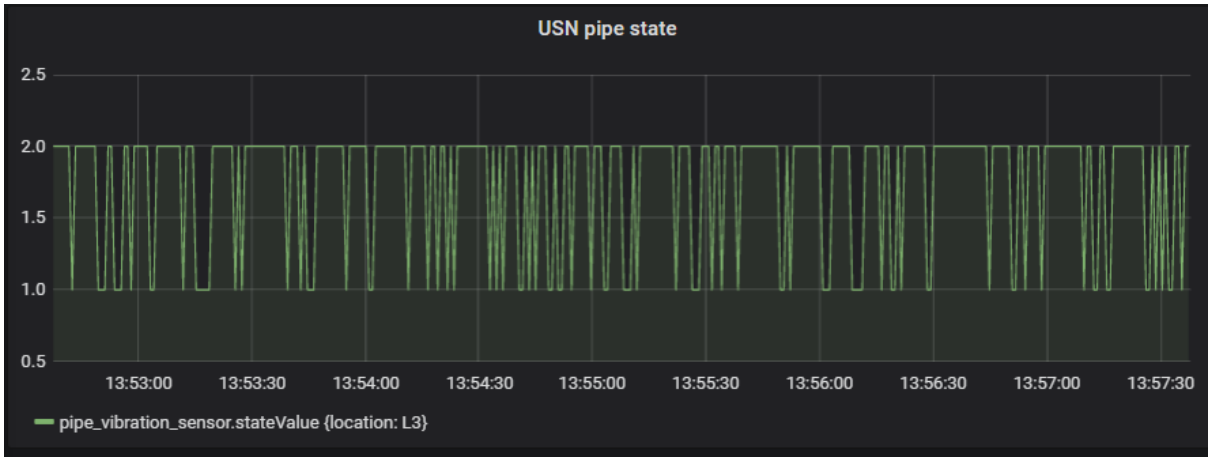
3 Resultater og diskusjon

Testresultater er presentert og diskutert i dette kapittelet. Testene er grunnet Covid-19 ikke utført av gruppemedlemmene da det ikke kunne bli gitt tilgang til produksjonsanlegget. Testene ble satt opp og utført av personalet i Inovyn. Dette gjør at det heller ikke var anledning til å følge tekstdokumentet slik det var planlagt. I tillegg har det ikke vært mulighet for å kontrollere alle testvariabler. Dette betyr at selv med tilsynelatende gode resultater fra test tre, er ikke resultatene pålitelige.

Se kapittel 2.10 for testoppsett og leseveileding for grafene i dette kapittelet.

3.1 Test én

Enheten tok i denne testen fire sampleserier, fouriertransformerte hver serie og tok deretter snittet av disse fouriertransformene for å glatte ut målingene. Hvordan dette er gjort er forklart i kapittel 2.8.3.



Figur 3-1 - Utklipp av linjestatus gitt av IoT-enhet ved første test. Leseveiledning for graf finnes i kapittel 2.10, og informasjon om «USN pipe state» i Tabell 2-6.

Som Figur 3-1 viser, ligger statusen fra enheten for det meste på «pulver i røret», med dipper til at det går «kun luft i røret». Grunnet testsituasjonen er det flere usikkerheter i testen. Mengden pulver som faktisk gikk i røret er estimert til mellom 30 % og 60 % av Inovyn, basert på trykket observert fra et manuelt manometer i transportrøret. Det er også usikkert om vibrasjoner utenfra påvirket sensoren under testen. Det er altså usikkert hva som forårsaker dippene over til status «luft», altså til verdi 1 i grafen. Det kan eksempelvis være forstyrrelser eller dårlig design av enhet.

Denne testen ble ikke regnet som en suksess. Det ble derfor utført en ny test på en annen transportlinje med trykkmåling montert, for å bedre kunne se reell tilstand på pulversendingen. Se kapittel 2.10.1 for forklaring av trykkmålingen.

3.2 Test to

Andre test ble utført over et lenger tidsrom. I oppstarten av pulversendingen var det problemer med å få pulvermatingen til å være stabil, og det ble derfor ikke anledning til å lagre referansebilde for tilstanden «pulver» for dette transportrøret. Det betyr at referansebildet for «pulver» er det samme som fra test én, altså en annen transportlinje, hvor vibrasjonsbildet kan være noe annerledes. Trykkmålingen øverst i Figur 3-2 viser, ifølge Inovyns antagelser, at det er stabil, uavbrutt pulversending i transportlinjen.

Figur 3-2 - Utklipp av linjestatus gitt av IoT-enhet og rådata fra trykkmåling ved andre test. Leseveiledning for graf finnes i kapittel 2.10, og informasjon om «USN pipe state» og «Pressure sensor RAW readout» i Tabell 2-6.

Likevel er det dipper ned i status «luft». Dette er ikke ønskelig. Etter test to ble det gjort endringer i programmet, slik at antallet sampleserier som det blir gjort en fouriertransformasjon av, og så tatt gjennomsnitt av, er økt fra 4 til 16. Gjennomsnittsbilde er forklart i kapittel 2.8.3. Endringen ble gjort for at enheten ikke skal reagere like fort på raske variasjoner.

Det ble deretter utført en ny test med oppdatert programvare.

3.3 Test tre

Grafen med rådata fra trykksensoren viser, ifølge Inovyn, i Figur 3-3 en uryddig oppstart av pulversending. Ifølge Inovyn er ikke pulversending stabil før rundt kl. 15:00, hvor det blir lagret nytt referansebilde for «pulver» tilstanden.

