Universitetet i Sørøst-Norge

www.usn.no

TS3000 1 Bachelor 23V

Utvikling av skyarkitektur og infrastruktur for industrielle IoT-enheter

IA6-5-23

Fakultet for teknologi, naturvitenskap og maritime fag
Campus Porsgrunn

Universitetet i Sørøst-Norge

www.usn.no

Emne: TS3000 Bacheloroppgave

Tittel: Utvikling av skyarkitektur og infrastruktur for industrielle IoT-enheter

Denne rapporten utgjør en del av vurderingsgrunnlaget i emnet.

Prosjektgruppe: IA6-5-23 **Tilgjengelighet:** Åpen

Gruppedeltakere: Djupvik Torbjørn Andreas Apeland
Nylund Peter Sneltvedt
Telle Edwin

Veileder: Halvorsen Hans-Petter

Prosjektpartner: PowerTech Engineering AS

Godkjent for arkivering: _____

Sammendrag:

Bacheloroppgave for PowerTech Engineering har bakgrunn i et ønske om å få i gang en trådløs IoT-enhet opp mot deres BMS/SCADA-system Zaphire. Oppgaven går ut på å koble opp og konfigurere enheten for å snakke med Zaphire og å utvikle det Zaphire trenger for å motta kommunikasjonen. Det er også et ønske om å gjøre oppsettprosessen så smerteløs og skalerbar som mulig. For å oppnå dette ble MQTT, LwM2M og CoAP vurdert som kommunikasjonsprotokoller. MQTT ble valgt fordi det allerede hadde delvis støtte fra Zaphire, et enklere design og det oppfylte alle kravene til systemet. Det blir gjennomgått hvordan IoT-enheten snakker med en MQTT broker og hvordan det blir videresendt til Zaphire. Sikkerhet er også et stort fokus for å beskytte kundedata.

Forord

Denne bacheloroppgaven avslutter en treårig elektroingeniørgrad innen informatikk og automatisering ved Universitetet i Sørøst-Norge. Årene på universitetet har bidratt til mye ny kunnskap og nye perspektiver. I den forbindelse har det åpnet seg nye interesseområder for oss i gruppen. Det har blant annet påvirket valgt tema for denne bacheloroppgaven.

Først vil vi takke PowerTech Engineering for forslaget til oppgaven, hjelp og støtte til å gjennomføre den. Spesielt vil vi takke Patrick Stave for å være en stor pådriver og støttespiller med oppfølging og forslag under hele bachelorperioden.

Vil også rette en takk til Norsk Titanium med Jon Forbord i spissen for deres interesse i prosjektet og for samarbeidet rundt montering av enheten vår og innsamling av data fra en ekte prosess. Dette gjorde at vi fikk bekreftet at løsningen vi hadde utviklet fungerer i praksis.

Til slutt vil vi takke Hans-Petter Halvorsen som har vært veilederen vår. Takk for at du veiledet oss på veiledningsmøter og den opplæring i struktur og orden på rapporter du har gitt. Takk for innsats og støtte under perioden.

Porsgrunn, 22. mai 2023

Nomenklaturliste

BMS – er kort for Building Managment System og er et system for å regulere bygninger med tanke på blant annet temperatur og ventilasjon.

Broker – refererer til MQTT broker. Denne har ansvar for å motta og distribuere meldinger fra sender klient til mottaker klientene som bruker MQTT.

Bug – refererer til software bugs. Dette er feil eller uønskede hendelser i softwaren.

CoAP – er kort for Constrained Application Protocol. Dette er en protokoll for å overføre data ofte brukt i IoT-applikasjoner.

Container – container refererer til et fullstendig softwaremiljø som kan inneholde spesifikke programpakker valgt av den som bygde den. Disse kan enkelt startes opp individuelt eller flere uavhengig av installasjon på en fysisk maskin.

DIP-switcher – dette er skyvebrytere, ofte små brytere for å velge mellom to alternativer i en elektrisk krets.

Frontend – Den delen av programmet som er vist til brukeren som kan se og trykkes på.

Header – er data som kommer før selve innholdet i en melding. Denne beskriver blant annet hvordan meldingen er strukturert.

I/O - In/Out refererer til innganger og utganger, ofte på en fysisk enhet. Dette kan være blant annet lyd i form av minijack, data i form av USB eller Ethernet. I denne oppgaven blir det mest referert til datainnganger og -utganger for målesignaler.

Image – er en kopi av et softwaresystem. Dette brukes som en oppskrift for å opprette instanser av samme system.

IoT – er kort for Internet of Things. Dette refererer til et nettverk av fysiske enheter som kommuniserer med hverandre. Det kan være alt fra biler til søppeldunker med sensorer som varsler hvor fulle de er.

JS – er kort for JavaScript. Dette er et programmerings-/skriptspråk som oftest brukes for å utføre operasjoner i en nettleser.

JSON – er kort for JavaScript Object Notation. JSON er et format for data som er mer leselig for mennesker enn andre formater. Det er ofte brukt for å sende forespørsler og motta data som respons.

LwM2M – er kort for Lightweight Machine to Machine. Dette er en protokoll som bygger på CoAP og brukes for å overføre data ofte brukt mot IoT-applikasjoner.

MQTT – er kort for Message Queuing Telemetry Transport. Dette er en protokoll for å overføre data ofte brukt mot IoT-applikasjoner.

NB-IoT – er kort for Narrowband IoT. Dette refererer til en spesifikk teknologi som har veldig god penetrasjonsevne med tanke på telenett. Denne har betydelig bedre dekning en 4G/5G nett.

Parsing – analysere og konvertere mellom tekst og datastruktur.

SCADA – er kort for Supervisory Control and Data Acquisition. Dette er ett datasystem som overvåker, styrer og samler inn data fra fysisk automasjonsutsyr.

Site – Zaphire site referer til et anlegg, ett eller flere bygg som har et styresystem.

Skyarkitektur – Dette referer til hvordan serverparker og software for «Skyløsninger» er satt sammen.

Skyløsninger – Dette er en måte å drifte dataressurser og gjøre de tilgjengelig over internett på en fleksibel og trygg måte.

Softwaremiljø – refererer til det dataområdet en softwarepakke «lever» i. Dette er ofte på en server og det er flere variasjoner. Det er hovedsakelig to miljøer, test/utvikling og produksjons miljø. Utviklingsmiljøet er ustabilt og endrer seg fort når utviklere tester ny software mens produksjonsmiljøet er stabilt og i orden. Produksjonsmiljøet er det som blir brukt av kunder.

UI – refererer til brukergrensesnitt (User Interface). Dette er noe som lar en bruker interagere med en applikasjon.

Innholdsfortegnelse

1 Bakgrunn for arbeidet

Dette kapittelet er en introduksjon til rapporten. Kapittelet tar for seg problemstillingen, målsettingen, struktur og tilnærming til oppgaven gitt av PowerTech. I kapittelet vil også forskningsspørsmålene bli spesifisert.

1.1 Problemstilling og forskningsspørsmål

Dagens 5G-teknologi tilrettelegger for stadig nye muligheter og fremskritt innen IT- og kommunikasjonsteknologi. I samarbeid med PowerTech Engineering, fra nå av kalt PowerTech, har det blitt undersøkt muligheter og laget løsninger for trådløs IoT ved hjelp av NB-IoT – en del av 5G utbyggelsen. [1] [2]

Hva må til for å koble Advantech WISE-4671 industriell IoT-enhet eller lignende til BMS/SCADA systemet til PowerTech? Hva kan gjøres for å få til dette på en skalerbar og datasikker måte?

1.2 Målsetning og avgrensning

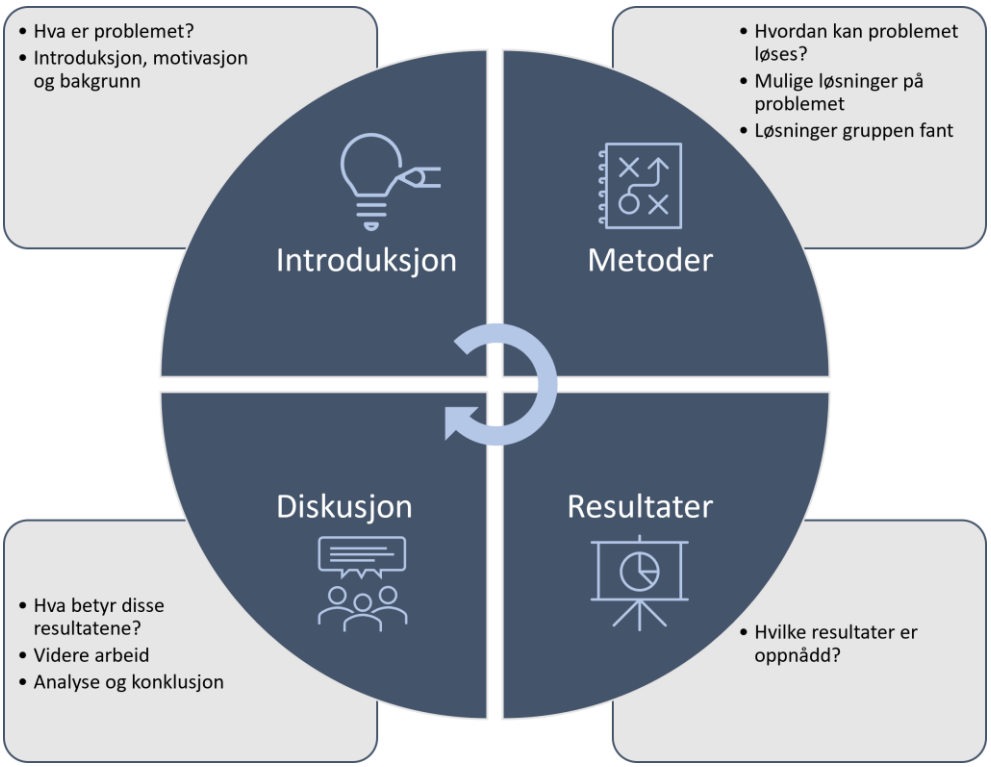
Målet med arbeidet er å undersøke dette feltet og forsøke å svare på disse forskningsspørsmålene på en ingeniørfaglig måte. I tillegg skal det lages en løsning for kommunikasjon mellom IoT-enheten og BMS/SCADA-systemet på en skalerbar og datasikker måte.

Arbeidet, herunder forskning og utvikling, praktisk gjennomføring og utarbeidelse av rapport, strekker seg fra januar til mai der det er avsatt en arbeidsmengde på 1620 timer (540 for tre personer).

Tilgjengelig utstyr og midler er i utgangspunktet Advantech WISE-4671 IoT-enheten med WISE-S614 IO modul montert og Zaphire BMS/SCADA-systemet.

1.3 Oppgavens struktur og oversikt over kapitlene

Strukturen til rapporten er basert på IMRaD-struktur. Akronymet beskriver en rapportstruktur i bruk blant annet til vitenskapelig forskning [3]. Figur 1.1 beskriver modellen som er lagt til grunn for rapporten. IMRaD-struktur gir gode rammer å forholde seg til for å få med de viktigste delene av en teknisk rapport. Det bidrar til å heve kvaliteten og å lage et godt sluttprodukt.



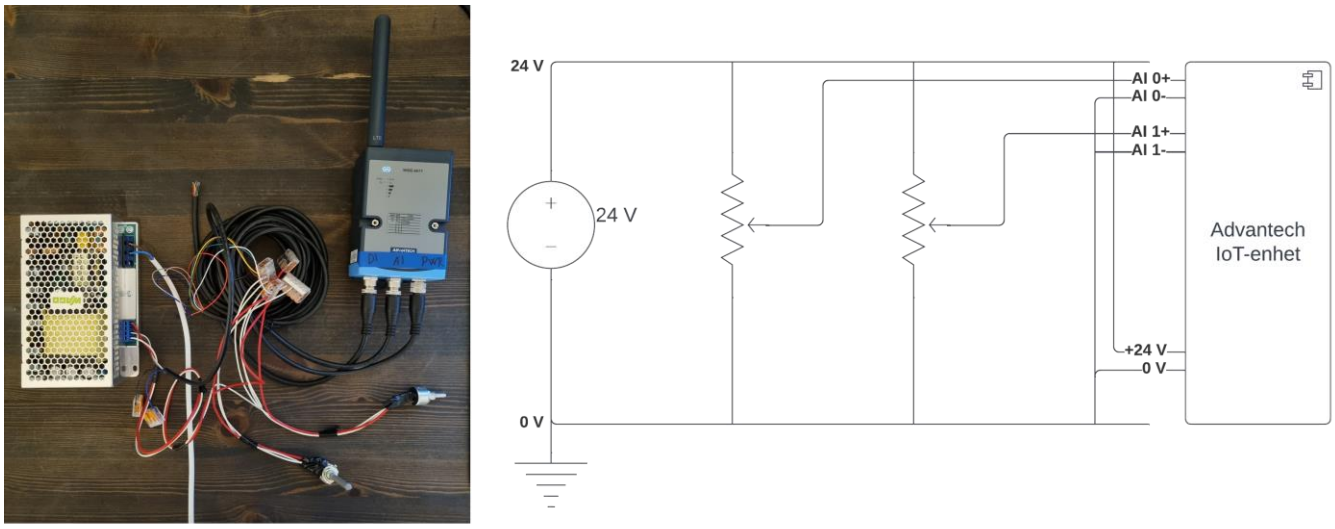
Figur 1.1: Illustrasjon og forklaring av rapportens struktur.

1.4 Tilnærming av problemstillingen

Samarbeidspartneren PowerTech la fram problemstillingen sammen med en idé om en enhet som kan samle inn måledata i felt ved bruk av NB-IoT teknologien. Eksempler på data kan være overvåkning av beholdning, nivå, klima, temperatur, lokasjon etc. Enheten bør være mest mulig kostnadseffektiv og kompakt. Det er en fordel om enheten har mulighet for batteridrift, med lengst mulig batterilevetid. [4]

Enheten bør også, ideelt sett, være så intuitiv og enkel at kunden selv kan installere og sette den i drift, uten behov for bistand. Nødvendig informasjonsmateriell må være inkludert (bruksanvisning/startveiledning). [4]

Den spesifikke enheten Advantech WISE-4671 med WISE-S614 IO modul montert, ble vurdert som passende til problemstillingen og ble anskaffet, gjennom PowerTech. Under oppsett og testing ble det koblet opp sammen med en 24V strømforsyning av WAGO og med to potensiometre for å generere analoge spenningssignal for testing som vist i Figur 1.2.



Figur 1.2: Oppkobling for test av IoT-enhet, strømforsyning og potentiometre.

1.5 Arbeidsmetode

Arbeidsstrategien til teamet har vært å først tilegne seg forståelse av problemstillingen, muligheter og inngående kunnskap på det aktuelle området. Deretter å komme raskt i gang med praktisk arbeid og teste teknologien. Gjennom tidligere erfaring og kunnskap om Scrum og Agile development, falt det naturlig med en iterativ arbeidsmetode. Fordelene her gjør at man raskt kommer i gang, ser teknologiens muligheter og begrensninger og setter seg mer oppnåelige delmål. Den iterative arbeidsmetoden passer også godt sammen med PowerTech sin arbeidsmetode.

2 Kravspesifikasjon for systemet

Ethvert system som lages skal gjøre en eller flere oppgaver og dekke behov. Dette er essensiell informasjon som utviklere og kunde må bli enige om. Kravspesifikasjon beskriver hva systemet skal gjøre [5]. Før og under utviklingen brukes kravspesifikasjonen til å planlegge og bestemme hvordan systemet skal lages. Dette inkluderer hvilke funksjoner og behov som skal implementeres. Etter utviklingen er kravspesifikasjonen for å sjekke i hvilken grad at sluttprodukt er som avtalt.

