OsloMet – storbyuniversitetet
Fakultet for teknologi, kunst og design
Institutt for maskin, elektronikk og kjemi
Postboks 4 St. Olavs plass, 0130 Oslo
Telefon: 67 23 50 00

PROSJEKT NR.

TILGJENGELIGHET

BACHELOROPPGAVE

BACHELOROPPGAVENS TITTEL	DATO
Dataoverføring fra PLS til sky	23.05.2019
	ANTALL SIDER / BILAG 71
FORFATTERE Christoffer Larsen Dreyer, Petter Hellum Foyn, Sjamil Mosajevitsj Gazimagamaev og Margarita Kirmichi,	INTERN VEILEDER Nils Sponheim

UTFØRT I SAMMARBEID MED	EKSTERN VEILEDER
StepSolution AS	Marius Slagsvold

SAMMENDRAG

Oppgaven går ut på å sette opp en PLS for kommunikasjon med en skytjeneste via en GSM-ruter. Oppsettet gjør et mulig for PLSen å laste opp og lagre måledata på skyen over mobilnettet uavhengig av hvor PLS skal holdes til. Måldataen som lagres av PLSen skal behandles gjennom en algoritme, der det blir vurdert om dataen skal overføres videre til skyen eller ikke avhengig av måledataenes status. PLS-programmet skal lages som en modul i TIA Portal, der modulen skal prøves å gjøres kompatibel med TIA Openness for enkel utvidelse og tilpassing av modulen for senere anledninger.

3 STIKKORD
PLS
Skytjeneste

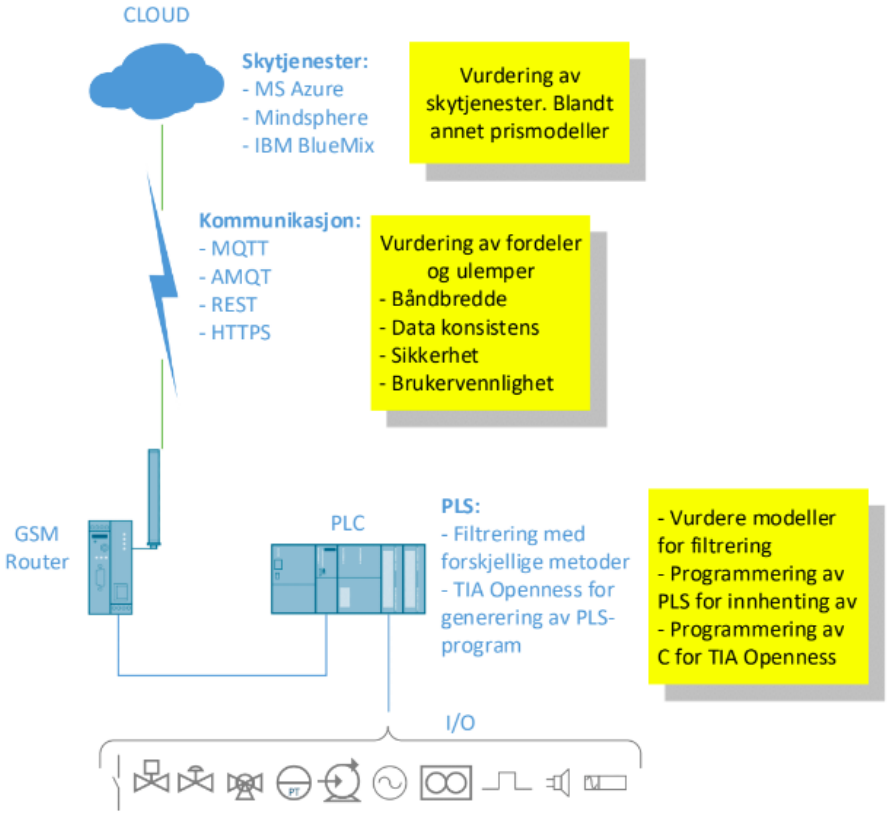
Innholdsfortegnelse

1. Innledning

Oppgaven går ut på å sette opp en PLS for kommunikasjon med en skytjeneste via en GSM-ruter. Oppsettet gjør det mulig for PLSen å laste opp og lagre måledata på skyen over mobilnettet uavhengig av hvor PLS skal holde til (gitt oppfylt krav om mobilnettdekning). Et eksempel på oppsett er visualisert i figur 1. Det finnes ulike kommunikasjonsprotokoller som kan brukes for kommunikasjonen, og vår gruppe skal vurdere hvilken som egner seg best til dette formålet med hensyn på båndbredde, datakonsistens, sikkerhet og brukervennlighet. Hvilken skytjeneste som skal motta og lagre måledata skal også vurderes og velges med hensyn på brukervennlighet, sikkerhet, båndbredde, kompatibilitet og pris.

Måledataene som lagres av PLSen skal behandles gjennom en algoritme, der det blir vurdert om dataene skal overføres videre til sky eller ikke avhengig av måledataenes status. Brukeren skal ha mulighet til å konfigurere detaljer på hva som er ønsket verdi ved målingene eller ikke som algoritmen skal ta utgangspunkt i. Disse kan være som maks- og minimumsverdier, målt stigning, diverse uregelmessigheter og lignende.

PLS-programmet bestående av måledataalgoritme og kommunikasjonsoppsett skal lages som en modul i TIA Portal. Det er ønskelig at denne modulen skal gjøres kompatibel med TIA Openness for enkel utvidelse og tilpassing av modulen for senere prosjekter, men ikke et krav for prosjektets funksjonalitet.



Figur 1: Figuren over viser et eksempel på ønsket systemløsning, levert av oppdragsgiver

Det at arbeidet lagres som en modul med støtte for generering med TIA Openness gjør det anvendelig for andre å ta det i bruk til senere prosjekter. Oppgaven er relevant for det oppdragsgiver leverer til sine kunder, slik at deres bedrift kan ha nytte av vårt arbeid i senere tid.

På grunn av industriens stadig hastede teknologiske utvikling er vi i dag midt i en ny industriell revolusjon. Denne kalles *Industri 4.0* og går ut på digitalisering, IoT og bruk og optimalisering av smarte løsninger. Arbeidet vårt er derfor aktuelt og fremtidsrettet i industrien da nettopp digitalisering og smart håndtering av måledata står sentralt i oppgaven.

Rapporten vektlegger mye fokus på teori og bakgrunn, da dette er en essensiell del av vurderingen av både skytjenestene og protokollene. Deretter står systemoppsett og konfigurasjon sentralt i forhold til PLS, ruter og skytjenesteleverandør. Siste innsats er programmeringen av PLS og algoritmen, og utforskning av TIA Openness.

2. Bakgrunn og hensikt

2.1. Status og problemstilling

Gitt mangfoldet av de store teknologiske endringer og oppfinnelser som har skjedd det siste tiåret, eksempelvis strømningstjenester, kryptovaluta, diverse apper, AR og VR, er det sannsynlig at *cloud computing* har vært innblandet. Det har påvirket både konsumer- og arbeidslivet i enorm grad, og deling av informasjon gjennom et skybasert distribuert nettverk tillater brukere på tvers av kloden å jobbe og interagere mer effektivt uten å nødvendigvis ofre sikkerheten eller autentisiteten til dataen som blir delt.