Figur 3-3 - Utklipp av linjestatus gitt av prototype og rådata fra Inovyns trykkmåling ved tredje test, hvor referansebilde for «pulver» blir endret rett før kl. 15:00. Leseveiledning for graf finnes i kapittel 2.10, og informasjon om «USN pipe state» og «Pressure sensor RAW readout» i Tabell 2-6.

Dette viser på Figur 3-3 at de grove tilstandsendringene nå blir detektert. Trykkmålingen viser, ifølge Inovyn, frem til 13:30 at systemet er av. Dette viser også vibrasjonsmåleren. Rundt 13:30 begynner trykkmåleren å vise at sendingen varierer mellom «kun luft» og «pulver». Her har fortsatt enheten et mer sensitivt referansebilde, altså med snitt av fire fouriertransformerte samples. Selv om øyeblikksbildet er et snitt av 16 fouriertransformerte samples blir dette altså ikke nok til å vise stabil tilstand. Med andre ord, selv om de har oppstartsproblemer skal man ikke se så mange dipper som man gjør mellom 13:30 og 15:00.

Figur 3-4 - Utklipp av linjestatus gitt av IoT-enhet og rådata fra trykkmåling ved tredje test, med nytt referansebilde for «pulver». Leseveiledning for graf finnes i kapittel 2.10, og informasjon om «USN pipe state» og «Pressure sensor RAW readout» i Tabell 2-6.

Etter 15:00 blir, ifølge Inovyn, sendingen mer stabil, og det blir sendt kommando til enheten om å lage nytt referansebilde for tilstanden «pulver». I kombinasjon med at sendingen blir mer stabil, og enheten får matchende, mindre følsomt referansebilde å sammenligne øyeblikksbildet med, gir dette et relativt stabilt signal om at normal pulversending pågår.

Dippen rett etter 15:13 ser ut til å være riktig, da lavere trykk i linjen i kort tid betyr at det er mindre pulver, og da mindre motstand i røret. Dette kan forklares med at pulveret i siloen klumper seg et øyeblikk, og ikke havner ned i skovlen. Denne løsner antakelig, og normal sending gjenopptar. Dette er ikke garantert grunnen, men er en av flere mulige årsaker. IoT-enheten har altså antageligvis detektert stopp i sending, altså status «kun luft i rør», på korrekt vis.

Dippen etter kl. 15:20 er ikke like forklarlig. Forslag til løsning for denne typen «feil» er kommentert i kapittel 3.4.6.

3.4 Forslag til videre arbeid

3.4.1 Systemstruktur

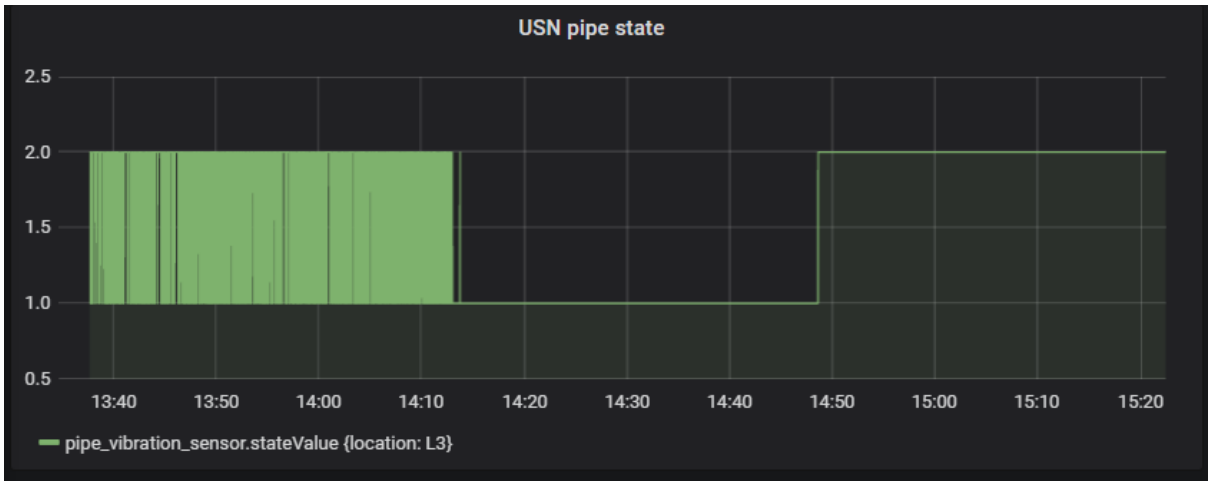
Det kan være aktuelt å ha vibrasjonsenheter på hvert transportrør for å få et fungerende system. Det er da en vurderingssak om hver av disse skal ha kontakt med MQTT via enkel systemstruktur, vist i Figur 2-9 i kapittel 2.5, eller om det skal brukes f.eks. en Raspberry Pi eller en annen kraftig minidatamaskin som mellomtjener for å lette arbeidstrykket på det overordnede systemet, som vist i Figur 2-8. Hvis det blir montert vibrasjonsenheter på mer utstyr enn transportlinjene vil dette være enda mer aktuelt.

3.4.2 Forstyrrelser fra andre linjer

Det er flere ting som må gjøres før systemet kan detektere pulversendingstilstanden pålitelig. Vibrasjoner som blir overført mellom linjene må kunne detekteres, og det kan da være aktuelt å lage flere enheter som sitter på hver sitt transportrør. Enhetene gir beskjed om vibrasjonen som kommer fra det røret de sitter på, og da kan det også lages til en beskjed om hva de andre sensorene på de forskjellige rørene kan forvente av forstyrrende vibrasjoner. Dette sammensatt med trykkmåling utført på linjene kan gi et helhetlig bilde av tilstanden i transportrørene.