God kravspesifikasjon leder til færre feilaktige antagelser fra utviklerne [6]. Uten tilstrekkelig utredelse av kravspesifikasjonen kan en risikere konsekvenser som å overskrive budsjett og tidsfrist, eller kunden kan bli misfornøyd da produktet ikke innfrir forventningene [7].

2.1 PowerTech sine krav til sluttproduktet

Ved design av ethvert system er det visse krav til utforming og funksjonalitet. FURPS+ (functionality, usability, reliability, performance og supportability) er et begrep som brukes om spesifisering av krav for systemer og software-løsninger [8, p. 116]. Begrepet er et akronym som omfatter forskjellige aspekter innen funksjonelle og ikke-funksjonelle krav.

PowerTech som vi skulle utvikle systemet for ansees som kunde og stiller krav til løsningen. Bestillingen til PowerTech var kort og med lite spesifikke krav da vi skulle sørge for at data blir trådløst overført fra sensor til deres BMS/SCADA-system, Zaphire. Rammene for prosjektet var at NB-IoT skulle brukes, data skulle mottas i Zaphire og løsningen skulle være godt egnet for skalering.

2.1.1 Funksjonelle krav

Funksjonalitet og tjenester systemet skal ha og måten systemet forholder seg til bestemte hendelser er en del av systemets funksjonelle krav [9]. Her skal en trådløs NB-IoT enhet kunne konfigureres ferdig hos leverandør så den er konfigurasjonsløs for kunden. Denne enheten må ha batteri med lang levetid og mulighet til å koble på signal fra en eller flere sensorer. Data fra sensorene skal være tilgjengelig for kunden i Zaphire.

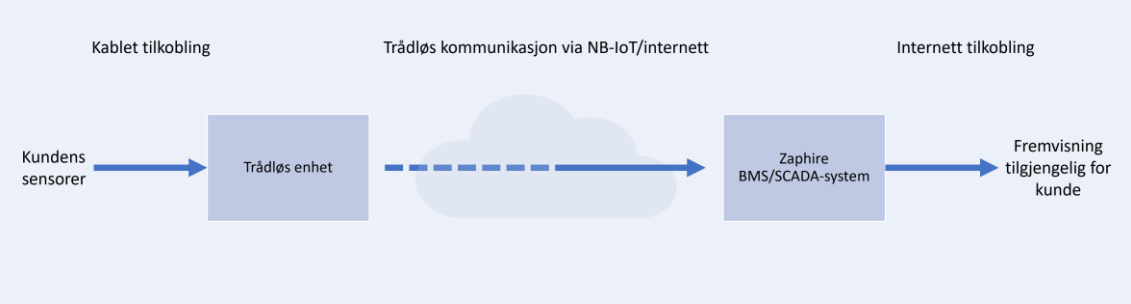
2.1.2 Ikke-funksjonelle krav

Ikke-funksjonelle krav er en beskrivelse av en observerbar karakteristikk, kvaliteter eller egenskaper til systemet [7]. Ettersom løsningen lages til et BMS/SCADA-system vil det være blant de ønskede kvalitetene til systemet at enheten skal støtte standard 4-20 mA signal. Overføring i sanntid og uten merkbare forsinkelser skal også være mulig med for at løsningen skal kunne benyttes i scenarioer som krever dette.

2.2 Systemdesign

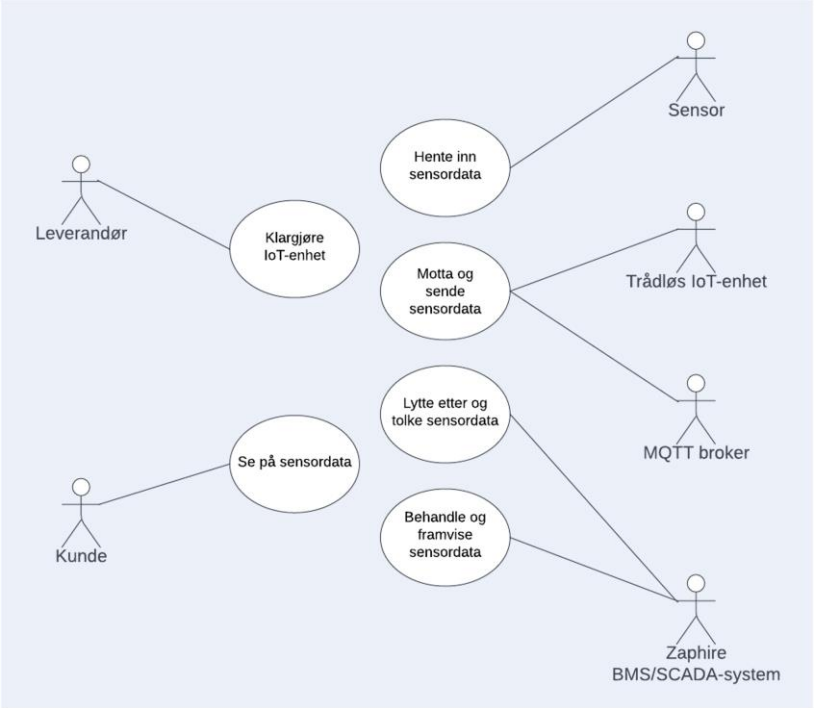
Systemet som skal lages omfatter da avgrensningen fra der sensorsignal kommer inn på trådløs enhet, til der sensordata kommer inn i Zaphire. Systemet er tiltenkt kunder som ønsker å ha sensorer eller liknende enheter til å kommunisere trådløst til BMS/SCADA-systemet Zaphire. Fra sensorene blir det kablet tilkobling til IoT-enheten som sørger for videre kommunikasjon. Gjennom NB-IoT reiser signal fra sensorene trådløst og ender til slutt i Zaphire der kunden vil ha tilgang til disse dataene.

Figur 2.1 viser en overordnet systemskisse for løsningen. Som følge av avgrensningene skal det ikke utvikles noe UI og frontend. Design av dette vil være opp til Powertech og påvirkes eventuelt av hver enkelt kundes ønske.



Figur 2.1: Skisse over systemet i løsningen.

For å få til løsningen i systemet kreves det en funksjoner og infrastruktur. Samspillet mellom dette er vist i Figur 2.2. Funksjonalitet vises som bobler og aktører som menneskefigurer. Dette kan være mennesker eller utstyr som en hardware- eller softwareenhet.



Figur 2.2: UML diagram for systemet med aktører og use case.

3 Oversikt over teknologier som ble tatt i bruk

Dette kapittelet inneholder kortfattet analyse og fundamental informasjon om teknologier som ble tatt i bruk i løsningen. Delkapitlene inneholder teknologiens opphav, hvordan den skalerer, hvilken grad av sikkerhet og hvilke nøkkelegenskaper den aktuelle teknologen har. Disse teknologiene blir tatt opp igjen og vurdert i kapittel 9 Diskusjon av IoT-løsningen.

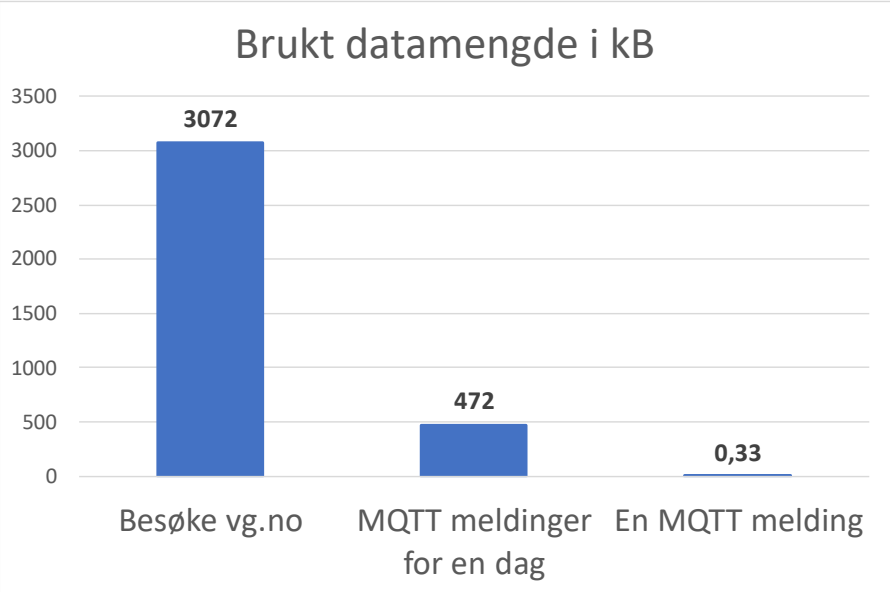
3.1 Introduksjon av NB-IoT

Overføring av måleverdier fra IoT-enheten til Zaphire foregår via Telias NB-IoT nett. Denne teknologien utvikler seg nå i Norden og Telia jobber med å sørge for dekning over store områder [10]. I Figur 3.1 viser grønnfargen områder på kartet i Norden som har dekning for NB-IoT.

Figur 3.1: Dekningskart for Telias NB-IoT i Norden. [10]

I dette prosjektet brukes Telias NB-IoT Starterkit som har en inkludert datamengde på 30 MB per SIM per måned [11]. Under utvikling av løsningen ble aldri datakvoten brukt opp. For å teste at SIM-kort var aktivert ble det puttet i en mobil og forsøkt å laste inn vg.no. Denne ene innlastningen av nettsiden brukte langt mer data enn noen annen enkelt operasjon gjort i løpet av utviklingen.

Til sammenlikning viser Figur 3.2 ulike mengder databruk. Under testing hos kunde med et sendingsintervall på 60 sekunder mellom hver melding, ble brukt datamengde under 500 kB per dag. Meldingene som sendes kan optimaliseres videre ved å kutte mer i JSON-strengen som sendes. Sett ut ifra hvilke datamengder IoT-enheten overfører er datakvoten på 30 MB derfor stor nok. Og starterkittet ansees som en god løsning for å komme i gang med NB-IoT.



Figur 3.2: Sammenlikning av databruk for ulike scenario.

I juni 2016 fastslo det globale initiativet 3GPP standardiseringen av NB-IoT (Narrowband Internet of Things). Standarden ble fastslått i deres utgivelse kalt «Release 13» med hastighet på opptil 27 kbps og tilpasset bruk av IoT-enheter, som fremkommer av navnet og teknologiens egenskaper. Denne har blitt videreutviklet gjennom flere utgivelser med tanke på fleksibilitet, strømforbruk, forsinkelse, støtte for 5G, hastighet og rekkevidde. [12]

Antennemaster for NB-IoT nettet oppføres nå ved alle Telias 5G-master. Teknologien er designet særlig for å kunne håndtere et stort antall enheter og tetthet mellom disse. Med god gjennomtrengningsevne og god rekkevidde, også tatt i betraktning, vurderes skalerbarheten til NB-IoT derfor som god. [13]

I følge PowerTech er det ikke kjent at denne teknologien er i bruk i stor grad i det norske marked [4]. Derfor sees det på som en god mulighet til å utvide bruksområder for IoT til fjerntliggende destinasjoner.

«Massive Machine Type» er en ny standard for kommunikasjon i IoT som kommer med 5G-teknologien. Standardiseringen sørger for økt sikkerhet og består av NB-IoT og LTE-M, som er en liknende teknologi der krav for hastighet er større [10].

For å være godt egnet til IoT-bruk kreves det en rekke egenskaper. NB-IoT er særlig godt egnet på grunn av god dekning innendørs, støtte for et stort antall av lavhastighetsenheter, lavt strømforbruk, lav sensitivitet for støy og gjerne meget lav kostnad per enhet [14]. Ifølge teleselskapet Telia, har NB-IoT god nok gjennomtrengningsevne til å fungere uten problemer for enheter plassert under bakken, 80 meter ned i en militærbunker [13]. I det samme forsøket mistet en mobiltelefon dekning etter 20 meter til sammenlikning.

Smarte parkeringshus, værstasjoner og pH-overvåkere til landbruk er eksempler på noen av de mange bruksområdene for NB-IoT. Dette er tilfeller der data overføres gjerne i små mengder, mellom lange distanser og med krav til lang batterilevetid. Teknologien kan derfor være med på å muliggjøre løsninger og markeder for IoT som ikke var der tidligere. [15] [16]

Frekvensbånd som brukes til NB-IoT i Norge er 800 MHz og 1800 MHz [13]. Sammenliknet med 4G og 5G til en vanlig mobiltelefon er dette relativt lavfrekvent. Standard, offentlig mobilkommunikasjon bruker ulike frekvensbånd fra 450 MHz og opp [17].

Bølgeformelen (3.1) der λ, v og f er henholdsvis bølgelengde, bølgehastighet og frekvens viser forholdet mellom disse [18]. Med en lav frekvens blir da bølgelengden lang, som gjør at signalet bærer over en lengre distanse [19]. Med dette kan det vises at NB-IoT nettet har god rekkevidde sammenlignet med en mobiltelefon.

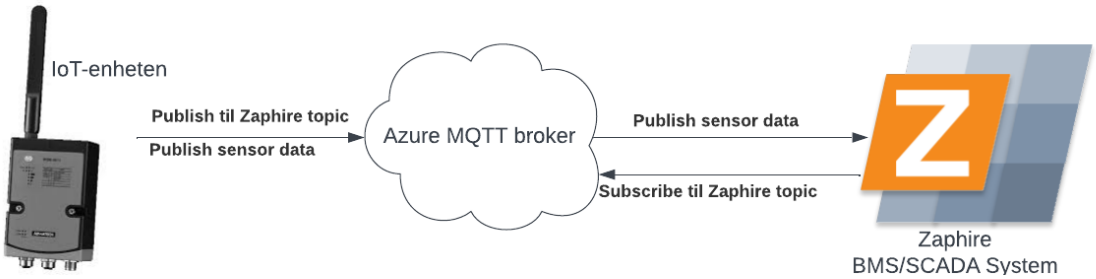
$$\lambda = \frac{v}{f} \tag{3.1}$$

3.2 Introduksjon av MQTT

Dette kapittelet vil ta for seg MQTT som en kommunikasjonsprotokoll med fokus på bruk mot IoT. MQTT er en relativt gammel teknologi, men fortsatt veldig relevant å ta opp. Her vil det bli tatt opp hvor, når og av hvem MQTT ble laget. Det blir også tatt opp hvordan det skalerer, hvilken grad av sikkerhet det bruker og hvilke kjerneegenskaper det har.

MQTT ble utviklet av Dr. Andy Stanford-Clark ved IBM og Arlen Nipper av Arcom (nå Eurotech) i 1999 [20]. Det ble utviklet som en simpel og lett meldingsprotokoll bygget for lav båndbredde og ustabile nettverk. Disse kvalitetene gjør det ypperlig for IoT-kommunikasjon.

MQTT baserer seg på TCP og trenger en kontinuerlig forbindelse for å kommunisere. Dette gjør at det ikke kan skaleres til det uendelige [21]. På grunn av publiser/abonner arkitekturen, som illustrert ved Figur 3.3, skalerer det bedre enn tradisjonelle client-server oppkobling [22]. Dette er fordi kommunikasjonen kan gjøres i stor grad parallelt og hendelsesbasert. Det er likevel ikke praktisk å skalere til millioner av enheter per broker.



Figur 3.3: Illustrasjon av kommunikasjon mellom IoT-enhet og Zaphire via MQTT.

Det er støtte for brukernavn og passord i versjon 3.1 av protokollen. Det er også støtte for SSL over nettverk [20]; dog SSL er relativt tungt å kjøre over nettet og bruker en del båndbredde. Krypterte meldinger kan også sendes over MQTT selv om dette ikke er bygd inn i protokollen.