Cloud computing er en betegnelse for en variasjon av aktiviteter som foregår på eksterne servere i internettilkoblede tjenerparker, eller skytjenester. Dette innebærer forskjellig programvare, prosessering og lagring av data som ligger på slike servere og som blir levert som en tjeneste (i motsetning til et produkt). Vi skal ta i bruk en slik tjeneste for å digitalisere målt data, og for å kunne observere den og mulige avvik uansett hvor man befinner seg (gitt internettdekning).

Denne typen digitalisering har de siste årene nådd den industrielle bransjen. Det har kommet en stor etterspørsel etter digitalisering av måledata, da lagring av slik data på en ekstern server gir bedre mulighet for monitorering, feilsøking og logging av prosessdata. Ved å ha dataen fra industrielle prosesser loggført på skyen kan bedrifter lettere analysere prosessene og gjøre tiltak deretter.

Vår problemstilling er hvordan kan vi behandle målt data lokalt, sende et bestemt utvalg av dataen til en sky og dermed lagre den på en effektiv, trygg og billig måte uten å aktivt administrere prosessen?

2.2. Mål og avgrensninger

Målet vårt er i prinsippet å få en vellykket overføring av data til valgt skyplattform. Data skal deretter bare sendes gitt uregelmessigheter eller verdier som forekommer utenfor definert og akseptabelt verdiområde. Dette innebærer at vi skal ha valgt en kommunikasjonsprotokoll og en skytjeneste som passer best til vårt formål. Det er ingen krav til beslutningen vår annet enn at det skal tilsynelatende være den beste løsningen, der vurderingene skal basere seg på brukervennlighet, båndbredde, datakonsistens, sikkerhet og pris.

Begrensningene våre ligger i de tekniske spesifikasjonene til maskinvaren og programvaren vi har til rådighet. Valg av skytjeneste og protokoll er noe avhengig av hverandre og sikkerhetsløsninger av noe begrenset av ruteren vi bruker, noe som vi går innpå senere.

Tabell 1
Utvalg av maskinvare og programvare vi har tilgjengelig til prosjektet

Type	**Utvalg**
Maskinvare	
PLS	Siemens S7-1200
Rutere	Sixnet series RAM 6000, SCALANCE M874-2, SIMATIC CP 1243-1 (GSM extension)
Programvare	
PLS Programvare	TIA Portal
	TIA Openness
Virtuell maskin	VMware
Internettleverandør/mobiloperatør	
	Com4 Abonnement på 1GB
Kommunikasjonsprotokoll	Ingen begrensninger
Skytjenester	Ingen begrensninger

2.3. Kravspesifikasjon og Functional Design Specification

Tabell 2
Kravspesifikasjon for prosjektet

Kravspesifikasjon	
Anskaffelsens formål	Behov for en IT-løsning for opplasting av data fra en S7-1200 PLS til skyen ved hjelp av en GSM-ruter.
Krav til egenskaper	En sammenligning og vurdering av skytjenester, kommunikasjonsprotokoller og filtreringsmetoder. En brukervennlig kommunikasjonsprotokoll med krav til lite båndbredde. En rimelig og brukervennlig skytjeneste for IoT og industriell prosessdata. En filtreringsalgoritme som lar kun interessant data bli lastet opp til en skytjeneste. En sikkerhetsløsning for overføringen av data. En vurdering av TIA Openness sin brukervennlighet.
Løsningsmodell og metoder	Strukturert systemutvikling (fossefallsmodell). Kronologisk sammensetning av valg og faser.
Produkter og tjenester	TIA Portal-prosjekt med moduler som kan gjøres kompatibelt med TIA Openness for videre bruk. Beskrivelse av løsningens oppsett: • PLS- og ruteroppsett • Skytjeneste-kontooppsett • Oppretting av kommunikasjon mellom PLS og skytjenesten med valgt protokoll • TIA Openness-oppsett • Sikkerhetstiltak
Tekniske krav	

PLS	Siemens S7-1200
Ruter	En GSM-ruter
Programvare	TIA Portal TIA Openness

FDS er lagt ved som vedlegg.

3. Teori

3.1. PLS

Bokstavene PLS står for Programmerbar Logisk Styring, Programmable Logic Controllers (PLC på engelsk). En PLS er en industriell kontrollerbar datamaskin med rader av innganger og utganger. Denne datamaskinen følger kontinuerlig med på tilstanden av inngangsenhetene som er koblet på enheten. Ut ifra et tilpasset program gjør PLSen bestemmelser basert på programmet til å kontrollere tilstanden til utgangsenhetene.

Med hjelp av en PLS kan man forbedre nesten alle typer produksjonslinjer, maskinfunksjoner og prosesser. Før var maskinene i automatiseringsindustrien drevet av elektromekaniske relékretser og disse var laget for bestemte formål. Mengden med kabler og reléer var plasskrevende, og enkle funksjoner og prosesser var avansert å automatisere. Endringer i den kablede "logikken" var også kostbare og omfattende. PLS'ene ble utviklet for å være lett programmerbare og omprogrammerbare om endringer skulle være ønskelig. *Ladder*-logikk og -diagrammene skulle etterligne kretsskjemaer og relélogikk, der strømmen beveger seg fra venstre til høyre gjennom lukkede kontakter for å aktivere en reléspole, og deretter ned til neste linje. Slik ligner diagrammet en stige. PLSene har løst mange av de tidligere ulempene og med tiden er også blitt forbedret med tanke på internminne og hastighet.

Det finnes PLSer i både enhetlig (fast) og modulær form. En enhetlig PLS inneholder alle grunnleggende komponenter i en enkel boks og er ofte direkte koblet til enheten som skal kontrolleres. Disse er generelt produsert for enklere funksjoner. En modulær PLS består av flere moduler som kan settes sammen for en tilpasset løsning. De ekstra modulene kan for eksempel være ekstra inn- og utganger, eller en A/D-omformer og kobles til kjerneenheten. Modulære PLSer er populære og passer større og mer avanserte systemer.

Fordi PLSens oppgave er å behandle signaldata, er den konstruert på en slik måte at den jobber syklisk. Det vil si at den prosesserer programmet den kjører på en metodisk måte, om og om igjen. Hver syklus fungerer ved at PLSen først oppdaterer status på dens innganger, deretter behandler PLS-programmet, og til slutt oppdaterer alle utganger. Dette skjer i en evig løkke i en høy hastighet, avhengig av hvor rask PLSens prosessor/CPU er og hvor mye data den må behandle. En typisk syklustid til en PLS er på omtrent 2-10ms.

Vi har blitt tildelt Siemens S7-1200 for gjennomføring av oppgaven. Denne PLSen er modulær og er laget for enkle operasjoner. S7-1200-systemet består av CPU-ytelsesklassen CPU 1212C.