3.4.3 Kontroll om linje faktisk er på

Det bør også lages en logikk som gjør at beskjeder fra vibrasjonsmålere er gyldig kun når pulversendingen er slått på. Som vist i Figur 3-5 vil enheten gi beskjed om at det går pulver i røret når det er av mens naborør sender pulver og lager støy. Dette er på grunn av den relativt lille forskjellen mellom tilstanden «av» og «pulver», vist i Figur 2-5 i kapittel 2.3. Dette er ikke riktig tilstand, men det er ingen måte for en enkeltstående enhet å vite det.



Figur 3-5 - Sensor gir beskjed om at det er «pulver i rør» selv om linje ikke sender. Leseveiledning for graf finnes i kapittel 2.10, og informasjon om «USN pipe state» i Tabell 2-6.

3.4.4 Energieffektivisering

Slik som prototypen er laget, arbeider den kontinuerlig. Dette er lite energieffektivt. Med en 8,8 Ah powerbank kan enheten jobbe kontinuerlig et sted mellom to og tre dager, ifølge test hos Inovyn. Det har ikke blitt overlevert data på dette, kun muntlig informasjon. Eksempel på energieffektiviseringstiltak kan være at den ikke trenger å kontinuerlig ta samples, ikke trenger å kontinuerlig sende beskjeder over MQTT eller at den kan gå i dvale når den ikke trenger å jobbe. Med dette vil batteritiden kunne bli lenger. En videre vurdering som kan gjøres er om batterilevetiden kan bli så lang at alle enhetene skal kjøres på batteri, eller om det er mer kostnadseffektivt å legge fast strøm til enhetene. Enhetene i flertall er forklart i kapittel 3.4.1.

3.4.5 Varierende trykk, varierende vibrasjonsbilde

PVC-pulveret blir sendt fra siloene som IoT-enheten(e) står montert etter, til diverse andre steder hvor det varierer hvor langt rørstrekket er. Dette gjør at fra en sending til en annen blir det brukt forskjellig trykk i sendeluften for å sende pulveret forskjellige distanser. Dette gjør igjen at vibrasjonsbildet antakelig blir forskjellig ved forskjellige sendinger. Det må lages kode som kan hente ut fra styresystemet hvilke destinasjon det sendes til og som da kan gi beskjed til den enkelte IoT-enhet hvilke referansebilder den skal bruke.

3.4.6 Bestemme gyldighet av status

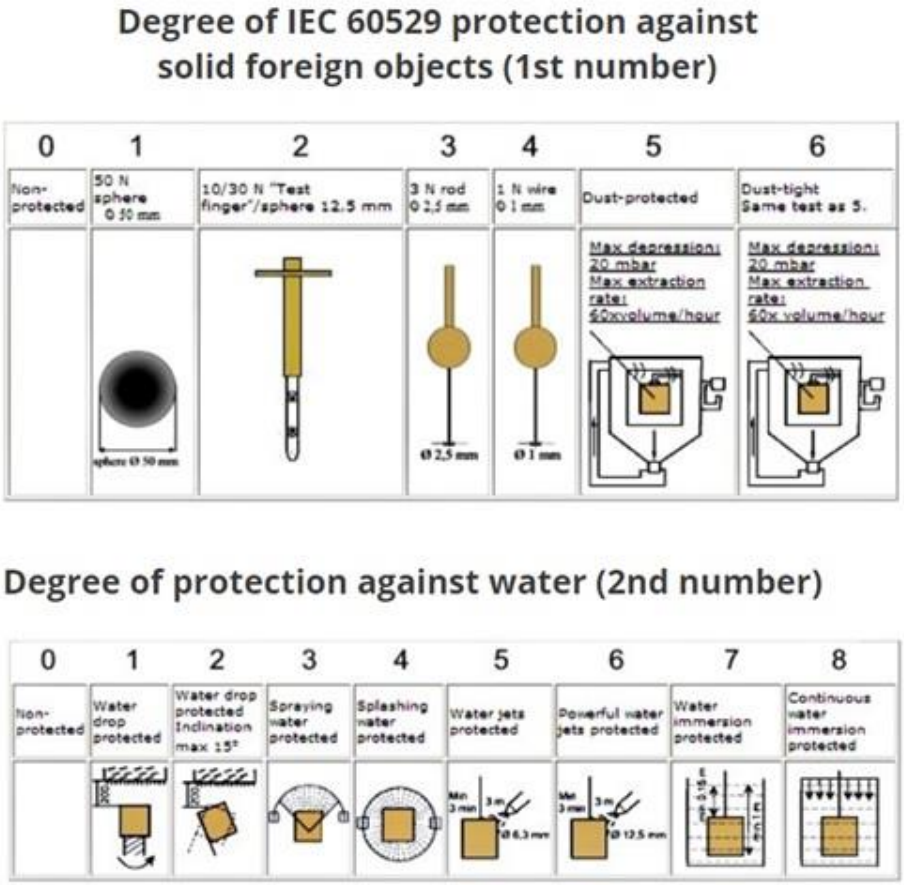
Enhetene vil alltids kunne påvirkes av forstyrrelser, gjerne kortvarige, som gjør at de vil melde fra om en annen status enn det som faktisk er reelt. En løsning på dette kan være at status må ha vært stabil i et antall programsykluser, eller en gitt tid, før det blir tatt nytt valg av status. En ting å vurdere er hvor denne filtreringen skal gjøres. Om det blir gjort i vibrasjonsenheten(e) er det mindre for det overordnede systemet å gjøre, men informasjonen om disse dippene går tapt. Om det er ønskelig å logge all informasjon fra vibrasjonsenheten(e) kan denne filtreringen gjøres i det overordnede systemet. Evt. kan det gjøres i en databehandlingsenhet som kan settes opp for å ta inn signaler fra flere vibrasjonsenheter, for så å videresende til logging og filtrert til overordnet system.