De viktigste egenskapene til MQTT er at publiseringsklienten og abonneringsklientene trenger ikke kjenne til hverandre, klientene trenger ikke å kjøre på samme tid, de trenger ikke å avbryte sitt arbeid under publisering og mottaking av meldinger og det støtter filter for emne og format som gjør det enkelt å sortere meldinger som er relevante for enhetene som abonnerer. [22]

3.3 Introduksjon av Zaphire og mikrotjenestearkitektur

Dette kapittelet vil handle om Zaphire, men vil også ta for seg litt om mikrotjenestearkitektur. Dette er fordi Zaphire er bygget opp av mikrotjenester det kan derfor være greit å gå gjennom det først. Deretter blir fokuset på Zaphire som er et skybasert BMS/SCADA-system.

3.3.1 Kort om mikrotjenestearkitektur

Mikrotjenestearkitektur er mer en filosofi enn det er en teknologi. Ofte bare kalt mikrotjenester, handler det mest om hvordan en applikasjon er bygget opp. Når noe bruker mikrotjenester betyr det at det er bygget opp av mange små deler eller programmer som til sammen danner en respons til en forespørsel. Disse mikrotjenestene er ofte bygget opp av containere som er små softwaremiljøer med en simpel applikasjon inni. Containerne blir enkelt dannet eller fjernet av et overordnet system som Kubernetes som blir dekket i kapittel 3.4 Introduksjon av Kubernetes. Disse kan også kommunisere med hverandre, dele last og/eller videresende oppgaver til hverandre. Egenskaper som dette gjør at mikrotjenestearkitektur kan skalere veldig godt. [23]

3.3.2 Introduksjon av Zaphire

Zaphire er et produkt utviklet av PowerTech som erstatning for dagens BMS/SCADA-systemer. Siden oppgaven er gitt av PowerTech og den handler om å utvide funksjonaliteten til Zaphire er det naturlig å ha med litt om hvor det kommer fra, hvordan det skalerer, sikkerheten brukt og kjerneegenskapene.

Det skybaserte BMS/SCADA-systemet Zaphire er utviklet av PowerTech for å redusere installasjon- og vedlikeholdskostnadene som er store for mer tradisjonelle styringssystemer. Ved å putte systemet i skyen gjør det at kunden ikke trenger å vedlikeholde en server og holde den oppdatert. En av de største fordelene PowerTech har oppnådd er evne til å skalere og oppdatere systemet fort og effektivt. [4] [24]

Zaphire er bygget på Kubernetes, mikrotjenester og containerapplikasjoner. De bruker flere containerapplikasjoner for å håndtere lastbalansering, blant annet, for webserver som mottar kundenes oppkoblingsforespørsler og fordeler de seg imellom. Ved å kjøre flere applikasjoner i parallell oppnår de høy oppetid og lett oppgradering av softwaren. Når en oppdatering skal rulles ut kjøres en ny instans av den oppdaterte softwaren opp og avslutter en eldre instans. Systemet blir derfor lett oppdatert uten noe nedetid. Modulariteten gjør at det skalerer spesielt godt, siden det alltid kan legges til en ekstra server eller applikasjon for å øke kapasiteten.
[4] [24]

Zaphire bruker en oppkoblingsmetode som gjør at kundens sikkerhet er godt ivaretatt. Dette gjøres ved at kunden kobler seg opp med utadgående trafikk mot Zaphire; siden det er kunden som kobler seg opp trenger de ikke åpne en port eller andre sikkerhetshull mot internett på sine maskiner. Zaphire sitter derfor med all risikoen for angrep hos seg. Dette gjør det lettere å ha kontroll på trusler. [24]

Zaphire er en veldig fleksibel plattform som lar kunden bygge opp et grensesnitt som passer deres behov. Den tillater oppkobling direkte mot PLS, MQTT og Rest API for andre IoT-enheter. Dette gjør at den er lett å utvide med mange forskjellige enheter og systemer for styring og overvåkning av bygninger og andre systemer. [4]

3.4 Introduksjon av Kubernetes

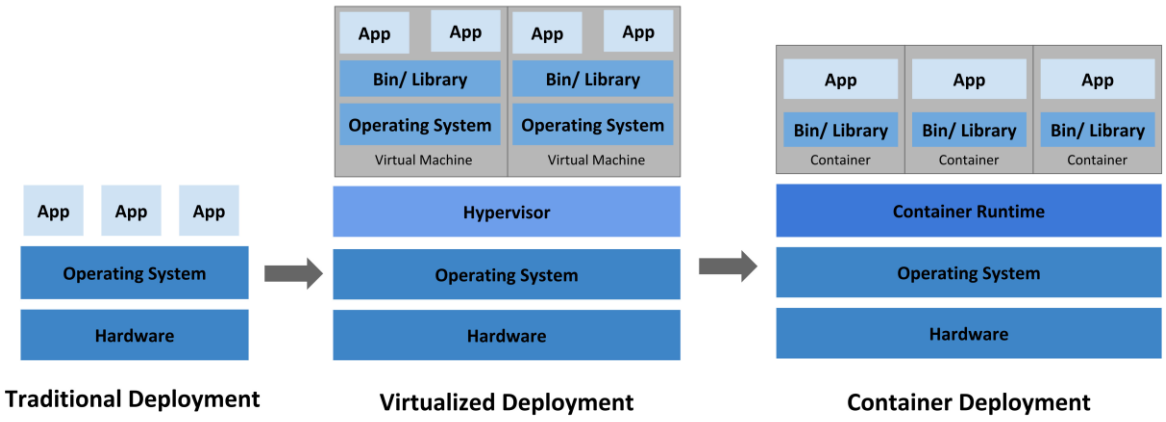
Softwareløsninger til et selskap som PowerTech består gjerne av flere applikasjoner som kjører på en server og krever en viss mengde datakraft eller ressurser. Oppetid, sikkerhet og skalering er viktige faktorer for en slik bedrift. I dette kapittelet forklares hva Kubernetes er, og hvilke fordeler det har for bruk i slike softwareløsninger.

Kubernetes er et rammeverk for drift av containere til applikasjoner. Rammeverket har åpen kildekode og mange fordeler for drift av softwareløsninger [25]. Det tradisjonelle og simpleste oppsettet å kjøre applikasjoner på er å kjøre på en datamaskin direkte i operativsystemet (OS-et). Dette er vist til venstre i Figur 3.4. [26]

En av bakdelene med å kjøre applikasjoner direkte i OS-et er at ressursene er rent fysisk begrenset til hva den maskinen har, fordelt på alle applikasjonene. Dagens maskinvare begrenser også hvor kraftig en enkelt maskin kan være. Disse begrensningene kan medføre at en maskin ikke klarer å levere ressurser nok til de applikasjonene som kjører. Dette kan føre til at applikasjoner krasjer eller verst tenkelig konsekvens, at hele maskinen krasjer og alle applikasjonene går ned. [26]

Siden disse applikasjonene kan ha veldig varierende ressursbehov blir det fort vanskelig å dimensjonere en maskin til å dekke behovene. For å ha nok overskudd til økende last på en eller flere applikasjoner vil det måtte brukes mange maskiner. Når det økte behovet ikke er der vil disse kjøre på for eksempel 20% av ytelsen og resten av kapasiteten er sløs og kostnad som ikke blir utnyttet. [27]

Ved å adskille applikasjonene til virtuelle maskiner (VM-er) som kjører på serveren, lager man en begrensning på hvor mye ressurser som kan tas i bruk av applikasjonen. På denne måten påvirker ikke en applikasjon alle de andre. Utviklingen av dette førte til at man begynte å bruke containere som kjører hver sin applikasjon slik at disse er adskilt. Denne måten å kjøre applikasjonene på gjør operasjonen mer effektiv og fleksibel. [26]



Figur 3.4: Oversikt for utvikling av systemer. [26]

En container er et datamiljø som kjører på en server og får tildelt en ønskelig andel av ressursene til hele serveren [26]. Denne containeren har et eget filsystem og kjører gjerne bare en applikasjon, som sørger for at applikasjonen ikke stjeler ressurser [25]. Dette kan minne om en VM, men OS-et ligger ikke inne i containeren slik som i en VM. Flere containere kan kjøre på samme OS som vist i Figur 3.4.

En MQTT broker er et eksempel på en applikasjon som kan kjøres i egen container. Nødvendige filer, standard konfigurasjon m.m. kan lages til et containerimage. Se på dette som en oppskrift som brukes til å lage en lik container [28]. Ved problemer eller krasj av MQTT brokeren startes en ny container ved hjelp av oppskriften og systemet er straks fungerende igjen. Dette kan oppdages og fikses automatisk i Kubernetes. [26]

PowerTech sine containere kjører i Microsoft Azure der serverressurser leies og enkelt kan utvides ved endring i ressursbehov [4]. Fordeling av applikasjoner ved hjelp av containere i Kubernetes fører til tettere ressursutnyttelse av hardwaren, da containere dynamisk tildeles ønsket mengde ressurser [28]. Denne måten forbedrer ressurstetthet og effektiv utnyttelse av den fysiske hardwaren.

Med hver applikasjon kjørt i egen container er disse isolert fra hverandre og får ikke automatisk tilgang til andres filer [28]. I situasjoner der containere trenger å ha tilgang på samme filer kan de få tilgang til felles filområde. I praksis kan containere brukes til å forsterke datasikkerheten betraktelig. Ved at applikasjoner til forskjellige kunder har forskjellige containere vil ikke disse har mulighet til å berøre hverandres filer eller minne. [26]

4 Konfigurasjon for kommunikasjon mellom IoT-enhet og styresystemet Zaphire

Dette kapittelet tar for seg implementasjon av IoT-enheten opp mot BMS/SCADA-systemet Zaphire gjennom NB-IoT. Valgt løsning på protokoll for data innsamling er MQTT da Zaphire har delvis støtte for dette.

4.1 Konfigurasjonsløs tilknytning

Det er ønskelig at sluttkunde skal måtte gjøre minst mulig for å få enheten opp og gå på sin ende. Det er derfor tenkt en konfigurasjonsløs tilknytning fra enhet til Zaphire. Dette er en problemstilling som må avgrenses noe siden det må konfigureres for å få koblet til noe. Det som er ønskelig er at kunden skal slippe og derfor er det naturlig at enten PowerTech eller produsenten av enheten tar denne oppgaven. Det som er foreslått er å lage en konfigurasjonsfil som kan lastes opp til enheten på en enkel og effektiv måte. Det som vil mangle for den enheten da er broker adressen. Dette kan løses med et skript som fyller inn dette feltet i konfigurasjonen eller så kan dette fylles inn manuelt. I dette senarioet trenger ikke kunden gjøre annet enn å motta enheten koble den til signalene sine og strøm. Det må også avklares om det skal tas i bruk strøm- eller spenningssignaler siden dette krever en fysisk endring av enheten ved å flippe DIP-switcher.

4.2 Oppsett av MQTT broker

Prosjektet trengte en måte å koble sammen IoT-enheten og BMS/SCADA-systemet. Det ble valgt å ta i bruk MQTT for dette. For å kunne kommunisere mellom Zaphire og enheten, via MQTT, trengs da en broker. Det som er ønsket, er å ha en broker per kunde eller site. Derfor må det være lettvint og dynamisk skalering av brokere. Dette kan oppnås ved bruk av container-applikasjoner som kjøres en per site. For å bruke containerapplikasjoner på en god måte trengs det infrastruktur som for eksempel Kubernetes for å styre hvilke brokere som kjører, hvilke brokere som har stoppet eller krasjet og om en ny site trenger en broker.

I dette senarioet er det en Mosquitto broker og de to klientene Zaphire og IoT-enheten som ble lagt til grunn. Denne brokeren er satt opp som en container bygget fra et image. Får å produsere dette imaget og få det tilgjengelig i Azure ble en guide tatt i bruk [29]. Denne guiden gjennomgår hvordan man bygger et image av Mosquitto og tilgjengeliggjør det i Docker-miljøet. Deretter hvordan sette opp Azure med lagringsområder og hvordan man lager en instans av imaget så det kan kjøres. Dette gjør løsningen skalerbar ved at brokerne kan dynamisk og automatisk bli produsert opp, startet, stoppet og slettet etter behov. Det guiden ikke gikk inn på er konfigurasjonsfilene til brokeren, dette er et forbedringspotensial.

Kubernetes er det naturlige valget for å regulere dette når alt er bygget opp. Med det arbeidet som er gjort så vil det enkelt kunne startes, stoppes, lages og fjernes brokere etter behov. Alt dette kan Kubernetes ta ansvar for å holde vedlike.

4.3 Konfigurering av IoT-enhet for kommunikasjon over MQTT via NB-IoT

Førstegangsoppsett av Advantech WISE-4671 IoT-enheten består av oppkobling og konfigurasjon av IoT-enheten med Telia SIM-kort. Deretter koble den opp mot MQTT broker for å sende meldinger.

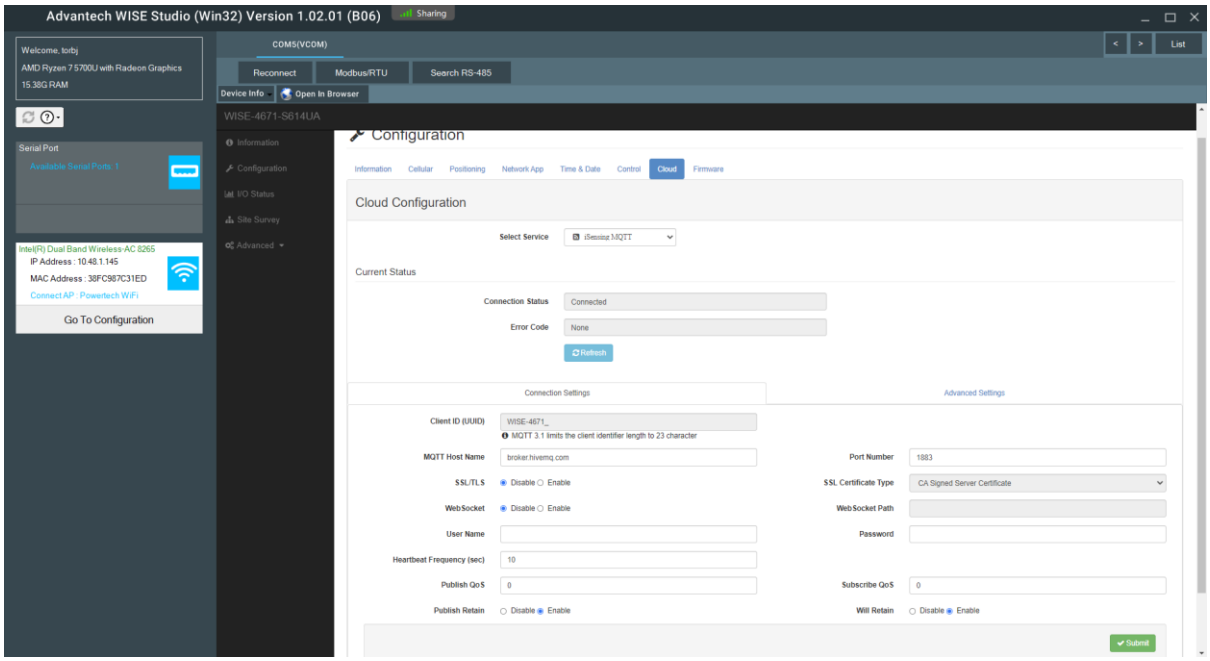
4.3.1 Konfigurering for NB-IoT

Til dette prosjektet ble det brukt et SIM-kort fra NB-IoT Starterkit levert av Telia [30]. Dette er et abonnement som er designet for prototyping. Oppsett av dette ble å konfigurere selve enheten via Advantech WISE Studio [31]. I WISE Studio ble det gjort noen få endringer, APN ble satt til «lpwa.telia.iot», frekvensbånd «20» ble valgt og pin «0000» ble tastet inn.