3.2. Programmeringsspråk

En PLS kan programmeres ved hjelp av forskjellige programmeringsspråk og metoder i et program på en PC. Deretter kan programmet lastes over på PLSen direkte over en kabeltilkobling eller et nettverk. Fem ulike programmeringsspråk er definert i den 3. delen av den åpne internasjonale standarden IEC-61131 (IEC-61131-3). Disse er Ladder-diagram (LD), Funksjonsblokkdiagram (FBD), Strukturert tekst (ST), Instruksjonsliste (IL) og Sekvensielle funksjonsdiagram (SFC). I prosjektet vårt legger vi hovedvekten på strukturert tekst, men bruker også elementer av funksjonsblokker og ladder.

TIA Openness som vi også skal se på, bruker C#. Det er et programmeringsspråk som blant annet kan brukes for å kjøre applikasjoner på .NET-rammeverk, som Openness benytter seg av. .NET-rammeverket er en programvareutviklingsplattform fra Microsoft, som gjør det mulig å få tilgang til og ta i bruk funksjoner fra andre program og inkludere de i sitt eget uten å måtte programmere disse på nytt.

3.2.1. Strukturert tekst (ST)

Strukturert tekst er et avansert tekstbasert språk som krever en struktur i programmeringen. Språkstrukturen (syntaksen) minner svært om PASCAL da ST er basert på det, men kan også sammenlignes med Basic eller C. ST er imidlertid mindre visuelt enn både ladder diagram og funksjonsblokkdiagram, men samtidig er det en ryddig struktur som går igjen som tar mindre plass og det kan oppleves enklere å fatte sammenhengen og flyten til større programmer. Språket tillater komplekse algoritmer og omfattende betinget kode, samt repeterbare operasjoner. Det støtter et bredt spekter av standardfunksjoner og operatorer.

3.2.2. Funksjonsblokkdiagram (FBD)

Funksjonsblokkdiagram er et grafisk språk som skildrer signal- og datastrømmer gjennom gjenbrukbare funksjonsblokker. En funksjonsblokk er en enhet med en eller flere innganger og en eller flere utganger. Hver blokk har en oppgave og behandler den innkommende dataen deretter, for å så oppdatere den i utgangene. Blokkene kan være timere, tellere, vipper og logiske porter, matematiske operasjoner og lignende, og er en visualisering av hva som skjer med signalstrømmen. Man kan også ha funksjonsblokker som inneholder funksjoner skrevet i strukturert tekst. Utgangene til en blokk kan kobles sammen utgangene til en annen, og slik kan man lage en betinget kode. Det er et godt språkalternativ dersom man ikke har mye erfaring med tekstbasert programmering.

3.2.3. Ladder Diagram (LD)

Ladder er et logisk grafisk programmeringsspråk kalt ladder av den enkle grunn at når det programmeres flere linjer blir programmet seende ut som en stige. Man starter å lese av øverst og følger strømmen fra venstre til høyre og så nedover i programmet. Språket hermer etter koblingsskjemaer med logiske kontakter som simulerer åpning eller lukking av reléer. Man kan også programmere inn funksjoner som tellere, timere, skiftregistre og andre operasjoner i den logiske strukturen på lik linje som med andre programmeringsspråk som FBD og ST.

3.3. Siemens PLS programvare

3.3.1. TIA Portal

TIA Portal er et verktøy fra Siemens som gir en tilgang til SIMATICs *controllere* og tilbyr et stort utvalg av funksjoner for disse *controllerne*. Verktøyets kjernefunksjon er konfigurering og programmering av PLSer, med en programmeringsflate som støtter både strukturert tekst, funksjonsblokkdiagram og ladder diagram. TIA Portal gir også mulighet for monitorering av programmene som kjøres på de tilkoblede enhetene, slik at brukeren kan følge med på de aktive prosessene som pågår. Verktøyets grensesnitt gir oversikt over de ulike PLSene og funksjonene som utvikles, og tilbyr enkel manøvrering mellom de forskjellige enhetene som er tilkoblet.

TIA Portalkan også brukes for programmering *HMI displayer* der man da kan legge til knapper og funksjoner for manuell styring og kontroll av prosesser, samt funksjoner for monitorering av disse.

3.3.2. TIA Openness

TIA Openness er et API-grensesnitt som benyttes for å få tilgang til TIA Portal eksternt. APIet tilbyr en rekke funksjoner som kan kjøres i et skript, og gjør det mulig å utvikle komplekse programmer for automatisering og dermed effektivisering av oppgaver i TIA Portal. Grensesnittets hensikt er å være et verktøy for utviklere til å skreddersy programmer med funksjoner som er tilpasset sitt bruk, som for eksempel generering av prosjekter i TIA Portal eller arbeid på prosjekter eksternt.

Operasjonene som kan utføres i TIA Portal ved bruk av TIA Openness er generering, modifisering, avlesning og sletting av prosjektdata, samt gjøre prosjekter tilgjengelige for bruk i annen programvare.

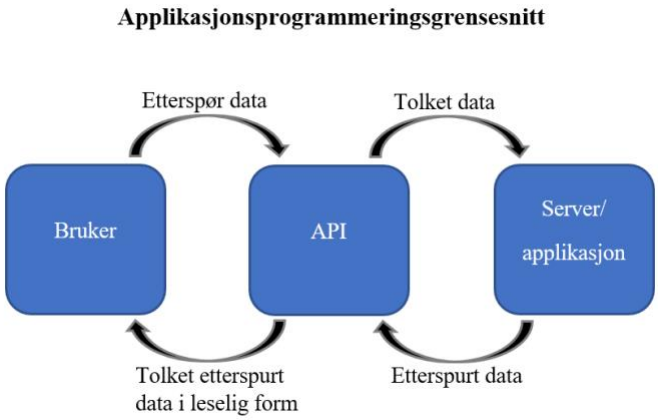
Utviklingen av et program med TIA Openness gjøres i programmeringsspråket C# gjennom Visual Studio, som er et utviklingsmiljø fra Microsoft. Selve grensesnittet installeres sammen med TIA Portal og gjøres tilgjengelig via en .dll-fil som linker TIA Portals struktur og funksjoner sammen med programmet man utvikler.

3.3.3. API

API står for Application Programming Interface. En API er en programvare som gir to separate applikasjoner lov og mulighet til å "prate med hverandre".
I enkle steg fungerer et API slik:

1. Bruker ønsker å bruke en applikasjon, denne sender data til for eksempel en server gjennom APIet.
2. APIet behandler og tolker dataen slik at serveren forstår dataen som kommer fra applikasjonen.
3. Serveren sender så tilbake dataen som først er etterspurt, igjen gjennom APIet.
4. APIet behandler og tolker denne dataen slik at den er lesbar for bruker.

Med bruk av API legger man i tillegg på en liten sikkerhet da man kommuniserer i små datapakker. Slik deles kun nødvendig informasjon for de ulike oppgavene, som gjør at større operasjoner skjer i flere små steg før all informasjon er utvekslet.