3.4.7 Lodding av ledningsfester

Ulike situasjoner kan føre til kraftige vibrasjoner i transportrørene, og ledninger og plugger må være godt sikret. I prototypen er det på ledningene mellom akselerometeret og SoC brukt klem-hylser med pinner av typen som kan trykkes ned i et «breadboard». Under testing falt en ledning mellom akselerometer og SoC ut. Ledningsendene bør loddes fast i begge ender, og ikke bare til akselerometer, slik det er nå. Plugger for strømforsyning har holdt seg på plass under testing, men det bør vurderes om en annen løsning skal brukes der også. Et eksempel kan være at innfesting til SoCen i den 3D-printede boksen kan lages slik at strømpluggen blir kilt fast i SoCen når den er montert.

3.4.8 Innkapsling for permanent montering

All elektronikk sitter i samme innkapslingen. Kapslingen bør tilfredsstille krav fra standarden IEC 60529 om beskyttelse av støv og vann. Ut fra standarden (se Figur 3-6) anbefales kapslingen å være godkjent med minimum IP 54 [27].



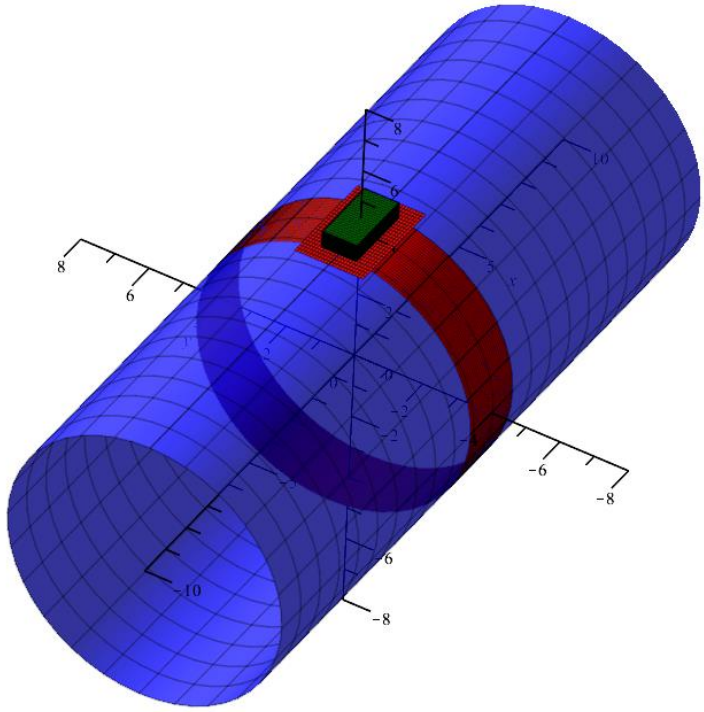
Figur 3-6 – Utklipp fra tabell som viser IP-graderingene [27]

Ved permanent montering bør innkapslingen være tilstrekkelig tett. Sikringen til røret bør også være tilstrekkelig sterk og holdbar. Dersom det er ønskelig å forsyne prototypen med fast strøm kan powerbanken fjernes. På grunn av at det blir nødvendig med kabel må koblingsboksen utstyres med passende kabelgjennomføring.

Stripsen bør byttes ut til en sterkere festeanordning som tåler de mekaniske belastningene over lengre tid. Alternativet blir følgende:

- Velge en koblingsboks med IP >= 54.
- Velge rørklammer som passer rørdimensjonen.
- Velge en stålplate med passende dimensjon til koblingsboksen som sveises på rørklammeret.
- Montere koblingsboksen på stålplata med bolter og låsemuttere.

Illustrasjonen i Figur 3-7 viser hvordan dette kan se ut.



Figur 3-7 – Illustrasjon av den permanente innkapslingen med en sterkere festeanordning. Blå sylinder er transportrøret, rød sirkel er rørklammeren, rødt rektangel er stålplata og grønt prisme er koblingsboksen.

Det er verdt å nevne at støy i form av EMI kan bli et problem. Til videre arbeid bør det gjøres en utredelse på:

- Hvordan beskytte IoT-enheten for EMI.
- Hvordan forebygge at spredning av EMI kommer fra IoT-enheten.

4 Konklusjon

Målene i innledningen 1.4 og nærmere detaljert i vedlegg 2, er nådd ved å gjøre følgende:

- Vurdert ulike SoC og valgt Wemos D1 mini.
- Vurdert ulike IoT-grensesnitt og valgt MQTT.
- Laget en prototype med ett akselerometer tilkoblet og grensesnitt med MQTT mot Inovyn.
- Laget programvare som analyserer frekvensbildet med FFT, samt har teach-funksjonalitet.
- Utviklet testdokument.
- Holdt prototypen innenfor budsjett.

Prototypen er i stand til å skille mellom de tre tilstandene «pulver», «luft» og «av» når det ikke er betydelige forstyrrelser fra andre linjer. Status blir gjort tilgjengelig for Inovyn via MQTT.