4.3.2 Konfigurering for MQTT

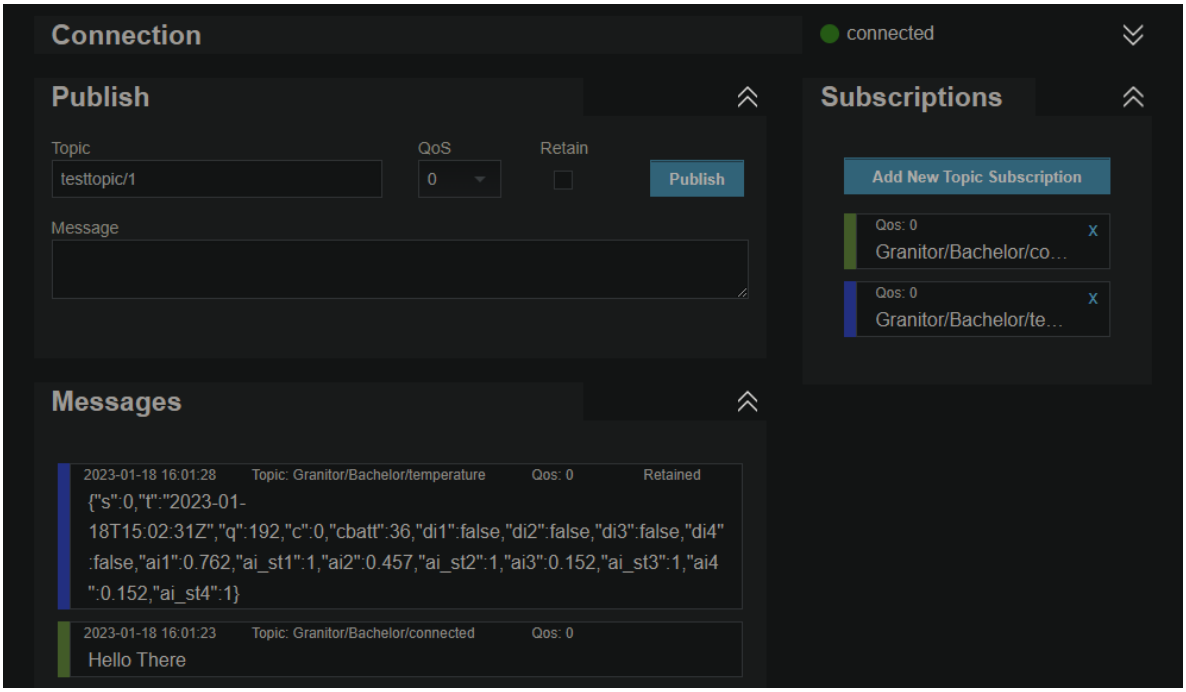
Konfigurering av enheten ble gjort over USB fra PC til enheten via Advantech WISE Studio vist på Figur 4.1. Først ble det valgt å bruke MQTT som tjeneste, «broker.hivemq.com» ble satt som broker, SSL/TLS samt WebSocket ble satt til «disabled» og portnummeret ble satt til «1883». Disse innstilingene kom fra HiveMQ sine sider om sin egen broker [32].

For å enkelt kunne verifisere at enheten sender og hvilket format den sender data på ble den først koblet til en testbroker levert av HiveMQ [32].



Figur 4.1: Konfigureringsside for MQTT i Advantech WISE Studio.

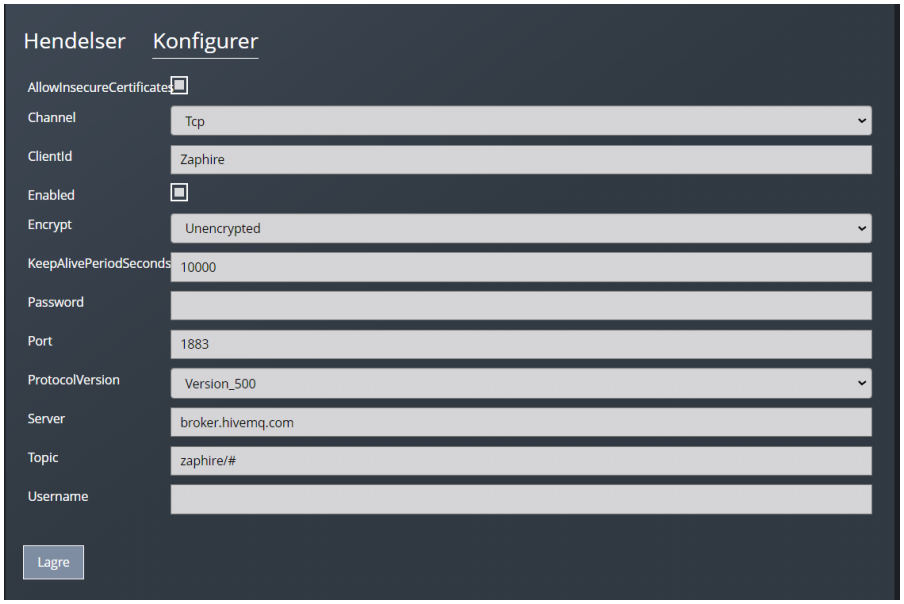
Da MQTT fungerer slik at du må ha en klient som abonnerer på topicen som enheten sender på for å se hva den sender ble en virtuell webklient levert av HiveMQ [33] brukt som klient nummer to. Med dette kan man se hva enheten sender og strukturen på dataen. Figur 4.2 viser dataen som kommer inn fra enheten som er i JSON format.



Figur 4.2: HiveMQ webklient som mottar test data fra enheten.

4.4 Konfigurering av styresystemet Zaphire for MQTT kommunikasjon

Zaphire har delvis støtte for MQTT. For nå har den kun støtte for å koble til en broker per site. Tilkobling til broker er ganske rett fram da det bare er å fylle ut brokerinformasjon som vist i Figur 4.3.



Figur 4.3: Konfigureringsside for MQTT i Zaphire.

Zaphire har ikke noe innebygd funksjonalitet for å framvise eller parse data i JSON format. Dataene må derfor først bli sendt til et JavaScript for å bli parset så de kan skrives inn i taglisten i Zaphire som vist i Figur 4.4.

Figur 4.4: Dataflyt omforming JSON til tagliste.

Etter å ha analysert innholdet i meldingen ble det konkludert at et generelt dynamisk skript er nødvendig for å konvertere fra JSON til taglisten som Zaphire bruker for å lese data. Et slikt skript er vist i Figur 4.5 der sensorverdier blir fylt inn i taglisten. Resultatet vises i Figur 4.6.

```
1   var topic = "zaphire/bachelor/#";
2
3   Zaphire.MQTT.RegisterHandler(
4   {
5       Topic: topic,
6       OnMessage: function(message)
7       {
8           var parsedMessage = JSON.parse(message.Payload)
9
10          for (const [tag, value] of Object.entries(parsedMe
11          {
12              Tag.WriteValue(tag,value);
13          }
14      }
15  });
```

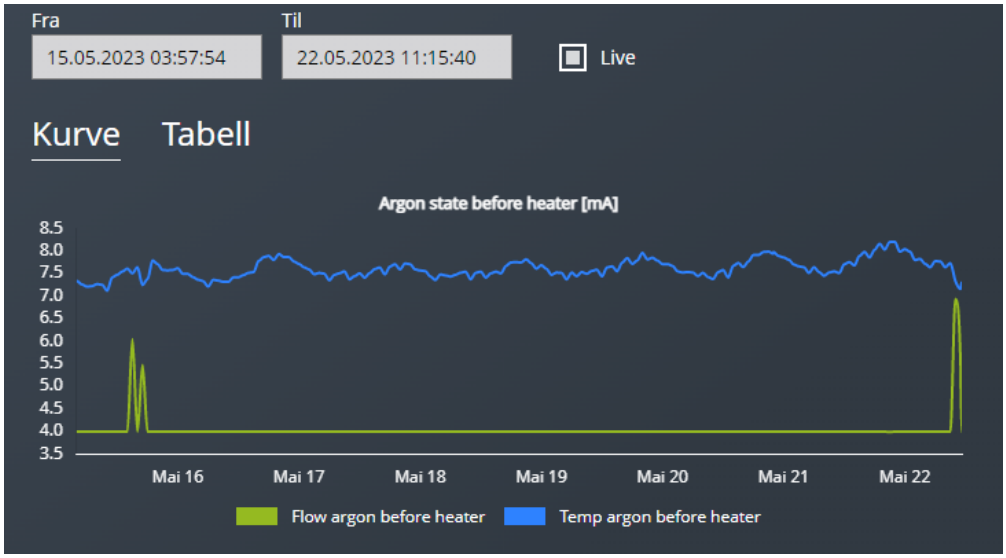
Figur 4.5: JavaScript parsing av JSON for utfylling av tags.

cbatt	Double		99,0	...
di1	Boolean		False	...
di2	Boolean		False	...
ai1	Double		4,0	...
ai4	Double		8,0	...
di3	Boolean		False	...
ai2	Double		7,5	...
ai3	Double		5,6	...
ai1_title	String			...
di3	Boolean		False	...
di4	Boolean		False	...
Time	String		0	...

Figur 4.6: Tagliste i Zaphire.

4.5 Framstilling av innkommende data

Selv om Zaphire mottar data blir det ikke visualisert av seg selv. Dette blir gjort ved hjelp av koden som vises i Figur 4.5. I dette senarioet kommer signalene inn i milliampere som vises i Figur 4.7. Dette er ikke umiddelbart nyttig og burde bli omregnet med en algoritme som eksempelvis den i Figur 4.8 som kan omforme fra milliampere til en mer informativ måleenhet gitt at grensene defineres.



Figur 4.7: Linjediagram over innsendte data fra Advantech enhet.

```python
def Convert(value, upperboundInput, lowerboundInput, upperboundOutput, lowerboundOutput):
    inputDiff = upperboundInput - lowerboundInput
    outputDiff = upperboundOutput - lowerboundOutput

    percentInput = (value-lowerboundInput)/inputDiff
    return percentInput*outputDiff + lowerboundOutput

upperRange = 20
lowerRange = 4
upperOutput = 5
lowerOutput = 2

for i in range(lowerRange,upperRange + 1):
    print(Convert(i,upperRange,lowerRange,upperOutput,lowerOutput))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Triny> & C:/Users/Triny/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Triny/Documents/Convert.py
2.0
2.1875
2.375
2.5625
2.75
2.9375
3.125
3.3125
3.5
3.6875
3.875
4.0625
4.25
4.4375
4.625
4.8125
5.0
```

Figur 4.8: Kodeeksempel for omregning fra skalaen til milliampere signaler til annen skala.

4.6 Konfigurering av IoT-enhet for bruk i prosessanlegg

Etter det første systemet er utviklet stod konfigurering og montering hos kunde for tur. Dette ble gjort hos Norsk Titanium. Av hensyn til datasikkerhet stilte Norsk Titanium spørsmål til kryptering. Under utvikling ble det testet å benytte kryptering, men uten suksess. Norsk Titanium foretrakk å ha kryptering, så det ble forsket og testet videre for å finne ut av hvorfor dette ikke funket. Mer informasjon om dette i kapittel 6.3 Testing av IoT-enhet i felt.

Det er ikke ønskelig at informasjon kommer på avveie, som er en risiko når kommunikasjonen ikke er kryptert. Sammen med Norsk Titanium ble det likevel valgt å gå videre ved å bruke privat broker uten kryptering. Ved å bruke en privat broker må de som ønsker å stjele data vite om at kommunikasjonen foregår og enten lytte etter pakker med data og analysere de. Et annet alternativ vil være å koble seg til brokeren direkte hvis man vet serveradressen. Ved å koble seg på brokeren vil det bli loggført, noe som kan brukes for å identifisere aktøren.

Siden det er en risiko for at informasjon kan komme på avveie. Ble det tatt en vurdering på hvilke måleverdier som ble koblet til IoT-enheten. Disse burde ikke være spesielt konfidensielle eller ha store konsekvenser hvis de kommer på avveie.

5 Oppkobling og testing av IoT-enhet i prosessanlegg

Dette kapittelet beskriver anlegget der installasjonen ble gjort. Hvordan den ble gjennomført og hvilke problemer som dukket opp underveis. Det blir også gjennomgått hva som ble gjort for å løse problematikken.

IoT-enheten Advantech WISE-4671 har innebygd batteri med mulighet for ladning både ved vanlig 24V, som er det som ble benyttet i prosjektet, og solceller. Den har også et utbyttbart kort for forskjellige I/O konfigurasjoner. I/O-modulen som er brukt i dette prosjektet er en type med 4 analoge og 4 digitale innganger.

5.1 Om anlegget til montering av IoT-enhet

Norsk Titanium i Hønefoss driver med additiv konstruksjon av titandeler til for eksempel strukturelle komponenter i fly. Dette gjøres gjennom det selskapet kaller en revolusjonerende additiv fabrikkeringsteknologi. Den patenterte teknologien kalles Rapid plasma deposition (RPD). [34]

Skapet som skulle kobles i inneholdt ventilasjonskobling og -styring for hallen og de interessante signallederne. Det ble koblet i serie med de eksisterende systemene som mottok signalene. Dette fordi signalene er strømsignaler mellom fire og tjue milliampere.

5.2 Oppkobling og montering av IoT-enhet

Dette kapittelet tar for seg den fysiske oppkoblingen hos Norsk Titanium som vist i Figur 5.1. Det inneholder både det rent praktiske for å koble seg på, men også en feilsituasjon som dukket opp underveis. Feilen ble til slutt rettet, noe som førte til en suksessfylt installasjon.

Figur 5.1: Fysisk oppkobling i prosessanlegget hos Norsk Titanium.

5.2.1 Fysisk tilkobling til IoT-enhet

Som vist i Figur 5.2 har IoT-enheten store tilkoblingsmuligheter. For den installasjonen som ble gjort ble det bare tatt i bruk de analoge inngangene og forsyningsstrøm til enheten. Tilkobling ble gjort tilsvarende det vist i Figur 5.3, men for alle 4 analoge innganger. For at dette skal virke er det kritisk at DIP-switchene står i riktig posisjon. De er derfor flippet til strømsignal siden det er det som kommer fra sensorene i fabrikken.

Pin Assignment of WISE-S600 I/O Module with M12 Connectors.