Figur 2: Viser trinn for trinn hvordan APIen tolker data i begge retninger

3.4. Skytjenester

Skytjenester er en vid betegnelse på ulike IT-ressurser som leveres over internett av ulike profesjonelle leverandører. Når vi tar i bruk skytjenester interagerer vi med eksterne servere over internett. Tjenestene kan være lagring av data, epost, strømmingstjenester, virtuelle maskiner, tilgang til programvare/applikasjoner og lignende. Dette utvalget av ressursene eller servere refereres til som skyen (eng: *the cloud*), og typisk for disse er at de leveres på kundeforespørsel. Kunden forsyner seg og betaler for tjenestene den bruker, og gjør avkall på disse når det ikke er lenger behov for dem. Slike tjenester er også fordelaktige for organisasjoner og diverse virksomheter da de fraskriver byrden av å vedlikeholde og oppgradere teknologien.

3.4.1. Tjenestemodeller

Skytjenester kan deles opp i tjenestemodeller, og det finnes tre ulike typer:

"Software as a service" (SaaS)

SaaS er en tjenestemodell der applikasjoner blir levert til brukerne gjennom Internett. Brukeren har tilgang til applikasjonene uansett hvor de befinner seg og til enhver tid, så lenge de har tilgang til Internett.

Brukeren har begrensninger på hva de kan kontrollere. De har blant annet ikke kontroll over det som skjer med applikasjoner, servere og nettverk, men slipper å måtte tenke på det tekniske, da det er noe tredjepartsleverandører ordner og juster ved behov.
Eksempler på skytjenester med SaaS er **Google Drive og Dropbox.**

"Platform as a service" (PaaS)

PaaS er en tjenestemodell der en leverandør gir brukeren en plattform de kan benytte seg av for å utvikle, administrere og kjøre sine egenlagde applikasjoner. For å lage applikasjoner kan brukeren bruke et støttet programmeringsspråk eller bestemte verktøy som er tilgjengelige.

Ved å bruke PaaS har ikke brukeren muligheten til å kontrollere faktorer som nettverk, lagring og servere, men har kontroll over applikasjonene de selv har laget.
Brukerne får tilgang til skytjenester som er på plattformen gjennom en nettleser så lenge de har tilgang til Internett.
Eksempler på skytjenester med PaaS er **Microsoft Azure** og **IBM Cloud**.

"Infrastructure as a service" (IaaS)
IaaS er en tjenestemodell der en leverandør gir brukerne sine en datainfrastruktur som de kan styre på egenhånd. Brukere har blant annet muligheten til å kontrollere servere, nettverk og lagring. Dessuten tilbys det ulike tjenester som de kan bruke ved behov. Ved å benytte IaaS brukes "pay as you go", der brukerne selv velger og betaler for det de selv ønsker å ha av tjenester og produkter som tilbys leverandøren. På den måten slipper de å måtte betale for tjenester som de ikke trenger.

Forskjellig fra de to andre tjenestemodellene er at brukere av IaaS har mer ansvar enn det som bruker SaaS og PaaS. De må blant annet holde styr på data, applikasjoner og mellomvare. Leverandøren har ansvaret for servere, nettverk, lagring, harddisk og virtualiseringen.

Brukerne får tilgang til skytjenester som benytter IaaS gjennom en nettleser så lenge de har tilgang til Internett.
Eksempler på skytjenester med IaaS er **Amazon Web Services (AWS)** og **Microsoft Azure**.

3.4.2. Leveransemodeller
Man kan dele skytjenester i tre ulike leveransemodeller.

Private skytjenester er skytjenester som kun er tilgjengelig for den kunden som benytter den. I de fleste tilfeller er det store bedrifter som benytter denne type skytjeneste. Bedriften kan velge om den skal være fysisk plassert hos bedriften sammen med alle komponentene, eller om den skal brukes gjennom en tredjepartsleverandør. Bruken av skytjenesten foregår på et privat nettverk, og på den måten vil dataen være beskyttet fra andre som ikke har tilknytning til det.

Offentlige skytjenester er skytjenester som er tilgjengelige for alle de som måtte ønske å bruke dem, gjennom internett. Ved å benytte seg av en offentlig skytjeneste slipper man å måtte tenke på den fysiske delen av skytjenester, da dette er noe leverandøren av skytjenesten kontrollerer og vedlikeholder.

En offentlig skytjeneste bruker et samlet lagringsområde som alle brukere av skytjenesten deler. Dette kan være mindre sikkert enn med en privat skytjeneste, men basert på hvilke sikkerhetsmessige tiltak leverandøren tar trenger ikke det å være et problem for brukerne.

Hybrid skytjeneste er en skytjeneste som er blanding mellom privat og offentlig. Med en slik skytjeneste har brukeren muligheten til å sende applikasjoner og data mellom privat skytjeneste og den offentlige. Dette gjør at man kan benytte tjenestene som begge typer av skytjenestene tilbyr.

3.4.3. Aktuelle skytjenester
Det er fordelaktig at vi velger en skytjeneste som er med på å gjøre implementering av oppgaven uproblematisk og effektiv. Det er også gunstig at skytjenesten og tjenestene som leveres mot betaling er tilbudt til en best mulig pris.
Det er spesielt fire skytjenesteleverandører som har utmerket seg for å være anerkjent på markedet, som vi skal se på:

- Google Cloud
- IBM Cloud
- Micosoft Azure (MS Azure)
- Amazon Web Services (AWS)

Grunnen til at vi fokuserer på større leverandører er fordi de har etablert flere ressurser, for eksempel i form av flere datasentre rundt om i verden, flere tjenester og en voksende og kompetent IT-support. Samtidig er brukergruppen større og dermed er det en større sannsynlighet for mer og lett tilgjengelig informasjon om feilsøking og tilrettelegging, og personer som har hatt erfaring med lignende prosjekter. De fleste leverandørene gir muligheten til å justere og endre betalingsmodellen i skytjenesten alt etter brukers behov.

Ved valg av en skytjenesteleverandør ønsker vi å se hva skytjenestene tilbyr av brukervennlighet, sikkerhet, pris, lagring og support. Med sikkerhet tenker vi på alt fra vakthold rundt de fysiske servere/datasentre til valg av ulike typer digitale sikkerhetstiltak og støttede krypteringsmetoder. Det er også behov for forskjellige betalingsplan, alt etter hvor stort et prosjekt skal være med varierende antall enheter som skal laste opp vekslende mengde med data. Fordi eventuelle prosjekter har ulike behov for tjenester, som visualisering, behandling og sortering av data, er det en fordel at man kun betaler for det man bruker.

Support er også relevant som en forebygging mot eventuelle kritiske situasjoner av ulik grad og som en kunnskapsressurs på et relativt ukjent område. Graden av support skyleverandørene tilbyr påvirker også brukervennligheten til tjenestene og produktene deres, i form av tidligere løste saker og FAQ.