Funksjon i prototypen står i samsvar med prosjektets beskrivelse, og alle delmål er oppnådd. Gjennom prosjektets utførelse har vibrasjonsbildet i transportrørene hos Inovyn vist seg å være mer kompleks enn først antatt. Prototypen tar ikke høyde for støy og at vibrasjoner i produksjonen varierer. Videre utvikling må utføres for å få en tilstandsdeteksjon som Inovyn kan bruke, hvor noe er diskutert i kapittel 3.4.

5 Referanser

[1] H. Holden, «Store Norske Leksikon,» 38. November 2019. [Internett]. Available: https://snl.no/fouriertransformen. [Funnet 7. Mai 2020].

[2] MIT, «MIT News,» 18. Januar 2012. [Internett]. Available: http://news.mit.edu/2012/faster-fourier-transforms-0118. [Funnet 4. Mars 2020].

[3] Ø. Grøn, «Store Norske Leksikon,» 28. Juni 2018. [Internett]. Available: https://snl.no/bølge. [Funnet 7. Mai 2020].

[4] H. Holden, «Store Norske Leksikon,» 12. Mars 2019. [Internett]. Available: https://snl.no/fourieranalyse. [Funnet 7. Mai 2020].

[5] D. S. Gage, «Wikimedia Commons,» 21. November 2016. [Internett]. Available: https://commons.wikimedia.org/wiki/File:FFT_of_Cosine_Summation_Function.png. [Funnet 4. Mars 2020].

[6] J. E. Vatne, «Store Norske Leksikon,» 31. Mai 2017. [Internett]. Available: https://snl.no/diskret_matematikk. [Funnet 7. Mai. 2020].

[7] P. B. Andersen, «Store Norske Leksikon,» 23. Januar 2018. [Internett]. Available: https://snl.no/FFT. [Funnet 7. Mai 2020].

[8] G. Jahns, W. Kowalczyk og K. Walter, «ScienceDirect,» 27. Mai 2017. [Internett]. Available: https://www.sciencedirect.com/science/article/pii/S1474667017412651. [Funnet 30. Mars 2020].

[9] OASIS, «MQTT.org,» OASIS, [Internett]. Available: http://mqtt.org/faq. [Funnet 23 04 2020].

[10] O.-M. H. Steinnes, «NTNU,» 24 03 2019. [Internett]. Available: https://www.ntnu.no/wiki/display/miiot/Mosquitto+MQTT. [Funnet 07 05 2020].

[11] OASIS, «AMQP,» OASIS, [Internett]. Available: https://www.amqp.org/about/what. [Funnet 23 04 2020].

[12] STOMP, «STOMP Protocol Specification,» STOMP, [Internett]. Available: https://stomp.github.io/stomp-specification-1.2.html. [Funnet 23 04 2020].

[13] CoAP, «CoAP,» [Internett]. Available: https://coap.technology/impls.html. [Funnet 23 04 2020].

[14] M. ASR, S. M and D. V. P. I, "COMPRESSOR HEALTH MONITORING USING IOT," *IJMPERD,* vol. 8, no. 3, pp. 117-124, 2018.

[15] InvenSense, «InvenSense,» 8. August 2013. [Internett]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf. [Funnet 6. Mai 2020].

[16] Arduino, «Arduino Official Store,» Arduino, [Internett]. Available: https://store.arduino.cc/arduino-uno-rev3. [Funnet 4 Mars 2020].

[17] WeMos, «WEMOS Electronics,» WeMos, 28 Desember 2018. [Internett]. Available: https://wiki.wemos.cc/products:d1:d1_mini. [Funnet 4 Mars 2020].

[18] Adafruit, «Adafruit,» Adafruit, [Internett]. Available: https://www.adafruit.com/product/3405. [Funnet 4 Mars 2020].

[19] J. D. Cutnell og K. W. Johnson, «Physics,» i *Physics 4th ed*, New York: Wiley, 1998, p. 466.

[20] Arduino, «Arduino Playground,» Arduino, [Internett]. Available: https://playground.arduino.cc/Main/MPU-6050/. [Funnet 31 Mars 2020].

[21] L. Fried, «Arduino,» Arduino, 07 02 2017. [Internett]. Available: https://www.arduino.cc/en/guide/libraries. [Funnet 08 05 2020].

[22] I. Earle F. Philhower, «GitHub,» GitHub, Inc, [Internett]. Available: https://github.com/esp8266/Arduino. [Funnet 08 05 2020].

[23] Amazon, «Amazon,» 12. Mai 2020. [Internett]. Available: https://www.amazon.com/wemos-d1-mini/s?k=wemos+d1+mini. [Funnet 12. Mai 2020].

[24] LOLIN, «AliExpress,» 12. Mai 2020. [Internett]. Available: https://lolin.aliexpress.com/store/1331105. [Funnet 12. Mai 2020].

[25] D. Impuls, «Digital Impuls,» 12. Mai 2020. [Internett]. Available: https://www.digitalimpuls.no/sensorer/. [Funnet 12. Mai 2020].

[26] C. Ohlson, «Clas Ohlson,» 12. Mai 2020. [Internett]. Available: https://www.clasohlson.com/no/Clas-Ohlson,-Powerbank-3350-mAh/. [Funnet 12. Mai 2020].

[27] «CVGStrategy.com,» CVG Strategy, [Internett]. Available: https://cvgstrategy.com/iec-60529/. [Funnet 25 04 2020].