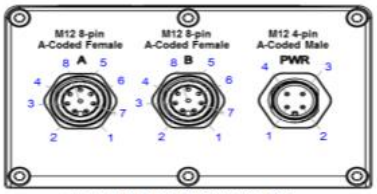
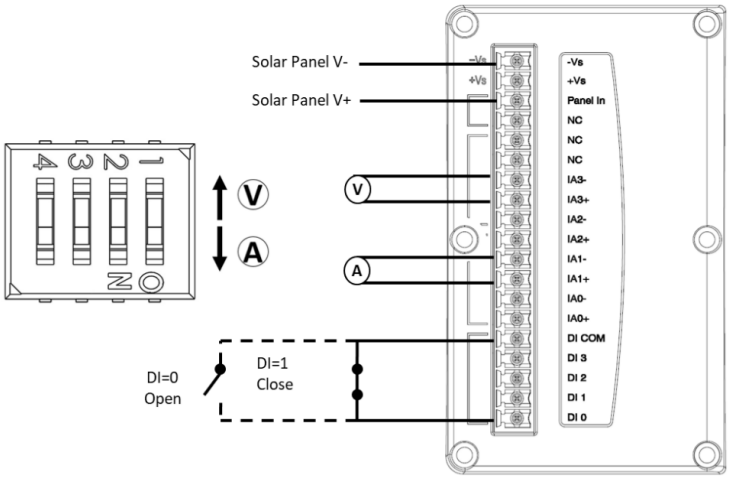


Figure 3.1 WISE-S600 Pin Out

	Model Name / Pin Number	M12 Cable	WISE-S614	WISE-S615	WISE-S617	WISE-S672
	P/N	4Pin: 1700028162-01 8Pin: 1700028163-01	WISE-S614-A	WISE-S615-A	WISE-S617-A	WISE-S672-A
A	1	White	DI0	RTD2+	IA0+	DI0
	2	Brown	DI1	RTD2-	IA0-	DI1
	3	Green	DI2	RTD2 COM	+12V_Out0	DI2
	4	Yellow	DI3	NC	+12V_Out_GND	DI3
	5	Gray	NC	RTD3+	IA1+	DI4
	6	Pink	NC	RTD3-	IA1-	DI5
	7	Blue	NC	RTD3 COM	+12V_Out1	NC
	8	Red	DI COM	NC	+12V_Out_GND	DI_COM
B	1	White	IA0+	RTD0+	DI0	DATA 0-
	2	Brown	IA0-	RTD0-	DI1	DATA 0+
	3	Green	IA1+	RTD0 COM	DI_COM	RS-232 TX
	4	Yellow	IA1-	NC	DO0	RS-232 RX
	5	Gray	IA2+	RTD1+	DO_GND	DATA 1-
	6	Pink	IA2-	RTD1-	RS-485 D+	DATA 1+
	7	Blue	IA3+	RTD1 COM	RS-485 D-	NC
	8	Red	IA3-	NC	RS-485 GND	RS-232 GND
PWR	1	Brown	+VS	+VS	+VS	+VS
	2	White	-VS	-VS	-VS	-VS/ SP-
	3	Blue	SP+	SP+	SP+	SP+
	4	Black	SP-	SP-	SP-	NC

Figur 5.2: I/O-tabell til Advantech WISE-S614. [35]



Figur 5.3: Koblingsskjema for I/O modul WISE-S614T. [35]

5.2.2 Feil i avlesning av signaler

Originalt tenkt var det bare å koble seg inn i skapet. Dette ble testet og resulterte i veldig merkelige verdier som traff mer enn en kanal på enheten. Feilmålingen var ganske uforutsett siden enheten hadde vært testet på forhånd med både digitale og analoge verdier. Det viste seg at IoT-enheten ikke hadde vært testet med strømsignaler, noe som var kritisk siden signalene i skapet er på 4-20 mA. Dette førte til mye feilsøking, testing og frustrasjon. Når signalet blir byttet fra spenning til strøm må det flippes noen DIP-switcher inni I/O modulen på enheten. Dette kom ikke frem i softwaren som en ting å huske på når denne innstillingen ble endret. Det var heller ikke skriftlig i leverandørens dokumentasjon [35] til enheten.

Etter dette ble klart ble enheten åpnet for å flippe switchene og alt ble skrudd sammen igjen. Dette løste problematikken og enheten ble installert etter original plan. Så lite skal til for å skape store problemer. Etter å ha installert enheten ble den testet og bekreftet at dataene kom frem til Zaphire som var en stor suksess.

6 Testing av systemet

Testing for å undersøke og sikre seg at systemet virker etter sin hensikt bør være en sentral del av ethvert softwareutviklingsprosjekt. Hensikten med denne fasen i prosjektet er i hovedsak å luke ut bugs [36]. I tillegg til dette kan det hjelpe å oppdage stabilitetsproblemer, problemer ved skalering og sjekke om systemet oppfyller punktene i kravspesifikasjonen.

6.1 Testing av IoT-enhet under utvikling

Ettersom arbeidsmetoden i prosjektet er basert på Scrum og Agile Development, faller det naturlig å teste underveis i utviklingen. Det iterative arbeidet legger til rette for kontinuerlig unit testing og integrasjonstesting. [37]. Likevel skal det lite til før systemet blir så komplekst at det krever bredere testing.

Fra start til slutt i utviklingsarbeidet ble ny og revidert funksjonalitet testet. Til å begynne med handlet det om å konfigurere IoT-enheten til å kommunisere med MQTT broker, så etterfulgt av testing. Deretter få Zaphire til å motta data fra den samme brokeren og teste igjen. Løsningen ble videreutviklet og påbygget mer funksjonalitet.

Ved denne kontinuerlige testingen ble ofte systemet testet i sin helhet, noe som sikret at oppførselen til systemet var som forventet. I forbindelse med installasjonen hos Norsk Titanium ble det gjort acceptance testing. Factory Acceptance Testing (FAT) gjort av utviklingsteamet før installasjon, og Site Acceptance Testing (SAT) gjort i etterkant av installasjon.

6.2 Konfigurering av IoT-enhet for testing i felt

Norsk Titanium ønsket kryptering, så det ble forsket videre på hvorfor dette ikke funket, med både offentlig HiveMQ broker, privat HiveMQ broker og ved å opprette egen broker. Ved privat HiveMQ broker trengs det å opprettes tilganger knyttet til brukernavn/passord. Ut ifra forsøk ser det ut som at ved offentlig HiveMQ broker tillattes alle klienter å koble til brokeren og den ser ikke på brukernavn/passord under autentiseringen. Videre ble det testet forskjellige konfigurasjoner for kommunikasjon mellom broker og IoT-enhet. Tabell 6.1 viser testene som ble sentrale for å trekke vår konklusjon for årsaken til at kryptering ikke lot seg gjøre. Alle testene med brukernavn og passord benyttet samme kombinasjon.

Tabell 6.1: Test av konfigurasjoner for kommunikasjon mellom IoT-enhet og broker.

Test	Resultat	Trukket konklusjon
Test av kryptert kommunikasjon fra Advantech IoT-enhet med HiveMQ privat broker med brukernavn og passord.	✗ Feil under autorisasjon ved oppretting av forbindelse.	Mulig at IoT-enhet ikke støtter kryptering med gjeldende TLS versjon 1.2 eller 1.3. Også mulig at brukernavn/passord ikke fungerer.
Test av kryptert kommunikasjon fra webklient med HiveMQ privat broker med brukernavn og passord.	✓ Suksess: forbindelse og kommunikasjon fungerer.	Bekreftet at HiveMQ privat broker fungerer som den skal.
Test av kryptert kommunikasjon fra Zaphire med TLS 1.2 mot HiveMQ privat broker med brukernavn og passord.	✓ Suksess: forbindelse og kommunikasjon fungerer.	Bekreftet at Zaphire og HiveMQ privat broker fungerer som det skal.
Test av kryptert kommunikasjon fra Zaphire med TLS 1.1 mot HiveMQ privat broker med brukernavn og passord.	✗ Feil under autorisasjon ved oppretting av forbindelse.	Siden denne testen resulterte i samme feilmelding som den med brukernavn/passord-problemet, ble det konkludert at TLS 1.2 eller 1.3 ikke er støttet. Dette stemte ikke som vist av neste test.
Test av kryptert kommunikasjon fra Advantech IoT-enhet med HiveMQ offentlig broker uten å fylle inn brukernavn og passord.	✓ Suksess: forbindelse og kommunikasjon fungerer.	IoT-enhet støtter TLS 1.2 og/eller 1.3. Problemet forsvinner når brukernavn/passord ikke er en faktor. Derfor er dette sannsynlig årsaken til feil.

Ut ifra testingen under utviklingen og testingen fra Tabell 6.1 konkluderes det med at Advantech IoT-enheten støtter gjeldende TLS versjon 1.2 og 1.3, men at enheten ikke håndterer brukernavn/passord som forventet. Dette gjør at autentiseringen feiler hver gang dette kreves.

6.3 Testing av IoT-enhet i felt

I Tabell 6.2 vises testplanen for installasjon av enheten hos Norsk Titanium. Den inneholder de testene som ble gjennomført dagen før og under installasjon. En av testene gjennomført viste at det var en feil i tolkningen av signalet inn på enheten. Det ble ikke mottatt det samme som ble målt. Selv med bare et lederpar koblet til så fløt signalet over i de andre kanalene. Dette viste seg å være en fysisk innstilling som måtte gjøres ved å flippe DIP-switcher fra spenning til strøm på I/O enheten dette er beskrevet i bedre detalj i 5.2.2 Feil i avlesning av signaler. Etter å ha endret innstillingen ble det koblet på igjen og testet, noe som viste at problemet var løst.

Tabell 6.2: Testplan før og under installasjon av enhet hos Norsk Titanium.

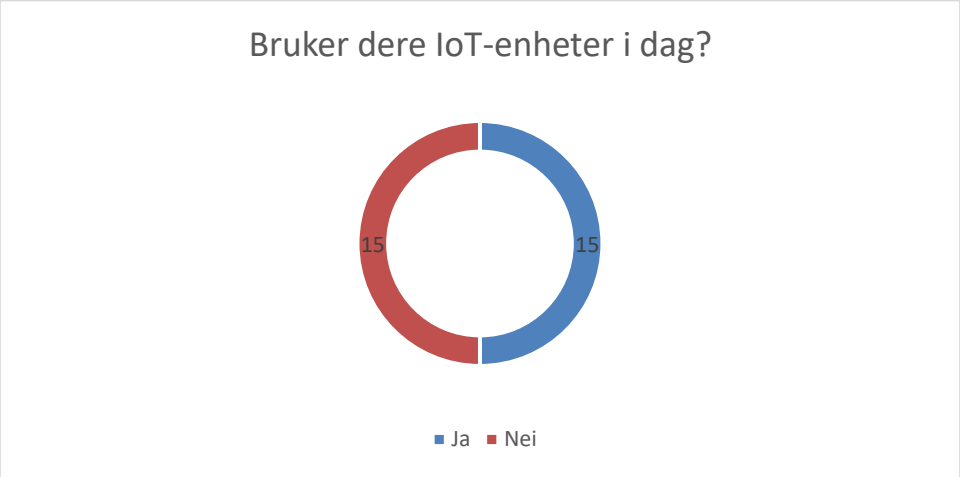
Tester i kronologisk rekkefølge	Dato	Resultat
Teste strømtilførsel til enheten	18.04.2023	Tilførsel OK
Teste at enheten starter opp	18.04.2023	Starter OK
Teste at enheten mottar strøm	18.04.2023	Lader OK
Teste at enheten mottar signaler	18.04.2023	Signal inn OK
Teste dataoverføring til Zaphire	18.04.2023	Kommunikasjon OK
Teste strømtilførsel til enheten	19.04.2023	Tilførsel OK
Teste at enheten starter opp	19.04.2023	Starter OK
Teste at enheten mottar strøm	19.04.2023	Lader OK
Teste at signalene er innenfor spesifikasjon	19.04.2023	Signal OK
Teste at enheten mottar signalene riktig	19.04.2023	Signaler IKKE OK
Teste forbindelsen mot broker	19.04.2023	Forbindelse OK
Teste å sende melding	19.04.2023	Melding OK
Teste dataoverføring til Zaphire	19.04.2023	Kommunikasjon OK

7 Resultat av IoT-løsningen

Dette kapittelet vil ta for seg resultatene fra prosjektet. Disse inkluderer en markedsanalyse og en løsning for å kommunisere mellom IoT-enheter og Zaphire på en skalerbar måte. Disse funnene vil danne grunnlaget for videre diskusjon og konklusjon rundt oppgaven.

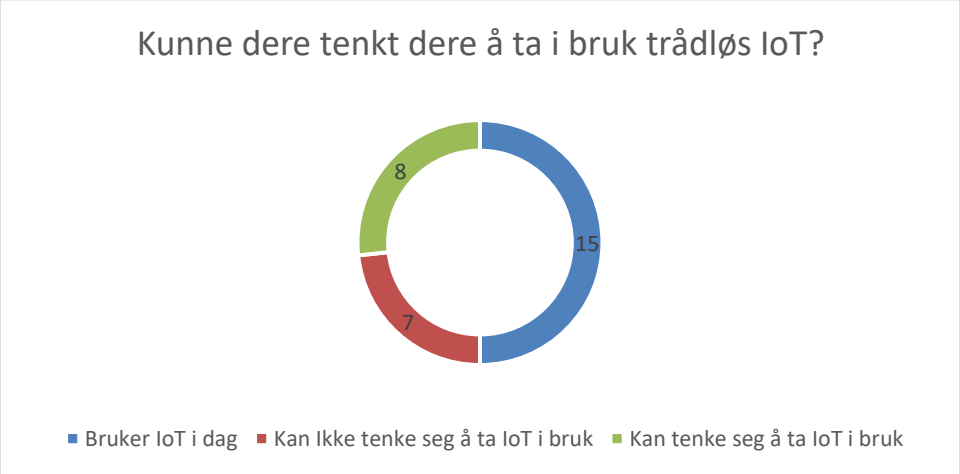
7.1 Markedsanalyse fra Industriuka

Det ble gjennomført en spørreundersøkelse under messen ved Industriuka i Porsgrunn 2023 hvor samtlige bedrifter i den ene hallen ble spurt forskjellige spørsmål. Det første av disse er det vist i Figur 7.1 hvor det kommer fram at halvparten av de som ble spurt benytter IoT-enheter i dag.



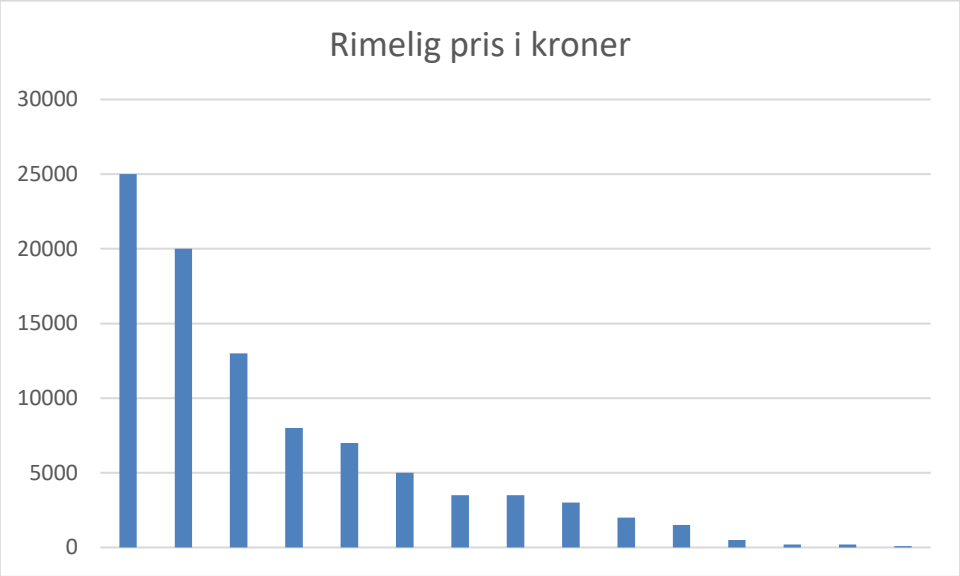
Figur 7.1: Andel spurte bedrifter som bruker IoT-enheter i dag.

Videre ble de som svarte at «de ikke brukte IoT i dag» spurt om de kunne tenke seg å ta i bruk trådløs IoT nå eller i fremtiden. Figur 7.2 viser hvordan de som svarte nei i Figur 7.1 brytes videre ned i de som kan og ikke kan tenke seg å ta i bruk IoT.



Figur 7.2: Viser andel som benytter, kunne tenkt seg å benytte og ikke ønsker IoT.