Tabell 3
Hva Google Cloud tilbyr på de ulike kategoriene

Google Cloud	
Sikkerhet	Google Cloud bruker disse sikkerhetstiltakene på skytjenestene sine: **OAuth** OAuth gjør det mulig for brukeren å logge seg inn på skytjenesten ved å benytte en tredjepartsapplikasjon. Et eksempel på en annen applikasjon som bruker OAuth som innloggingsalternativ er Spotify. Med Spotify kan brukeren velge å logge seg inn ved å benytte tredjepartsapplikasjonen Facebook. Ved at skytjenesten har OAuth gjør det at brukeren slipper å måtte bruke et brukernavn og passord, dette gjør at det blir mer sikkert når brukeren skal logge seg inn. Det gjør også at brukeren slipper å måtte huske på brukerinformasjonen sin. **To-faktor autentisering** To-faktor autentisering fungerer ved at brukeren logger seg inn på skytjenesten ved å benytte et brukernavn og passord. Etter å ha oppgitt sin brukerinformasjon vil brukeren få en kode på enten tekstmelding eller på en app. Denne kode bruker de for å kunne logge seg inn på skytjenesten. Koden er tilfeldig hver gang de logger inn, slik at brukeren ikke står i fare for at andre skal kunne vite den. **Kryptering av data**

	Når brukeren laster opp dataen sin på Google Cloud vil den bli delt opp i mange deler. Hver eneste del av dataen deretter bli kryptert med sin egen nøkkel. Denne nøkkelen blir igjen kryptert med egen nøkkel. Ved å benytte GCP-KMS *(Google Cloud Platform – Key Management Service)* har brukeren muligheten til å kunne styre hvordan nøklene for dataen skal brukes. Google Cloud tilbyr også sikkerhetsprodukter som brukeren kan enten velge å prøve gratis i en viss periode eller kjøpe.
Prising	**Pay as you go** Når brukeren bruker denne skytjenesten har de muligheten til å velge selv hvilke ressurser de har lyst til å kjøpe og når de har lyst til å kjøpe dem. Dette gjør at brukeren selv kan styre hvordan de har lyst til å benytte skytjenesten og på den måten slipper å måtte forholde seg til en bestemt måte å bruke skytjenesten på. **Free trial** *Free trial* gir kunder som har lyst til å prøve ut Google sin skytjeneste muligheten til å gjøre det uten å betale. I 12 måneder får brukeren bruke alle tjenestene som Google tilbyr, med $300 i kredit som de kan bruke på tjenestene de ønsker. Dette tilbudet varer i 12 måneder eller til man har brukt opp krediten man fikk. Etter det vil brukeren få muligheten til å bruke skytjenesten gratis med tilgang til visse ressurser.
Lagring	Google Cloud har tre ulike kategorier for lagring av data. *Hot storage*, *Nearline* og *Coldline*. Disse kategoriene er basert på hvor ofte brukeren trenger tilgang til dataen de har lagret på skytjenesten. **Hot storage** Hvis brukeren trenger tilgang til dataen ofte, vanligvis et par ganger i måneden, er dette kategorien som passer best til det. Hot storage blir også delt opp i to underkategorier. Multiregionalt er data som brukeren har lastet opp er lagret i ulike datasentre i verden, ikke i et bestemt land. Regionalt er data som brukeren har lastet opp er lagret i et bestemt land, men er fordelt i ulike datasenter i det landet. **Nearline** Hvis brukeren trenger tilgang til dataen litt mindre enn med hot storage, vanligvis par ganger i året, er dette kategorien som passer best til det. **Deadline** Hvis brukeren trenger tilgang til dataen sin sjeldent, mindre enn en gang i året, er dette kategorien som passer best til det. Å lagre data på skytjenesten vil koste penger. Prisen er basert på tre faktorer: 1. Hvor ofte brukeren trenger tilgang til data de ha har lastet opp 2. Hvor mye data som de har lyst til å laste opp

	3. Om det er regionalt/multiregionalt
Support	Google Cloud tilbyr fire ulike typer av support: **Basic** Basic support er tilgjengelig for alle de som benytter Google Cloud og er gratis. Med basic har brukeren muligheten til å få hjelp gjennom internettet, telefonen og chat. Hjelpen brukeren kan få er begrenset til fakturering og salg. **Development** *Development* support har en minste pris på $100 i måneden. Med development har brukerne tilgang til samme tjenester som med basic og hjelp til tekniske problemer 24/5 med svar innen 4 arbeidstimer. **Production** *Production* support har en minste pris på $250 i måneden. Production gir brukeren tilgang til samme tjenester som med basic og development, men med raskere responstid og tilgang til hjelp 24/7. **Enterprise** Enterprise support har en minste pris på $15 000 i måneden. Enterprise gir brukere tilgang til samme tjenester som de tre tidligere nevnt supporttjenestene, men med raskere responstid, tilgang til hjelp 24/7 og personlig hjelp av Google Cloud-ansatte for å utnytte skytjeneste på en best mulig måte. Google Cloud har også gratis dokumenter og filer på hjemmesiden sin som brukerne kan bruke.
Nettverk/ datasenter	Google Cloud har 20 regioner med datasenter fordelt i verden, hvor de fleste ligger i Nord-Amerika, sentral Europa og Øst-Asia.

Note. Informasjonen er hentet fra Google Cloud sine nettsider

Tabell 4
Hva IBM Cloud tilbyr på de ulike kategoriene

IBM Cloud	
Sikkerhet	IBM Cloud gir brukerne sine ulike typer sikkerhet, de benytter blant annet: **OAuth** **To-faktor autentisering** **Data kryptering** IBM Cloud tilbyr også sikkerhetsprodukter som brukerne kan enten velge å prøve gratis i en viss periode eller kjøpe.
Prising	**Pay as you go** **Free lite:** Med *free lite* har brukeren muligheten til å bruke IBM Cloud gratis uten noen begrensningstid og uten å måtte bruke kredit kort for å kunne starte en "free