Vedlegg

Vedlegg 1 – Testdokument

Vedlegg 2 – Prosjektbeskrivelse

Vedlegg 3 – Programkode

Vedlegg 4 – Programflyt

Prosjekt: PRH612-1 20V Bacheloroppgave	**Oppdragsgiver:** Inovyn Norge AS
Testforbereder: Henning Engesæt	
Dato forberedt: 20.03.20	**Periode:** Vår 2020

Objekt	Forutsettinger for utførelse av test
1a	Batteripakke skal være oppladet.
1b	Indikatorlys på batteripakke skal være blått og klar til bruk.
2a	Kapsling med utstyr skal være fastmontert til rør.
2b	Akselerometer skal ikke ha slingring/bevegelse i montering.
3	Blått lys ved antenne på SoC skal blinke når «Reset»-bryter trykkes inn.
4	Wi-Fi med internettilgang skal være innen rekkevidde.
5a	MQTT-server skal svare på gitt adresse på port 1883.
5b	MQTT-server skal få en ekstra klient etter systemet startes.
6a	Melding om «off», «air» eller «powder» fra SoC skal kunne sees på topic «FFT/state».
6b	Melding «off», «air» eller «powder» skal sendes til SoC av bruker på topic «FFT/teach» ved respektive systemstadier.
7a	Melding «off» skal sees på topic «FFT/state» når systemet er av.
7b	Melding «air» skal sees på topic «FFT/state» når systemet bare sender luft.
7c	Melding «powder» skal sees på topic «FFT/state» når systemet sender pulver.
8	Melding «save» skal sendes til SoC av bruker på topic «FFT/teach» for lagring av systemstadier.
9a	SoC skal restartes av bruker via «Reset»-bryter.
9b	Melding «off» skal fortsatt sees på topic «FFT/state» når systemet er av.
9c	Melding «air» skal fortsatt sees på topic «FFT/state» når systemet bare sender luft.
9d	Melding «powder» skal fortsatt sees på topic «FFT/state» når systemet sender pulver.

Objekt	Beskrivelse av test	Godkjent (**x**)/ Underkjent (**-**):	Signatur:
1a	Batteripakke er oppladet?		
1b	Indikatorlys på batteripakke er blått og klar til bruk?		
2a	Kapsling med utstyr er fastmontert til rør?		
2b	Akselerometer har ikke slingring/bevegelse i montering?		
3	Blått lys ved antenne på SoC blinker når «Reset»-bryter trykkes inn?		
4	Wi-Fi med internettilgang er innen rekkevidde?		
5a	MQTT-server svarer på gitt adresse på port 1883?		
5b	MQTT-server får en ekstra klient etter systemet startes?		
6a	Melding om «off», «air» eller «powder» fra SoC sees på topic «FFT/state»?		
6b	Melding «off», «air» eller «powder» sendes til SoC av bruker på topic «FFT/teach» ved respektive systemstadier?		
7a	Melding «off» sees på topic «FFT/state» når systemet er av?		
7b	Melding «air» sees på topic «FFT/state» når systemet bare sender luft?		
7c	Melding «powder» sees på topic «FFT/state» når systemet sender pulver?		
8	Melding «save» sendes til SoC av bruker på topic «FFT/teach» for lagring av systemstadier?		
9a	SoC restartes av bruker via «Reset»-bryter?		
9b	Melding «off» sees fortsatt på topic «FFT/state» når systemet er av?		
9c	Melding «air» sees fortsatt på topic «FFT/state» når systemet bare sender luft?		
9d	Melding «powder» sees fortsatt på topic «FFT/state» når systemet sender pulver?		

Signatur oppdragsgiver:	Dato:

Universitetet i Sørøst-Norge

Fakultet for Teknologi, Naturvitenskap og Maritime fag

inovyn
An INEOS company

PRH612 Bacheloroppgave

Studieretning: Informatikk og automatisering

Prosjektgruppe: IA6-3-20

Tittel: IoT-enhet (Internet of Things) for akustikkanalyse av pneumatisk transport

Veileder: Nils Olav Skeie

Samarbeidspartner: Inovyn

Prosjektets bakgrunn:

Under produksjon av plastpulver blir pulveret fraktet gjennom fabrikken ved hjelp av pneumatiske transportsystemer. Problemer oppstår når matingen av pulver stopper uforutsett, og dette resulterer i at det kun går luft gjennom røret. Forskjellen i lyd når dette skjer er betydelig. Kan dette detekteres på en enkel og billig måte?

Målbeskrivelse for prosjektet:

Hovedmålet i prosjektet er å lage en prototype av et system som kan detektere når matinga av pulver stopper, og det kun går luft i røret. Det skal detekteres tre tilstander: ingen pulver i rør (ikke i drift), pulver i rør (ok drift) og kun luft i rør (ikke ok drift).

Gjennom prosjektet skal følgende delmål være oppnådd:

- Vurdere ulike «system on chip» (SoC) og velge en type som kan brukes for prototypen.
- Prototypen skal bestå av et SoC med behandling av ett eller flere akustiske signaler.
- Vurdere ulike IoT-grensesnitt og sammenlikne med MQTT-protokoll.
- Prototypen skal ha IoT-grensesnitt med anbefalt MQTT-protokoll mot fabrikken.
- Design av prototype.
- Det skal lages programvare for å analysere frekvensbildet gitt av pulver/ikke pulver i røret. Programvaren skal også ha en «teach»-funksjon.
- Utvikle testplan og testdokument for prototype.
- Det skal utarbeides en teknisk rapport som dokumenterer prosjektet.