Pris var neste spørsmål på listen og her ble hver bedrift spurt «Hva er en rimelig pris for en IoT-enhet?». Dette viste seg at var et vanskelig, potensielt umulig spørsmål å få et godt svar på. Det var flere bedrifter som var misfornøyd med hvor vagt spørsmålet var. Uavhengig av det viser Figur 7.3 de svarene som ble gitt sortert fra størst til minst. Her er det stor variasjon noe som kan få en til å reflektere over spørsmålet. Det er mulig å stipulere at det er rom for mange forskjellige enheter til mange forskjellige priser gitt at den oppfyller ønsket krav fra kunden.



Figur 7.3: Viser variasjonen i «rimelig» pris.

Andre nyttige svar var «Så lenge det er gunstig på bunnlinjen» og «Hvis det er mer lønnsomt enn det vi har, så investerer vi». De gir også inntrykk av at pris må matche funksjonaliteten forventet og da går det meste greit.

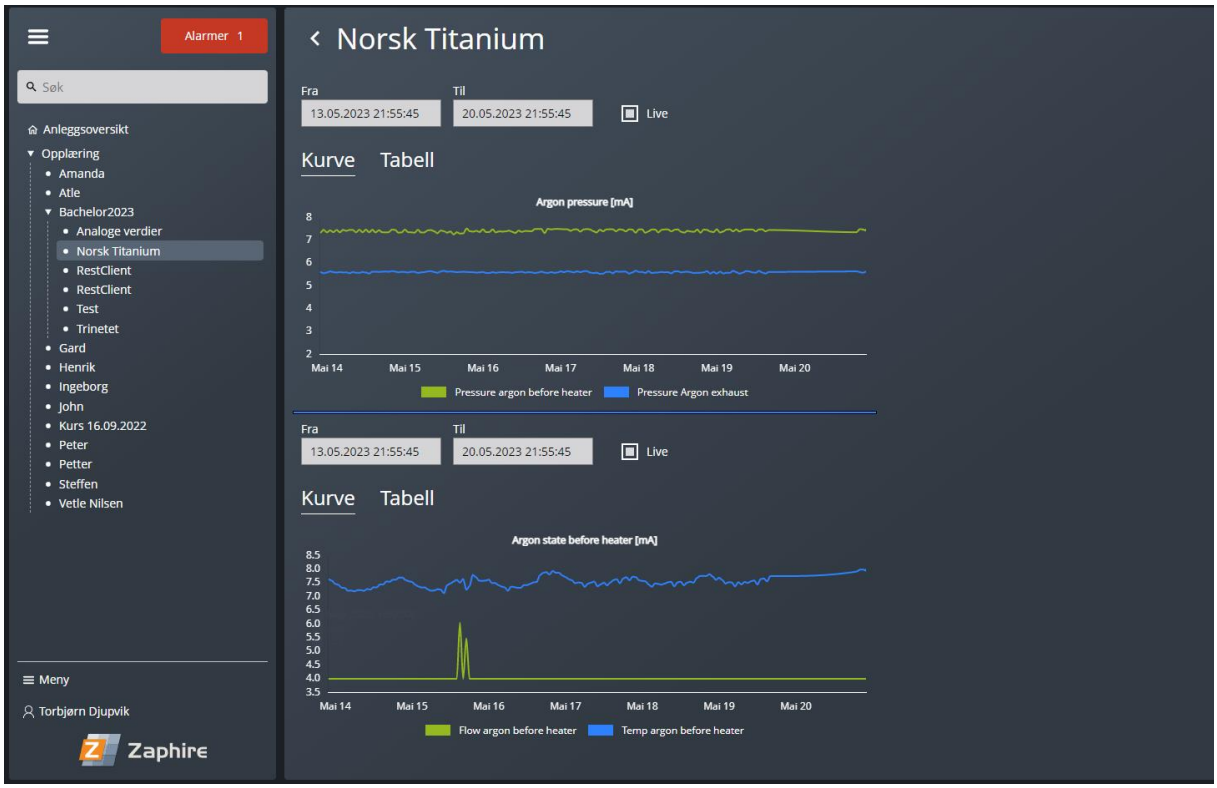
Det siste som ble spurt om var om bedriftene savnet noe funksjonalitet. Her var det litt forskjellig nyttig:

- Trådløst
- Live Data
- Flow Måling
- Bedre batterilevetid
- Integrasjon mot andre systemer
- Lokasjon sporing, varsling ved slag
- Praktisk informasjon som gjør meg i stand til å ta bedre valg

Noen av disse manglene er dekket av dagens enheter hvis man ser etter dem. Det kan uansett være nyttig å ha i bakhode når man vurderer hvor man skal angripe markedet.

7.2 IoT-Løsningen

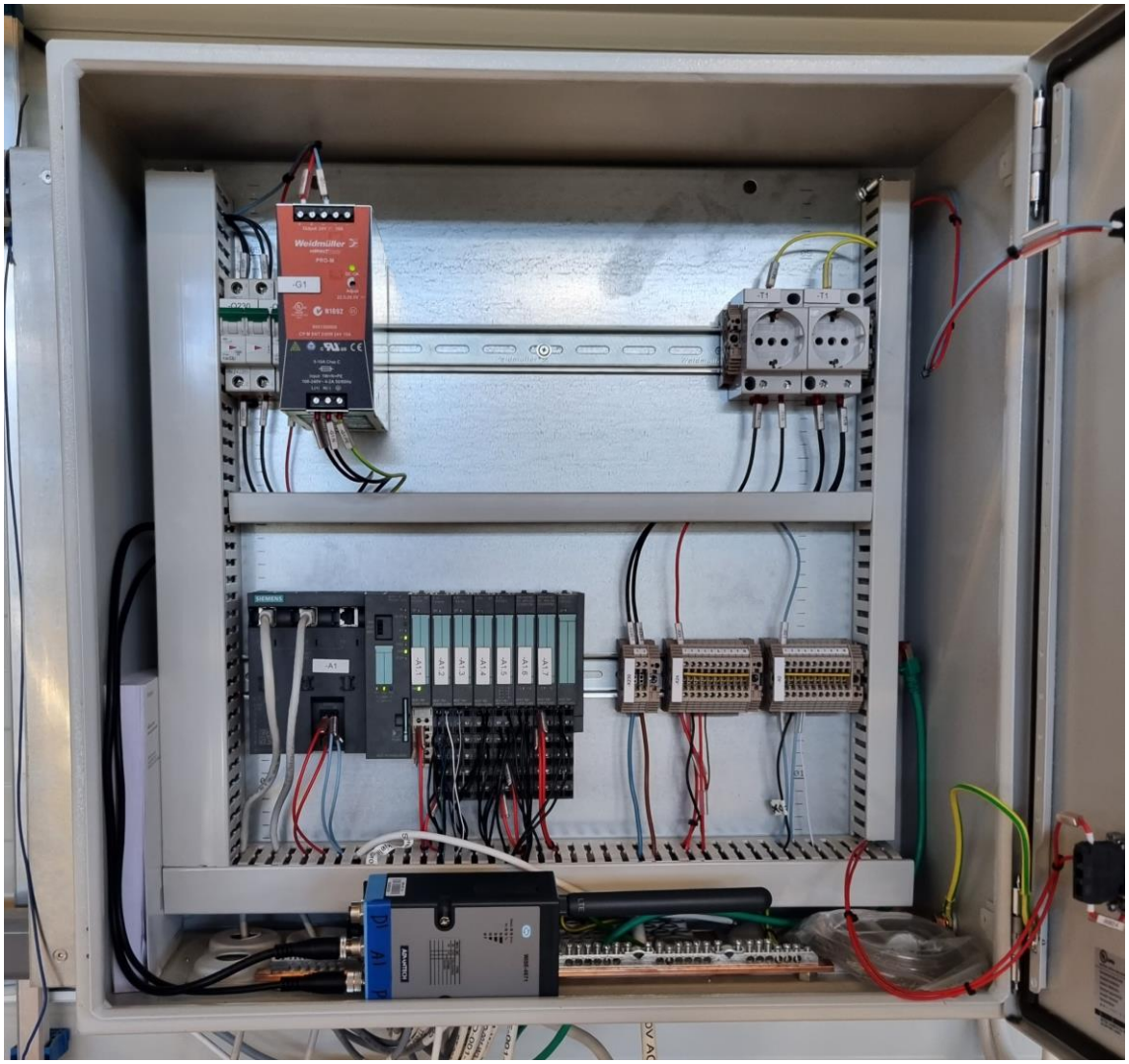
Prosjektet resulterte i en løsning som produserer data og overfører det til Zaphire som vist i Figur 7.4. Gjennomføringen er delvis tilfredsstillende ved at løsningen er funksjonell og vil kunne skalere godt og det dekker hovedkravet, men det er ikke tilfredsstillende sikkert. Dette kommer av manglende kryptering eller andre alternativer for å redusere leseligheten av kommunikasjonen av uvedkommende. Dette er noe som bør adresseres før løsningen blir satt i drift i stor skala.



Figur 7.4: Skjermbilde fra Zaphire som viser sensordata fra Norsk Titanium.

Løsningen benytter MQTT for å overføre data fra IoT-enheter via en eller flere brokere til Zaphire. Kommunikasjonen blir sendt over NB-IoT nettet for så å komme frem til en broker og deretter videre til Zaphire over vanlig kablet nett. Dette er funksjonelt og har potensiale til å bli en komplett løsning med litt videre arbeid for å bedre sikkerhets situasjonen.

Noen dager etter at løsningen ble testinstallert hos Norsk Titanium som vist i Figur 7.5: ble det oppdaget at løsningen hadde sluttet å sende data. Dette problemet kom av at kommunikasjonen hadde brukt mer data en først forventet. Datapakken på 30 MB hadde blitt brukt opp i løpet av en uke. Dette ble rettet opp ved å endre sendingsintervallet fra vært tiende sekund til hvert minutt noe som gjorde at datakvoten ville vare hele måneden.



Figur 7.5: IoT-løsningen installert hos Norsk Titanium.

8 Kommunikasjonsprotokoller som ble vurdert, men ikke brukt

Dette kapittelet vil beskrive de kommunikasjonsprotokollene som ble vurdert til bruk for å lage infrastruktur for datakommunikasjonen. Protokollene nevnt her er blitt vurdert, men ikke tatt i bruk. Det er ikke nødvendigvis fordi de er bedre eller dårligere enn MQTT som ble valgt. Det kommer mer om hvilke valg som ble tatt i kapittel 9.1 Vurdering og sammenlikning av MQTT, LwM2M og CoAP.

8.1 Introduksjon av CoAP

CoAP er en lett kommunikasjonsprotokoll som er designet for begrensede og ressursfattige IoT-enheter. Protokollen ble utviklet av "Constrained RESTful Environments" (CoRE) arbeidsgruppen i Internet Engineering Task Force (IETF). Den ble utviklet som en lett protokoll med målsetting om å muliggjøre kommunikasjon mellom IoT-enheter og skytjenester. [38]

CoAP servere ligner veldig på webservere i form av skalering og kommunikasjon. Begge kommuniserer forbindelsesløst ved å sende meldinger med headere og får statuskoder i respons. Protokollen har HTTP liknende egenskaper og skalerer veldig godt i likhet med webservere. Hvis det blir for mange enheter på serveren så den blir presset på ressurser kan kommunikasjonen bli forsinket eller forstyrret. For å få CoAP lett brukes UDP som transport lag. Dette reduserer mengden data brukt på header i forhold til TCP, som er vanlig på webservere. En annen egenskap er at meldingene kan bli sendt i binært format istedenfor tekst som reduserer dataforbruket. [38]

CoAP har ingen spesifikke sikkerhets funksjoner innebygd. Dette gjør protokollen avhengig av underliggende sikkerhetsfunksjoner fra transportlaget. Når denne protokollen blir tatt i bruk må sikkerhet tas ekstra hensyn til siden det ikke er innebygd. [39]

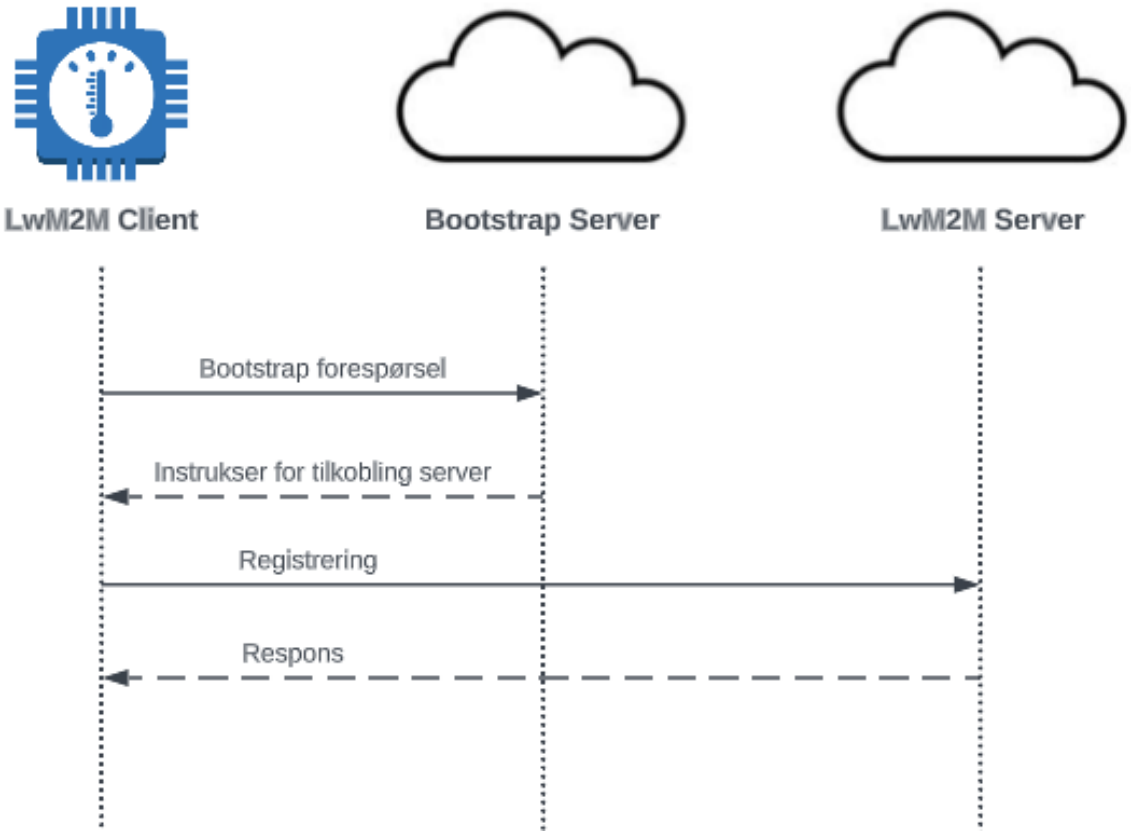
De viktigste egenskapene til CoAP er at klient og server ikke trenger å kjenne til hverandre på forhånd og kommunikasjonen foregår forbindelsesløst som bruker lite båndbredde. Videre er ikke protokollen avhengig av tilkobling for å kommunisere og det er støtte for Multicast og observasjon av ressurser. [38]

8.2 Introduksjon av LwM2M

LwM2M er en åpen kommunikasjon protokoll for IoT enheter fra Open Mobile Alliance (OMA). Protokollen er basert på CoAP og er laget for å enkelt sende meldinger og få oversikt på enheter. Målsettingen til LwM2M er å være så lett å ta i bruk som mulig. [40]

LwM2M støtter kommunikasjon over TCP, UDP og SMS. Dette gjør det veldig lett å bruke til de fleste anvendelser. Protokollen har ikke noe innebygde sikkerhetsfunksjoner, men kan få det fra underliggende sikkerhet i transportlaget. Et eksempel på sikkerhet er ende til ende kryptering med sikkerhetsnøkler. [41]

LwM2M skalerer veldig bra av samme grunn som CoAP. Protokollen støtter bootstrapping som gjør utbygging betydelig lettere. Figur 8.1 illustrerer hvordan det enkelt setter opp nye enheter. Enhetene kommer fra fabrikk konfigurert til å koble seg til en bootstrap server. Her får de tildelt ny sikkerhets nøkkel og server adresse automatisk. Enhetene kobler seg da opp mot den nye serveren og kan begynne å kommunisere. Protokollen støtter også sending av firmware over nett. Administrering blir enklere og mer fleksibelt siden styring og konfigurasjon kan gjøres via server. [42]



Figur 8.1: Sekvensdiagram for LwM2M-enheter ved førstegangsoppstart.