	lite" bruker. "Free lite" gir brukeren tilgang til visse tjenester og ressurser som IBM Cloud tilbyr sine kunder.
Lagring	IBM Cloud har til sammen sju kategorier for lagring av data, som blir delt opp i hvor mange steder man har lyst til å lagre dataen og hvor ofte man skal benytte den. For hvor mange steder brukerne har lyst til å lagre dataen, har IBM tre kategorier. **Cross Region:** Med *cross region* blir dataen til brukeren lagret i tre ulike regioner i et bestemt land. **Regional:** Med *regional* blir dataen til brukeren lagret i flere datasenter lokaler i en bestemt region. **Single-datacenter:** Med *single-datacenter* blir dataen til brukeren lagret i flere enheter i et bestemt datasenter. For hvor ofte brukerne skal benytte dataen sin, har IBM fire kategorier: **Standard:** Hvis brukeren trenger tilgang til dataen ofte, vanligvis par ganger i måneden, er dette kategorien som passer best til det. **Vault:** Hvis brukeren trenger tilgang til dataen litt mindre enn med "standard", vanligvis par ganger i året, er dette kategorien som passer best til det. **Cold-Vault:** Hvis brukeren trenger tilgang til dataen sin sjeldent, mindre enn en gang i året, er dette kategorien som passer best til det. **Flex:** Med Flex har brukeren mulighet til å variere fra måned til måned hvor ofte de har lyst til å benytte dataen de har lagret . Å lagre data vil koste penger. Prisen er basert på fire faktorer: 1. Hvor ofte brukeren trenger tilgang til data de ha har lastet opp 2. Hvor mange steder man har lyst til å lagre data 3. Hvor i verden brukeren befinner seg 4. Hvor mye data som de har lyst til å laste opp
Support	IBM Cloud tilbyr tre typer for support: **Basic** Basic support er tilgjengelig for alle de som benytter IBM Cloud sin 'pay as you go' eller har et abonnement. Basic gir brukeren muligheten til å få hjelp gjennom internett og telefon. **Advanced** Advanced support har en minste pris på $200 i måneden. Med advanced får brukeren tilgang til samme tjenester som med basic, men med raskere responstid. **Premium** Premium support har en minste pris på $10 000 i måneden. Med premium får brukeren tilgang til samme tjenester som med advanced, med enda raskere responstid. Dessuten får brukeren tilgang til personlig hjelp av IBM Cloud ansatte for å styre skytjeneste brukeren på en best mulig måte.

Nettverk/ datasenter	IBM har 60 datasentre over verden fordelt i størst grad over Nord-Amerika, sentral Europa og Øst-Asia

Note. Informasjonen er hentet fra IBM Cloud sine nettsider

Tabell 5
Hva Microsoft Azure tilbyr på de ulike kategoriene.

Microsoft Azure	
Sikkerhet	Azure gir brukerne sine ulike typer sikkerhet som blant annet: **To-faktor autentisering** **Data kryptering** MS Azure tilbyr også sikkerhets produkter som brukerne kan enten velge å prøve gratis i en viss periode eller kjøpe.
Prising	**Pay as you go** **Free Trial:** Med *free trial* får brukeren gratis tilgang til visse tjenester og $200 kredit som de kan bruke på det de ønsker. Etter at 12 måneder er over kan brukeren fortsatt benytte 25 andre tjenester som MS Azure tilbyr uten noe tidsbegrensning.
Lagring	Azure baserer lagring på hvor ofte man skal bruke dataen og har delt det opp i tre ulike kategorier: **Hot:** Baserer seg på at man skal bruke dataen flere ganger i måneden, hvor det vil koste mer for lagringen enn for selve bruken av dataen. **Cool:** Baserer seg på at bruker har et variert bruk av data fra måned til måned. **Archive:** Med *archive* er det sjelden bruk av dataen som er tatt til grunne. Hver av disse kan bli delt i to kategorier: **GRS:** Med GRS blir dataen lagret i flere nettverkssenter lokaler. Dette gjør denne løsningen dyrere enn LRS. **LRS:** Med LRS hvor dataen blir lagret i et bestemt nettverkssenter. Prisen for lagring er basert på tre faktorer: 1. Hvor ofte skal man bruke data. 2. Hvor mange steder vil man lagre data. 3. Størrelse på data lastet opp.
Support	Microsoft tilbyr fem ulike supporttjenester: **Basic** Basic er gratis for alle Azure brukere, dette inkluderer kun hjelp med salg og fakturering på telefon. **Developer** For $29 i måneden, får man i tillegg går man begrenset tilgang til support ingeniører vie email. Ubegrenset med kontakter og saker. Man får også

	interoperabilitet, konfigurasjonsveiledning og feilsøking fra en tredjeparts programvarestøtte. Raskere responstid og generell arkitekturstøtte **Standard** For $100 i måneden får man alt i Developer pakken, pluss 24/7 tilgang til support ingeniører via email og telefon, og raskere responstid for forskjellige forretningsvirkninger **Professional Direct** For $1000 I måneden får man alt i Standard pakken, i tillegg får man ekstra arkitekturstøtte, Onboarding tjenester, service vurderinger og konsultasjoner. Man får også webseminarer og proaktiv veiledning. **Premier** Denne pakken gir bruker alt fra de tidligere pakkene, men den inkluderer også enda raskere responstid, brukerspesifikk arkitektursupport. Gir også teknisk kontoadministrator-ledende serviceanmeldelser og rapportering. Man får en utpekt teknisk kontoadministrator. Denne pakken må man derimot kontakte Microsoft Azure for å få en pris på.
Nettverk/ datasenter	Microsoft har nå datasentre i 54 regioner over hele verden og er tilgjengelig i 140 land, de fleste er plassert i Nord-Amerika, sentral Europa, India, Øst-Asia og Australia

Note. Informasjonen er hentet fra Microsoft Azure sine nettsider

Tabell 6
Hva AWS tilbyr på de ulike kategoriene.

AWS	
Sikkerhet	AWS gir brukerne sine ulike typer sikkerhet som blant annet: **To-faktor autentisering** **Data kryptering** **Datasentre med høysikkerhetsstatus**
Prising	**Pay as you go** **Free Trial:** Med *free trial* har brukeren tilgang til visse tjenester gratis i 12 måneder. Etter 12 måneder får brukeren tilgang til andre gratis tjenester som AWS tilbyr uten noe tidsbegrensning. AWS tilbyr også to andre løsninger: **Save when you reserve:** Med denne kan man investere i reservert kapasitet. Jo større for betaling jo desto større tilbud. **Pay less by using more:** Med denne betaler du mindre jo mer data du bruker.
Lagring	AWS baserer lagring på hvor ofte man skal bruke dataen og har delt det opp i tre ulike kategorier: **Standard:** Baserer seg på at man skal bruke dataen veldig ofte.

	Standard-IA: Baserer seg på at bruker har et variert bruk av data fra måned til måned. **One Zone-IA:** Med One Zone-IA er det sjelden bruk av dataen som er tatt til grunne. Denne lagres kun i en tilgjengelig sone. Prisen for lagring er basert på tre faktorer: 1. Hvor ofte skal man bruke data. 2. Hvilken region brukeren har lyst til å laste opp i. 3. Størrelse på data lastet opp.
Support	AWS tilbyr fire ulike supporttjenester i tillegg til et forum: **Basic** Basic er gratis for alle AWS brukere, men inkluderer kun hjelp med salg og fakturering på telefon. **Developer** For $29 i måneden, får man i tillegg går man begrenset tilgang til support ingeniører vie email. Ubegrenset med saker og begrenset kontakter. **Business** For $100 i måneden får man alt i Developer pakken, pluss Ubegrenset med kontakter og saker, 24/7 tilgang til support ingeniører via email og telefon, Man får også interoperabilitet, konfigurasjonsveiledning og feilsøking fra en tredjeparts programvarestøtte. Raskere responstid og bruker basert arkitekturstøtte. I tillegg får man tilgang på AWS' support API **Enterprise** For $1500 I måneden får man alt i Business pakken, i tillegg får man ekstra arkitekturstøtte, Onboarding-tjenester, service vurderinger og konsultasjoner. Man får også webseminarer og proaktiv veiledning.
Nettverk/ datasenter	AWS er tilgjengelig i 60 soner innenfor 20 regioner, med datasentre spredd mest utover kysten av Nord-Amerika, sentral Europa og kysten av Øst-Asia