Signatur

Veileder (dato og signatur):
Nils Olav Skeie

Studenter (dato og signatur):
Gaute Eriksen
Gaute Eriksen 5/3-20

Henning Engesæt
Henning Engesæt 4/3-20

Jarle Haugland Berge
Jarle H. Berge 4/3-20

William French
William French 4/3-20

```
/*
This code will need the following plugins for Arduino IDE installed to work:

ESP8266 at File>Preferences>Additional boards mananger URLs:
https://arduino.esp8266.com/stable/package_esp8266com_index.json

esp8266 by ESP8266 Community at Tools>Board>Boards Mananger... (2.6.3 last tested)

The following libraries at Tools>Manange Libraries:
arduinoFFT by Enrique Condes (1.5.5 last tested)
PubSubClient by Nick O'Leary (2.7.0 last tested)
*/

#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <PubSubClient.h>
#include <EEPROM.h>
#include "arduinoFFT.h"
#include "Wire.h"

/*-- Connectivity properties --*/
const char* ssid = "######";                //Wi-Fi SSID
const char* password = "####";     //Wi-Fi Password
const char* mqtt_server = "###.###.###.###"; //MQTT server address

/*-- Accelerometer --*/
const int MPU_ADDR = 0x68; //Address for accelerometer
int mpuAxis = 0x43; //Address for chosen axis. 0x43 for x-axis, 0x45 for y-axis,
0x47 for z-axis.

/*-- FFT --*/
const uint16_t samples = 128; //Number of FFT samples, must be power of 2 (32, 64,
128..)
String state; //Detected state, off/air/powder
int avgSample = 16; //Samples used for an average FFT image
double vReal[samples]; //FFT real values
double vImag[samples]; //FFT imaginary values
double vRef1[samples]; //Stored image for "powder" state
double vRef2[samples]; //Stored image for "air" state
double vRef3[samples]; //Stored image for "off" state

WiFiClient espClient; //Create Wi-Fi client object
PubSubClient client(espClient); //Create MQTT client object
arduinoFFT FFT = arduinoFFT(); //Create FFT object

/*-- Setup --*/
void setup() {
  Serial.begin(115200);
  Serial.println("Ready");
  EEPROM.begin(3500); //Using ~3.5mb of 4mb total
  loadReference();

  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  Wire.begin();
  Wire.beginTransmission(MPU_ADDR); // Begins a transmission to the I2C slave
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes up the MPU)
  Wire.endTransmission(true);

  /*-- Standard "example" OTA setup --*/
  ArduinoOTA.setHostname("D1 mini");
  ArduinoOTA.onStart([]() {
    String type;
    if (ArduinoOTA.getCommand() == U_FLASH) {
      type = "sketch";
    } else { // U_FS
      type = "filesystem";
    }
    Serial.println("Start updating " + type);
```

```
  });
  ArduinoOTA.onEnd([]() {
    Serial.println("\nEnd");
  });
  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
  });
  ArduinoOTA.onError([](ota_error_t error) {
    Serial.printf("Error[%u]: ", error);
    if (error == OTA_AUTH_ERROR) {
      Serial.println("Auth Failed");
    } else if (error == OTA_BEGIN_ERROR) {
      Serial.println("Begin Failed");
    } else if (error == OTA_CONNECT_ERROR) {
      Serial.println("Connect Failed");
    } else if (error == OTA_RECEIVE_ERROR) {
      Serial.println("Receive Failed");
    } else if (error == OTA_END_ERROR) {
      Serial.println("End Failed");
    }
  });
  ArduinoOTA.begin();
}

/*-- Loop --*/
void loop() {
  ArduinoOTA.handle();
  //Reconnect if disconnected
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  state = "";
  checkImage();                                       //Run checkImage function
  char sendMessage[state.length()+1];                 //Message to be sent over MQTT
  state.toCharArray(sendMessage, state.length()+1); //publish method only accepts
array of char, string with message is converted
  client.publish("FFT/state", sendMessage);           //Publishing message to topic:
FFT/state
  delay(500);                                         //Delay to slow down overall
progam speed
}

/*-- Fetch accelerometer values and convert to FFT image --*/
void runFFT(){
  for (uint16_t i = 0; i < samples; i++){
    Wire.beginTransmission(MPU_ADDR);
    //Starting with register (if x-axis chosen) 0x43 (GYRO_XOUT_H) [MPU-6000 and
MPU-6050 Register Map and Descriptions Revision 4.2, p.40]
    Wire.write(mpuAxis);
    //The parameter indicates that the ESP will send a restart. As a result, the
connection is kept active.
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_ADDR, 2, true); //Request a total of 2 registers

    //"Wire.read()<<8 | Wire.read();" means two registers are read and stored in
the same variable
    vReal[i] = Wire.read()<<8 | Wire.read(); //Reading registers: 0x43
(GYRO_XOUT_H) and 0x44 (GYRO_XOUT_L)
    vImag[i] = 0.0; //Imaginary part must be zeroed in case of looping to avoid
wrong calculations and overflows
  }
  FFT.Windowing(vReal, samples, FFT_WIN_TYP_HANN, FFT_FORWARD);  //Weigh data
  FFT.Compute(vReal, vImag, samples, FFT_FORWARD); //Compute FFT
  FFT.ComplexToMagnitude(vReal, vImag, samples); //Compute magnitudes
  delay(5);
}

/*-- Create average array of n FFT images and compare to reference arrays --*/
void checkImage(){
  double vImage[samples];
  memset(vImage,0,sizeof(vImage));
  double error1 = 0;
  double error2 = 0;
```

```
  double error3 = 0; //Variables for error
  int n = avgSample; //Number of sampled arrays to use for average