9 Diskusjon av IoT-løsningen

Dette kapittelet vil ta for seg de forskjellige valgene som ble gjort, hvorfor de ble gjort og hvordan de ble implementert. Det blir også diskutert forskjellige situasjoner produktet kan bli tatt i bruk i, og hva som kan være viktig å ta med seg i disse situasjonene. IoT-enheten som ble brukt i oppgaven og generelle sikkerhetstanker blir også tatt opp.

9.1 Vurdering og sammenlikning av MQTT, LwM2M og CoAP

For valg av protokoll til prosjektet og fremtidig utvidelse ble protokollene MQTT, LwM2M og CoAP vurdert opp mot hverandre. Protokollene ble vurdert på egenskaper og nytteverdi for prosjektet. Tabell 9.1 viser teknologienes egenskaper opp mot hverandre.

Tabell 9.1: Sammenligning av egenskapene til MQTT, LwM2M og CoAP. [38] [40]

Egenskap	MQTT	LwM2M	CoAP
Hensikt	Meldingsoverføring for IoT-enheter	Enhetsadministrasjon og datautveksling for IoT-enheter	Lett protokoll for ressursbegrensede enheter
Arkitektur	Publiser/abonner-modell	Klient-tjener-modell	Klient-tjener-modell
Enkelhet for implementasjon	Mulig gjennom støtte i Azure, AWS, Google Cloud	Ikke direkte støtte fra store tilbydere av skytjenester	Ikke direkte støtte fra store tilbydere av skytjenester
Utgitt	1999	2017	2014
Transportprotokoll	TCP/IP, WebSockets	UDP, TCP, SMS m.m.	UDP
Sikkerhet	Støtter SSL/TLS for kryptering og autentisering	Ingen integrert i protokoll bruker underliggende transportlag	Ingen integrert i protokoll bruker underliggende transportlag
Minste header-størrelse	2 byte	4 byte	4 byte
Meldingsstruktur	Kan bruke JSON, XML eller andre formater	TLV, JSON, Plain text, binary og en spesifikk objektstruktur definert av LwM2M-standarden	Bruker en spesifikk meldingsstruktur definert av CoAP-standarden

LwM2M har fordeler når det gjelder konfigurasjonsoppsett for firmwareoppdateringer og generelt oppsett. Det tillater også forespørsel for overføring av spesifikk sensordata. På papiret virker LwM2M som det beste alternativet, men for enkelhetens skyld ble MQTT valgt. MQTT oppfyller kravene fra PowerTech, selv om det ikke skalerer like godt som LwM2M. MQTT kan likevel skalere til rundt hundre tusen enheter [43], som er tilstrekkelig for PowerTechs plan om en broker per bygning.

Totalt sett handler prosjektet om å overføre data fra IoT-enhet til skyen på en enkel og trygg måte. MQTT er en vel prøvd metode som dekker behovet på en god måte. Det er en enkel arkitektur som er lett å skalere til behovet for bygningsautomasjon for enkelt bygg i dag. CoAP og LwM2M som bygger på CoAP er gode alternativer også, men de trenger noe mer oppsett for å virke på en god måte. Disse er betydelig nyere teknologier med bedre skaleringsevne og kunne godt vært det alternativet prosjektet tok i bruk.

Selv om LwM2M og CoAP er ny teknologi med begrensede ressurser for enkelt oppsett, er MQTT et godt kompromiss. MQTT er et etablert alternativ som egner seg for oppgaven, og det finnes mye god informasjon om hvordan det kan implementeres. Selv om LwM2M kunne vært bedre på lang sikt etter nødvendig oppsett, er MQTT et godt valg for nå.

9.2 Kubernetes opp mot andre alternativer

Kubernetes er det mest brukte verktøyet for containerbasert applikasjon. Fordeler med Kubernetes er at det kan automatisk rulle ut, skalere og orkestrere software. Annen viktig funksjonalitet er lastbalansering, lagrings organisering, automatisk tilbakestilling og selv gjenoppretting. Kubernetes har åpen kildekode og et stort samfunn for å få hjelp når det trengs, kan kjøres på hva som helst og er lett å organisere. Hovedulempene med Kubernetes er at installasjonsprosessen kan være kompleks og Docker Compose er ikke støttet. [44] [45]

Docker Swarm er et alternativ for å orkestrere Docker-containere som er lett å installere og ta i bruk. Det bruker lite ressurser og har åpen kildekode. Disse tingene gjør det til et godt alternativ for mindre prosjekter som ikke trenger hundre prosent oppe tid. Docker Swarm benytter seg av Docker API noe som begrenser funksjonaliteten og gjør det relativt lett å bryte ned, sammenlignet med andre alternativer. [44] [45]

OpenShift er litt annerledes enn Kubernetes og Docker Swarm siden det er bygget for å være på en lokal server. Det er utviklet av RedHat og er et annet populært alternativ til Kubernetes. På lik linje med de andre alternativene har OpenShift også åpen kildekode og har mange ferdig sertifiserte containerimages. OpenShift er mest anbefalt for RedHat og deres distribusjoner. Det kan være litt komplekst å installere og ta i bruk og skyvariant er ikke tilgjengelig. [44] [45]

Kubernetes blir da det naturlige valget mellom OpenShift, Docker Swarm og Kubernetes. Dette er fordi det kan kjøres i sky og har mye annen god funksjonalitet som lastbalansering, automatisk gjenoppretting og skalering. Oppetid er veldig viktig og det kan være store variasjoner i hvor mye trafikk som kommer inn. Derfor blir skalering og lastbalansering viktig for å håndtere trafikk variasjonen og automatisk gjenoppretting holder tjenesten oppe.

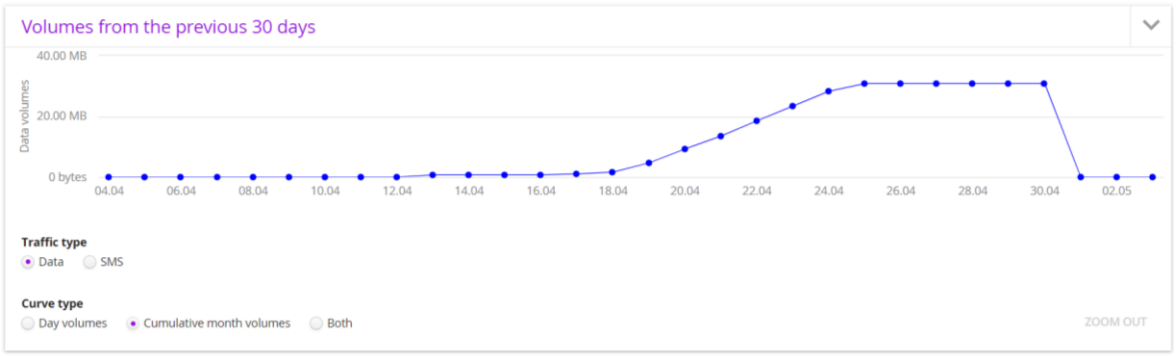
9.3 Integrasjon av systemet i praksis

For å se at kommunikasjonen virket ble det tatt i bruk en allerede eksisterende offentlig broker fra HiveMQ. Dette har funket utmerket for å teste utstyret, men for å kjøre dette for kunder så er ikke dette en løsning. HiveMQ selger også hosting av broker; dette er noe som kunne blitt tatt i brukt, men da blir prosjektet avhengig av HiveMQ sin løsning og hvis den endrer oppsettet eller lignende vil det kunne medføre store ulemper for PowerTech. Et annet alternativ er å hoste broker selv. Dette kan gjøres i skyløsningen til Microsoft Azure med containere noe som gjør det lettvint å skalere og veldig dynamisk. Ønsket er å ha en broker per site noe som skiller kommunikasjonen til kundene fra hverandre. Dette er et godt utgangspunkt med tanke på sikkerhet. Ved å kjøre i Azure så får PowerTech full kontroll på hva som kjører, når det kjører og trafikken på de forskjellige brokerne. Dette passer godt inn med resten av arkitekturen til PowerTech som også ligger i Azure. En ulempe med å kjøre broker selv er at PowerTech får ansvaret for at den virker. Dette er noe man slipper ved å betale noen andre for å kjøre den. Denne ulempen har PowerTech allerede demonstrert at de håndterer ved at det meste av infrastrukturen deres allerede er skybasert. Derfor ansees det som overkommelig at systemet deres tar på seg en oppgave til. Det blir derfor ingen drastisk endring eller utfordring i dette senarioet.

9.4 Bruk av NB-IoT i IoT-applikasjoner

Løsningen er i utgangspunktet tenkt å bruke NB-IoT telenett for å kommunisere. Det et godt alternativ som har stort fokus på sikker forbindelse og dekning sammenlignet med 4G/5G hvor kravet handler mer om hastighet [10]. Et annet alternativ kan være å koble det rett på kabel eller Wi-Fi. Det er tenkt at for eksempel en bonde som ønsker å overvåke nivået i siloene sine, skal praktisk og enkelt kunne bruke denne IoT-enheten. I den situasjonen er det sannsynligvis ikke Wi-Fi eller kablet internett å koble seg på. Derfor er NB-IoT et godt alternativ som tillater lettvinn installasjon.

Etter noen dager i drift hos Norsk Titanium sluttet enheten å virke på grunn av oppbrukt datakvote. Den kumulative databruken vises i Figur 9.1. Sendingsintervallet satt var for lavt til at datakvoten skulle holde ut måneden. Dette ble fikset ved å justere den fra hvert tiende sekund til hvert minutt og be enheten koble seg opp på nytt. I et reelt scenario ville sendingsintervallet blitt tilpasset etter testfasen er gjennomført, så dette er ikke et stort problem.



Figur 9.1: Oversikt i Telia IoT Service Portal over kumulativ databruk etter installasjon.

IoT handler om å sende små mengder data og få så mye som mulig ut av dem. Med det sagt er det noen åpenbare optimaliseringer som kan bli gjort i løsningen. Det blir for øyeblikket sendt betydelig mer data pr melding enn de fire analoge signalene som kunden behøver. I tillegg blir blant annet batterinivå og digitale innganger sendt, selv om disse ikke brukes.

9.5 Vurdering av Advantech WISE-4671 som IoT-enhet

Enheten som er brukt i oppgaven er foreslått av PowerTech for å være en brukbar enhet å teste med. Den har fungert fint til store deler av testingen og har kommunisert godt over NB-IoT nettet. Det har vært større problemer rundt MQTT oppkoblingen spesielt med tanke på kryptering. Her er det konkludert at det har vært noen problemer med passord og brukernavn for autorisering hos brokeren. Dette kan sannsynligvis fikses med en senere firmware oppdatering.

Det har i utgangspunktet vært en fin enhet å jobbe med. Den har vært god som en utviklingsplattform for å teste ut og raffinere idéer og metoder, men den er upraktisk dyr og har varierende grad av funksjonalitet. Dette gjør at det anbefales å finne en mer egnet enhet for det spesifikke behovet den skal fylle. Enheten vil da sannsynlig bli billigere, noe som gir større potensiale for lønnsomhet.

9.6 Sikkerhet nå og fremtidig

Det er ganske lett å lage noe som virker, men så fort sikkerhet kommer inn i bildet blir alt mer komplekst og vanskelig. Denne oppgaven er ikke et avvik fra det. Det ble forsøkt å ta i bruk TLS for å kryptere kommunikasjon mellom IoT-enheten, brokeren og Zaphire. Dette viste seg for mye å be om. Det er store fordeler ved å bruke kryptering med tanke på sikkerhet. Kryptering gjør det betydelig vanskeligere å lese av hva som blir sendt siden alt innholdet er obfuskert. Dette er utmerket for sikkerhet siden man da kan sende dataene sine over usikkert nett og ikke være bekymret for om andre kan lese hva du sender. Når produktet av denne bacheloroppgaven skal tas i bruk anbefales det på det sterkeste å ta i bruk kryptering og derfor valg av en annen enhet som støtter kryptering på en god måte, med det sagt har det blitt demonstrert at kryptering funker helt fint mellom Zaphire, broker og testklient. Dette er et godt tegn for senere implementasjon av kryptering for mer industriell bruk.

Det er også verdt å nevne at dagens kryptering kan være særdeles sårbar for «store now, decrypt later» angrep [46]. Dette er når noen lytter på kryptert informasjon og lagrer det selv om de ikke klarer å lese det akkurat nå. Den lagrede informasjonen kan bli dekryptert med kvantedatamaskiner i ikke så fjern fremtid. Dagens kryptering er potensielt veldig sårbar for kvantedatamaskiner og det er allerede påbegynt andre alternativer for kryptering som ikke kan så lett knekkes av kvantedatamaskiner [47]. Med dette sagt så er det foreløpig ren spekulasjon, men en mulig seriøs trussel verdt å gjøre noe med.

10 Oppsummering av prosjektet

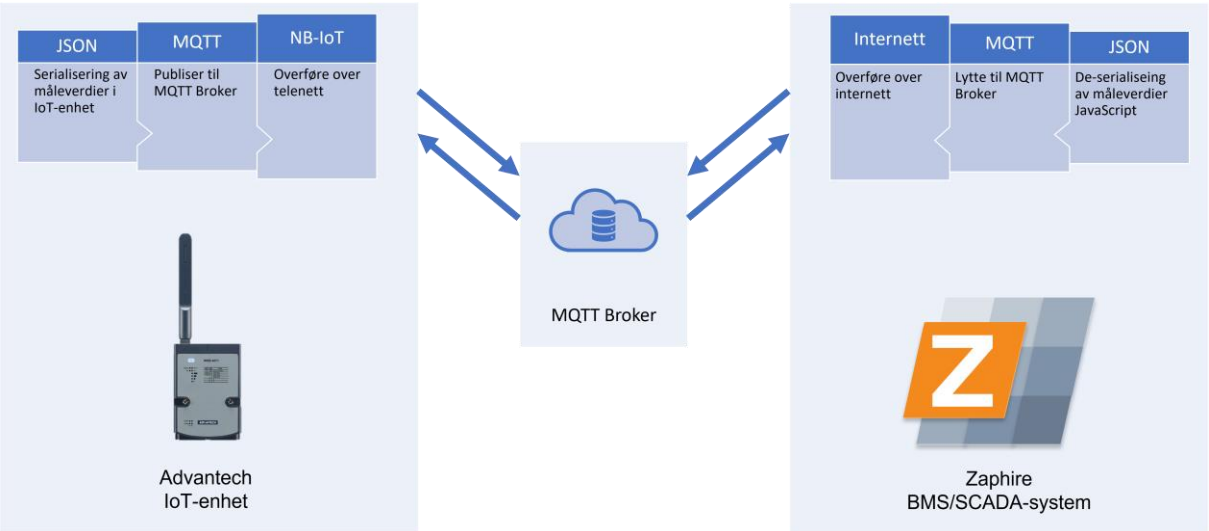
IoT-interaksjon med Zaphire er fullstendig innenfor rekkevidde. En praktisk metode er å bruke MQTT som har vært fokusert på i denne oppgaven, men det er andre alternativer også. For at dette skal være mulig å selge som et sluttprodukt burde infrastruktur bygges opp for å sette opp brokerne automatisk via Azure eller andre tjenester. Det må da også tas sikkerhetshensyn for å beskytte kundedata. Kryptering av dataene vil være en god start, og det vil være naturlig å regulere tilgangen til å abonnere på datastrømmen. Det er uansett betydelig vanskeligere å beskytte mot andre fiendtlige sendere. Der må det gjøres analyser av penetrasjonsevnene via inngående datastrømmer. Det kan også bygges et system hvor kunden kan få generert brukere til enhetene sine. Dette vil gjøre systemet betydelig mindre sårbart, men også vanskeligere og dyrere å komme i gang med. Totalt sett er det forbedringspotensialer på idriftsettelse, sikkerhet og enheten som sådan, men det virker som et produkt som vil kunne gjøre en god jobb i fremtiden.