Note. Informasjonen er hentet fra AWS sine nettsider

Tabell 7
Oppsummering av skytjenestene

Skytjeneste	Google	IBM Cloud	Microsoft Azure	AWS
Sikkerhet	OAuth To-faktor autentisering Data kryptering Produkter for sikkerhet	OAuth. To-faktor autentisering Data kryptering Produkter for sikkerhet	To-faktor autentifisering Data kryptering Advanced-encryption standard	To-faktor autentisering. Data kryptering
Prising	Pay as you go Free trial	Pay as you go Lite (free) 'Subscription'	Pay as you go Free trial i 12 måneder med $200 i kredit.	Pay as you go Save when you reserve Pay less by using more Free trial i 12 måneder
Lagring	Lagring av data er basert på hvor ofte man skal bruke den: • Hot storage • Nearline • Coldline	Lagring av data er basert på hvor mange steder den skal lagres: • Cross-region • Regional • Single data center og hvor ofte man skal bruke dataen • Standard • Vault • Cold vault • Flex	Lagring av data er basert på hvor ofte man skal bruke dataen: • Hot • Cool • Archive Hver av dem kan igjen bli delt opp i to kategorier: • GRS • LRS	Lagring av data er basert hvor ofte man skal bruke dataen: • Standard • Standard-IA • One Zone-IA
Support	Gratis dokumenter og filer på nettsiden. Fire ulike supporttjenester: • Basic • Development • Production	Gratis dokumenter og filer på nettsiden. Tre ulike supporttjenester: • Basic • Advanced • Premium	Gratis dokumenter og filer på nettsiden. Fem ulike supporttjenester • Basic • Developer • Standard	Gratis dokumenter og filer på nettsiden. Forum Fire ulike supporttjenester • Basic

	• Enterprise		• Professional Direct • Premier	• Developer • Business • Enterprise
Data-sentere	20 datasenter fordelt i verden.	60 datasentre fordelt i verden.	Har 54 regioner over hele verden og er tilgjengelig i 140 land.	Har 60 soner innenfor 20 regioner i verden

3.4.4. Konklusjon for valg av skytjeneste

Etter å ha gjort en vurdering og sett igjennom hva skytjenestene hadde å tilby for hvert av kategoriene, kom vi frem til at det er lite som skiller dem fra hverandre i disse kategoriene. Det er derimot små detaljer som var med på å bestemme endelig valg av skytjenesten som skulle brukes i oppgaven.

Når det gjaldt sikkerhet var dette noe som alle skytjenestene hadde i felles, ved at de tok i bruk like typer sikkerhetsmodeller. Grunnet været dette var sikkerhet ikke en stor faktor ved valg av skytjeneste. Dessuten er sikkerhetsmodellene som tas i bruk av skytjenestene sikre nok til at det ikke skulle være en stor risikofaktor for oss senere i oppgaven.

Det som derimot skilte skytjenestene fra hverandre var prisen ved bruken av dem. Dette gjaldt kategoriene *prising*, *lagring* og *support*. Siden vi ved valg av skytjeneste skulle vurdere prismodellene til skytjenestene var det her viktig at vi gikk i detalj på hva prisene var for disse tre kategoriene. Etter å ha gjort en vurdering kom vi frem til at *Microsoft Azure* var den skytjenesten som brukere måtte betale minst for ved bruken av deres produkt og tjenester. På grunn av at *Azure* tilbød sitt produkt og tjeneste til en minst mulig pris, av de skytjenestene vi valgte å ta hensyn til, valgte vi derfor å ta i bruk *Azure* videre i vår oppgave. Vi fant gjennom å bruke *Azure* til vårt formål at den ikke tilegnet seg for vår oppgave. Den var ikke brukervennelig for oss som ikke har erfaring fra tidligere, dessuten støtte ikke *Azure* ukryptert kommunikasjon ved opplastning av data.

Grunnet været at *Azure* ikke var godt nok tilegnet for vår oppgave valgte vi å heller ta i bruk *IBM Cloud*. Vi hadde ved vurdering av skytjenestene lest at *IBM* er en skytjeneste som egner seg godt ved opplastning av data til en skytjeneste. Vi skulle ved opplastning av data ta i bruk *MQTT* protkollolen. Dette passet godt med *IBM* da de er selskapet som opprinnelig har laget *MQTT*, som gjør at skytjenesten er tilrettelagt for kommunikasjon over denne protokollen. Når det gjelder *IBM* som skytjeneste generelt kom vi frem til at det ikke er stort forskjellig fra *Azure* med tanke på prisingen, tilgjengeligheten og support.

Grunnet være at *IBM* er såpass tilrettelagt for kommunikasjon ved bruk av protokollen *MQTT* og at det er en skytjeneste som tilbyr sitt produkt og tjenester til en god nok pris valgte vi derfor å ta i bruk *IBM* videre i oppgaven vår.

3.5. GSM
GSM er et globalt og digitalt mobiltelefonisystem som brukes til overføring av både lydsignaler og annen data. Systemet opererer over radiobølger i UHF-båndet (ultrahøy frekvens), som er elektromagnetisk stråling mellom 300MHz og 3GHz. Nettverket er opprettholdt av celletårn, eller nettverksbaser.

En GSM-ruter gjør det mulig å koble seg til det mobile nettverket internasjonalt, gitt det er dekning der den befinner seg. En slik ruter bruker et SIM-kort på lik linje som en mobiltelefon. Vi skal bruke en ruter av denne typen slik at det skal være mulig å laste opp data fra måleenheter og systemer som ligger avsides.

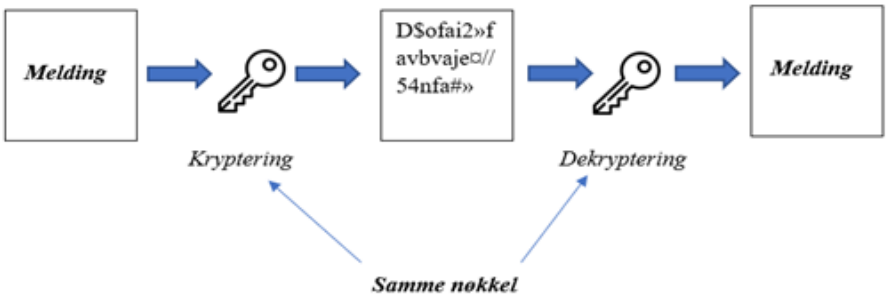
3.6. Kryptering
At man krypterer data betyr at man ved hjelp av algoritmer gjør dataen som blir sendt av en adressant uleselig for uvedkommende, ved at den blir seende ut som tilfeldige symboler og tall. Dataen blir leselig igjen ved å dekryptere den med en bestemt nøkkel.