  for(uint16_t i=0; i<n; i++){
    runFFT(); //Fetch new values to vReal
    for (uint16_t i=0; i<samples; i++){
      vImage[i] += vReal[i]/n;
    }
  }

  /*-- Compares every element in current array with reference array --*/
  for (int i=0; i<samples; i++){
    error1 += abs(vImage[i]-vRef1[i]);
    error2 += abs(vImage[i]-vRef2[i]);
    error3 += abs(vImage[i]-vRef3[i]);
  }

  /*-- Serial output for debugging --*/
  Serial.println("-------");
  Serial.print("Error from powder image: ");
  Serial.println(error1);
  Serial.print("Error from    air image: ");
  Serial.println(error2);
  Serial.print("Error from    off image: ");
  Serial.println(error3);

  /*-- Setting state to error with lowest value --*/
  if (min(error1, error2) == error1) {
    if (min(error1, error3) == error1) {
      state = "powder";
    }
  }
  if (min(error2, error1) == error2) {
    if (min(error2, error3) == error2) {
      state = "air";
    }
  }
  if (min(error3, error1) == error3) {
    if (min(error3, error2) == error3) {
      state = "off";
    }
  }
}

/*-- Compare topic and incomming message from MQTT --*/
void handleMessage(String topic, String message){
  if(topic == "FFT/restart"){
    if(message == "restart"){
      Serial.println("resetting..");
      ESP.restart();
    }
  }
  if(topic == "FFT/teach"){
    if(message == "powder"){
      Serial.println("teaching powder image..");
      //Passing first reference array to function as a reference variable starting
at 0
      teachReference(&vRef1[0]);
    }
    else if(message == "air"){
      Serial.println("teaching air image..");
      //Passing second reference array to function as a reference variable starting
at 0
      teachReference(&vRef2[0]);
    }
    else if(message == "off"){
      Serial.println("teaching off image..");
      //Passing third reference array to function as a reference variable starting
at 0
      teachReference(&vRef3[0]);
    }
    else if(message == "save"){
      Serial.println("saving images to flash memory");
      saveReference(); //Stores all reference arrays to flash memory
```

```
    }
    else if(message == "load"){
      Serial.println("loading images from flash memory");
      loadReference(); //Loads all reference arrays from flash memory
    }
  }
  if(topic == "FFT/axis"){
    if(message == "x"){
      Serial.println("Changing axis to x..");
      mpuAxis = 0x43; //0x43 for x-axis
    }
    else if(message == "y"){
      Serial.println("Changing axis to y..");
      mpuAxis = 0x45; //0x45 for y-axis
    }
    else if(message == "z"){
      Serial.println("Changing axis to z..");
      mpuAxis = 0x47; //0x47 for z-axis.
    }
  }
//  if(topic == "FFT/samples"){
//    samples = message.toInt();
//  }
  if(topic == "FFT/avgSamples"){
    avgSample = message.toInt();
  }
}

/*-- Store n average FFT image in referenced array --*/
void teachReference(double *vImage){
  for (uint16_t i=0; i<samples; i++){
      vImage[i] = 0;
  }
  int n = avgSample; //Number of sampled arrays to use for average
  for(uint16_t i=0; i<n; i++){
    runFFT(); //Fetch new values to vReal
    for (uint16_t i=0; i<samples; i++){
      vImage[i] += vReal[i]/n;
    }
  }
  delay(5);
  Serial.println("done");
}

/*-- Permanently store referenced arrays --*/
void saveReference(){
  int addr = 0;

  for (uint16_t i=0; i<samples; i++){
    EEPROM.put(addr, vRef1[i]);
    addr += sizeof(double);
  }
  for (uint16_t i=0; i<samples; i++){
    EEPROM.put(addr, vRef2[i]);
    addr += sizeof(double);
  }
  for (uint16_t i=0; i<samples; i++){
    EEPROM.put(addr, vRef3[i]);
    addr += sizeof(double);
  }

  if (EEPROM.commit()) {
    Serial.println("References successfully saved");
  }
  else {
    Serial.println("ERROR! EEPROM commit failed");
  }
}

/*-- Permanently store referenced arrays --*/
void loadReference(){
  int addr = 0;

  for (uint16_t i=0; i<samples; i++){
```

```
    EEPROM.get(addr, vRef1[i]);
    addr += sizeof(double);
  }
  for (uint16_t i=0; i<samples; i++){
    EEPROM.get(addr, vRef2[i]);
    addr += sizeof(double);
  }
  for (uint16_t i=0; i<samples; i++){
    EEPROM.get(addr, vRef3[i]);
    addr += sizeof(double);
  }
}

/*-- Standard "example" Wi-Fi setup --*/
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

/*-- Standard "example" reconnect function --*/
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266";
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      client.subscribe("FFT/restart");
      client.subscribe("FFT/teach");
      client.subscribe("FFT/axis");
      client.subscribe("FFT/avgSamples");
    }
    else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

/*-- Standard "example" callback function,
     modified to handle incomming messages with handleMessage function --*/
void callback(String topic, byte* payload, unsigned int length) {
  String message;
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (uint16_t i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
    message += (char)payload[i]; //Create string of incomming characters
  }
  Serial.println("");
  handleMessage(topic, message); //Passes topic and message to function
  message = "";
}
```

Programflyt - Vedlegg 4