10.1 Konklusjon av forskningsspørsmålene

Dette kapittelet vil besvare forskningsspørsmålene. En kort oppsummering av systemet vil også bli gitt og hvordan dette kan skalere. Dette er kjernen i prosjektet og er sentralt at blir besvart.

10.1.1 Kortfattet systembeskrivelse

Det første forskningsspørsmålet i problemstillingen er «Hva må til for å koble Advantech WISE-4671 industriell IoT-enhet eller lignende til BMS/SCADA-systemet til PowerTech?». Et svar på dette kan være å sette opp en Mosquitto MQTT broker. Zaphire må koble seg til brokeren og dirigere denne informasjonen til riktig kunde. IoT-enheten vil da kunne kobles opp til brokeren og kommunikasjon er etablert. Systemet er vist i Figur 10.1.



Figur 10.1: Overordnet systembeskrivelse.

10.1.2 Skalering og datasikkerhet

Det andre forskningsspørsmålet i problemstillingen er «Hva kan gjøres for å få til dette på en skalerbar og datasikker måte?». Et svar på det kan være å bruke Azure til å kjøre Mosquitto MQTT brokere dynamisk med Kubernetes, som forklart nøyere i kapittel 4.2 Oppsett av MQTT broker. Zaphire må så kobles opp til alle brokerne og dirigere denne kommunikasjonen til riktig kunde. IoT-enhetene må så kobles til den brokeren som er assosiert med sin kunde.

10.2 Betraktning av markedsundersøkelsen

Etter å ha vurdert resultatene fra markedsundersøkelsen er det gjort noen forsiktige konklusjoner. Markedet virker klart for å ta i bruk IoT-enheter og halvparten har alt gjort det. Dette lover godt for prosjektet som er gjennomført. Med tanke på naturen til IoT vil det alltid være behov for en leverandør av nye enheter, ettersom IoT-nettverkene til kunder utvides. Hvis en leverandør kan treffe riktig funksjonalitet, pris og kvalitet virker det veldig trygt å satse på IoT i dag. Markedet virker sultent for industriell IoT med den forventning at systemene fungerer over tid med høy oppe tid.

Med tanke på at det var relativt få svar i undersøkelsen blir ikke konklusjonene like sikre som de ellers kunne vært. Det burde også nevnes at bedriftene som ble spurt inkluderte alle i hallen, det vil si at det inkluderer for eksempel rekrutteringsbyråer og andre bedrifter som fort ville blitt utelukket av en selger av IoT. Med dette vurderes det til å være brukbart representativt av markedet, men større undersøkelser burde gjennomføres.

10.3 Videre arbeid

Løsning i prosjektet fungerer, men sikkerheten er svak. Dette burde være hovedfokus for videre utvikling. Kryptering er en potensielt god forbedring, men det er ikke sikkert det er mulig å få til med den valgte IoT-enheten. Derfor burde en alternativ IoT-enhet vurderes. Med tanke på sikkerhet burde det også vurderes å se på andre former for kryptering enn TLS. Denne formen for kryptering kan være utsatt med tanke på kvantedatamaskiner som nevnt i kapittel 9.6 Sikkerhet nå og fremtidig.

For å komme helt i mål med hensikten til oppgaven må infrastrukturen integreres inn i Zaphire arkitekturen. Zaphire trenger å ha oversikt over hvilke brokere som kjører og burde kunne regulere hvilke brokere som skal startes, stoppes, lages eller fjernes. Disse oppgavene kan være enkle å ordne for Zaphire sine utviklere, men uventede problemer oppstår ofte i software utvikling.

10.4 Avsluttende tanker

Prosjektet har uten tvil vært spennende, frustrerende, tilfredsstillende og givende. Det har vært en stor prosess hvor vi har fått brynet oss på forskjellige utfordringer. Oppgaven oppleves som svært likevel at det har vært interessant og lærerikt. Det har vært noen mindre vanskeligheter med å reise flere timer om dagen til PowerTech sine kontorer for å ha ansikt til ansikt diskusjon og veiledning der. Mindre diskusjoner og oppfølging har blitt håndtert over Teams.

Vår lokale veileder har også bidratt med raske tilbakemeldinger når vi har stått fast eller hatt spørsmål. Vi er veldig takknemlige for muligheten til å gjennomføre dette prosjektet i samarbeid med PowerTech og USN. Vi er spesielt takknemlige for innspillene fra både PowerTech og vår veileder på USN samt tilretteleggingen gjort av PowerTech med personell og utstyr.

Referanser

[1] Telenor, «Telenor dekningskart NB-IoT,» 2023. [Internett]. Available: https://www.telenor.no/bedrift/iot/dekning/. [Funnet 22 Mai 2023].

[2] Telia, «Telia dekningskart,» 2023. [Internett]. Available: https://www.telia.no/nett/dekning/. [Funnet Mai 22 2023].

[3] The Writing Center, «Writing a Scientific Research Report (IMRaD),» George Mason University, [Internett]. Available: https://writingcenter.gmu.edu/writing-resources/im-rad/writing-an-imrad-report. [Funnet 3. mai 2023].

[4] P. Stave, *privat samtale,* Drammen: PowerTech Engineering AS, 2023.

[5] F. Tsui, O. Karam og B. Bernal, «Essentials of Software Engineering, 4th Edition,» i *Chapter 6: Requirements Engineering*, Burlington (MA), Jones & Bartlett Learning, 2016.

[6] A. Moen, «Defining Requirements and Specifications,» ArgonDigital, 19. desember 2022. [Internett]. Available: https://argondigital.com/blog/product-management/requirements-vs-specifications-create-a-shared-vocabulary/. [Funnet 18. mai 2023].

[7] I. Sommerville, «An introduction to Requirements Engineering,» 6. desember 2013. [Internett]. Available: https://www.youtube.com/watch?v=Ec0s0z5uXQ8. [Funnet mai 2023].

[8] S. H. Kan, Metrics and Models in Software Quality Engineering, Second Edition, Boston (MA): Addison-Wesley Professional, 2003.

[9] N.-O. Skeie, «An introduction to Unified Modeling Language (UML) in Software Engineering,» Universitetet i Sørøst-Norge, 16. februar 2021. [Internett]. Available: https://www.halvorsen.blog/documents/teaching/courses/software_engineering/topics/uml.php. [Funnet mai 2023].

[10] Telia, «Low power wide area IoT,» [Internett]. Available: https://www.telia.no/bedrift/digitalisering/iot/skreddersydde-iot-losninger-for-din-bedrift/low-power-wide-area-iot/. [Funnet 16. mars 2023].

[11] «Startup: Kom i gang med IoT,» Telia, 29. september 2020. [Internett]. Available: https://www.telia.no/magasinet/startup/slik-kommer-din-startup-raskt-i-gang-med-iot/. [Funnet 19. mai 2023].

[12] 3GPP, «Standardization of NB-IOT completed,» 22. juni 2016. [Internett]. Available: https://www.3gpp.org/news-events/3gpp-news/nb-iot-complete. [Funnet 16. mars 2023].

[13] Telia, «Low Power Wide Area Guide,» juni 2021. [Internett]. Available: https://www.telia.no/contentassets/764ba9872ea94f9790a6ed500059f073/guide_lpwa_lowres.pdf. [Funnet februar 2023].

[14] 3GPP, «NarrowBand IOT,» 17. september 2015. [Internett]. Available: https://www.3gpp.org/news-events/3gpp-news/niot. [Funnet 16. mars 2023].

[15] Ericsson, «NB IoT Smart Parking,» YouTube, 23. august 2016. [Internett]. Available: https://youtu.be/U-yPdrDwE9E. [Funnet 16. mars 2023].

[16] IDG TECHtalk, «What is NB-IoT?,» YouTube, 1. februar 2018. [Internett]. Available: https://youtu.be/pf7wcl1IZYc. [Funnet 21. mars 2023].

[17] Nasjonal kommunikasjonsmyndighet (Nkom), «Frekvenskompass for mobilkommunikasjon,» 31. desember 2020. [Internett]. Available: https://www.nkom.no/frekvenser-og-elektronisk-utstyr/tillatelse-til-a-bruke-frekvenser/offentlig-mobilkommunikasjon-i-norge/_/attachment/download/4043371a-9651-44e8-b69c-60f338c5d39c:c598cdd12ad28482803f58a084eabab9e3be979e/Nkom%20-%20Frekvenskompass%20fo. [Funnet 16. mars 2023].

[18] M. V. Bøe, «Bølgelengde,» Store norske leksikon, 23. juni 2021. [Internett]. Available: https://snl.no/b%C3%B8lgelengde. [Funnet 21. mars 2023].

[19] Innosent GmbH Germany, «On the same wavelength – Radar frequencies, eligibility for approval, and bandwidth,» [Internett]. Available: https://radar-blog.innosent.de/en/on-the-same-wavelength/. [Funnet mai 2023].

[20] «MQTT,» 2022. [Internett]. Available: https://mqtt.org/faq/. [Funnet 24. januar 2023].

[21] M. Walter, «Top 10 IoT Scalability Tests for an MQTT Broker,» HiveMQ, 26. mars 2020. [Internett]. Available: https://www.hivemq.com/article/mqtt-broker-scalability-tests/. [Funnet 23. februar 2023].

[22] The HiveMQ Team, «HiveMQ,» 19. januar 2015. [Internett]. Available: https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/. [Funnet 23. februar 2023].

[23] Google, [Internett]. Available: https://cloud.google.com/learn/what-is-microservices-architecture. [Funnet 20. mai 2023].

[24] C. M. Lundheim, *Presentasjon på USN campus,* Porsgrunn: PowerTech Engineering AS, 2023.

[25] Microsoft, «What is Kubernetes?,» Microsoft Azure, [Internett]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-kubernetes/. [Funnet mai 2023].

[26] The Kubernetes community, «Kubernetes Documentation | Overview,» 3. januar 2023. [Internett]. Available: https://kubernetes.io/docs/concepts/overview/. [Funnet 26. april 2023].

[27] Amazon Web Services, Inc., «What Is Load Balancing?,» [Internett]. Available: https://aws.amazon.com/what-is/load-balancing/. [Funnet mai 2023].

[28] The Kubernetes community, «Containers | Kubernetes,» 21. november 2022. [Internett]. Available: https://kubernetes.io/docs/concepts/containers/. [Funnet 29. april 2023].

[29] «TatvaSoft,» Vishal Shah, 8 Desember 2021. [Internett]. Available: https://www.tatvasoft.com/blog/how-to-setup-your-own-mqtt-broker-on-azure/. [Funnet 9 Mars 2023].

[30] Telia, «Telia NB-IoT + LTE-M Starter-kit,» [Internett]. Available: https://shop.business.teliacompany.com/s/product-detail-page?name=NB-Iot-LTE-M-Starter-Kit&language=no.

[31] Advantech, «WISE Studio - Advantech Support,» 12. mars 2023. [Internett]. Available: https://www.advantech.com/en-eu/support/details/utility?id=1-1MJSJKX. [Funnet 18. januar 2023].

[32] HiveMQ, «HiveMQ Public MQTT broker,» [Internett]. Available: http://www.mqtt-dashboard.com/. [Funnet 18. januar 2023].

[33] HiveMQ, «MQTT Websocket Client,» [Internett]. Available: https://www.hivemq.com/demos/websocket-client/. [Funnet januar 2023].

[34] Norsk Titanium AS, «Norsk Titanium | Technology,» [Internett]. Available: https://www.norsktitanium.com/technology#process. [Funnet mai 2023].

[35] Advantech Co., Ltd, «User Manual for WISE-4671 Series,» 7. desmeber 2020. [Internett]. Available: https://www.advantech.com/en-eu/support/details/manual?id=1-20K4T6P. [Funnet februar 2023].

[36] H.-P. Halvorsen, «Software Testing,» 2022. [Internett]. Available: https://www.halvorsen.blog/documents/teaching/courses/software_engineering/topics/software_testing.php. [Funnet mai 2023].

[37] IBM, «What is software testing?,» [Internett]. Available: https://www.ibm.com/topics/software-testing. [Funnet mai 2023].

[38] Internet Engineering Task Force (IETF), «The Constrained Application Protocol (CoAP),» juni 2014. [Internett]. Available: https://datatracker.ietf.org/doc/html/rfc7252. [Funnet 20. mai 2023].

[39] A. Capossele, V. Cervo, G. De Cicco og C. Petrioli, «Security as a CoAP resource: An optimized DTLS implementation for the IoT,» IEEE, 12. juni 2015. [Internett]. Available: https://ieeexplore.ieee.org/document/7248379. [Funnet mai 2023].

[40] Open Mobile Alliance, «LwM2M v1.1,» 2019. [Internett]. Available: https://www.openmobilealliance.org/release/LightweightM2M/Lightweight_Machine_to_Machine-v1_1-OMASpecworks.pdf. [Funnet februar 2023].

[41] L. Slats, «What's this LwM2M standard, and why should you care?,» AVSystem, 22. desember 2022. [Internett]. Available: https://www.wevolver.com/article/whats-this-lwm2m-standard-and-why-should-you-care. [Funnet mai 2023].

[42] Open Mobile Alliance, «Lightweight Machine to Machine Technical Specification,» 8. februar 2017. [Internett]. Available: http://www.openmobilealliance.org/release/LightweightM2M/V1_0-20170208-A/OMA-TS-LightweightM2M-V1_0-20170208-A.pdf. [Funnet mars 2023].

[43] Scalagent, «Benchmark ofMQTT servers,» Scalagent, januar 2015. [Internett]. Available: http://www.scalagent.com/IMG/pdf/Benchmark_MQTT_servers-v1-1.pdf. [Funnet 19. mai 2023].

[44] P. V. K. Rajagopal, «Docker Swarm vs Kubernetes vs OpenShift,» Digital Varys, [Internett]. Available: https://digitalvarys.com/docker-swarm-vs-kubernetes-vs-openshift/. [Funnet 20. mai 2023].

[45] CloudZero Team, «Kubernetes Vs. Docker Vs. OpenShift: What's The Difference?,» 18. november 2022. [Internett]. Available: https://www.cloudzero.com/blog/kubernetes-vs-docker. [Funnet 20. mai 2023].

[46] L. M. Gowran, «Siliconrepublic,» Silicon Republic, 19. august 2022. [Internett]. Available: https://www.siliconrepublic.com/enterprise/quantum-apocalypse-store-now-decrypt-later-encryption. [Funnet 18. mai 2023].

[47] J. Duran, «Guest Post: Harvest Now, Decrypt Later? The Truth Behind This Common Quantum Theory,» The Quantum Insider, 7. februar 2023. [Internett]. Available: https://thequantuminsider.com/2023/02/07/guest-post-harvest-now-decrypt-later-the-truth-behind-this-common-quantum-theory/. [Funnet 18. mai 2023].