Det finnes to krypteringsmetoder som er vanligst å ta i bruk: Offentlig nøkkel kryptering (asymmetrisk kryptering) og privat nøkkel kryptering (symmetrisk kryptering).

Ved kryptering og dekryptering brukes nøkler, som består av et bestemt antall rad med bits. Nøklene man kan bruke er offentlig nøkkel og privat nøkkel.

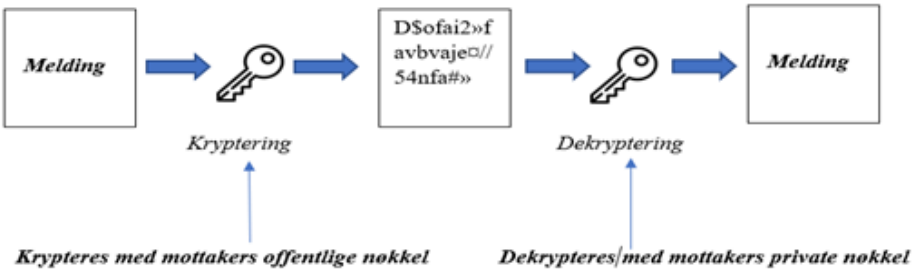
- Offentlig nøkkel: Hver enkel person har en offentlig nøkkel. Denne type nøkkel er tilgjengelig for hvem som helst.
- Privat nøkkel: Privat nøkkel er en nøkkel som kun en person har tilgang til, men det er noen tilfeller der et visst antall person deler samme private nøkkel.

Privat nøkkel kryptering, eller symmetrisk kryptering, fungerer ved at avsender av data og mottakeren bruker en felles privat nøkkel som kun de har. Når avsender sender data vil de kryptere den med den private nøkkelen og når dataen har kommet frem til mottaker vil de dekryptere (åpne) den med den samme private nøkkelen.



Figur 3: Figuren viser steg for steg hvordan «privat nøkkel kryptering fungerer»

Offentlig nøkkel kryptering, eller asymmetrisk kryptering, fungerer ved at dataen som blir sendt fra en avsender til en mottaker blir kryptert med den offentlige nøkkelen til mottakeren. Når dataen har nådd frem til mottakeren vil den bruke sin private nøkkel for å dekryptere dataen.



Figur 4: Figuren viser steg for steg hvordan «offentlig nøkkel kryptering» fungerer

Det er i dag ulike algoritmer som brukes for å kryptere dataen som blir sendt. De mest benyttet algoritmene som protokollene tar i bruk er 3DES, AES og RSA.

Det disse algoritmene har til felles er at de tar i bruk nøkkelstørrelser. Nøkkelstørrelsen baserer seg på hvor mange bit nøkkelen som brukes for å kryptere dataen med består av. Jo flere bit nøkkelen består av, jo flere kombinasjoner av forskjellige nøkler vil det være der én av de er den gyldige for øyeblikket.

3.7. VPN

VPN står for *Virtual Private Network*. Som navnet tilsier er VPN et privat nettverk av maskiner eller andre separate nettverk som opererer over internett. VPN blir oftest brukt som et sikkerhetsverktøy, for å forsikre at det man sender fra sin ende til en annen over et offentlig nettverk er kryptert og holdt privat fra uønskede tilskuere. Andre brukseksempler kan være å kunne navigere anonymt på nettet, eller koble seg til en VPN *gatew comon ay* på et bedriftsnettverk.

Når vi snakker om gateways, tenker vi på en nettverksenhet som knytter sammen andre eksterne maskiner eller nettverk. En slik enhet er vanligvis en fysisk ruter, men kan også være andre lignende enheter som har en funksjon for dataoverføring (for eksempel en server). Med gateway konfigurerer man deriblant hvilke forbindelser som skal passere eller blokkeres.
Et virtuelt nettverk forklares også ofte som en *proxy* med kryptering. Med proxy menes det vanligvis en proxyserver som fungerer som en mellomstasjon der internettforespørsler behandles og leverer det klienten spør etter. Slik kan man være på nettet uidentifisert, komme seg rundt diverse restriksjoner eller filtrere nettinnhold, men standarden er at proxyforbindelsen er ukryptert.

VPN først ble brukt av bedrifter, slik at ansatte kunne få tilgang til filer fra ulike kontorlokaler på en sikker måte. Dette gjorde at en ansatt kunne sitte hjemme eller på et kontor og fortsatt få tilgang til filer i et annet kontor, uten å være i fare for at andre på Internett kunne få tak i filene fra bedriften.

VPN er nå tilgjengelig for alle de som har tilgang til Internett. Det finnes mange ulike VPN-klientprogramvare som brukere kan velge mellom. De fleste setter opp VPN for å kunne bruke Internett på en måte slik at informasjonen de deler ikke er tilgjengelig for andre, som kan være sensitiv økonomisk, medisinsk eller annen personlig data. Andre bruker VPN til å få tilgang til tjenester/nettsider som ellers er blokkert eller utilgjengelige i det landet de er.

Eksempel på dette kan være land der regjeringen har blokkert utvalgte nettsider, eller underholdningstjenester som er tilgjengelig i begrensede landområder.
For å bruke VPN må man laste ned en VPN-klient på enheten man bruker. De fleste VPN-klienter koster penger, men det finnes også noen som er gratis. Etter at VPN-klienten har blitt lastet ned må man vanligvis opprette en bruker.

Protokoller
Protokollene VPNer bruker fungerer som et sikkerhetstiltak for å forsikre avsenderen at dataen de sender kommer frem til mottaker uten at den blir avlyttet eller utnyttet av andre. Det finnes flere protokoller, men de som er mest brukt er:

- OpenVPN
- L2T/IPSec
- PPTP
- IKv2

Hvilken protokoll som brukes med VPNen er avhengig av faktorer som:

- Hvilken protokoll som VPN-klienten eller programmet som benyttes har gjort tilgjengelig for sine brukere å velge imellom.
- Hvilket operativsystem man bruker sammen med VPN-klienten/programmet, da det er noen protokoller som kun støtter bestemte operativsystemer.
- Hvordan sikkerheten som protokollen tilbyr er. Det kan være av betydning å velge noen protokoller ovenfor andre, da protokollene har ulike nivåer av sikkerhet.

Protokoller sikrer dataen til avsenderen ved å kryptere den. Dette gjør de ved å bruke ulike krypteringsalgoritmer. Hvilke algoritmer en protokoll bruker er forhåndsbestemt, men de vanligste algoritmene protokollen bruker er 128 og 256-bit kryptering. Dette er to krypteringsalgoritmer som ved bruk danner mange kombinasjoner om noen skulle prøve å dekryptere den avsendte dataen.

Krypteringsalgoritmen *256-bit kryptering* har for eksempel 2^256 = 1.1579 * 10^77 mulige kombinasjoner, og å teste alle disse kombinasjonene for å dekryptere dataen ville ha tatt urimelig lang tid. Derfor er algoritmene som protokollene tar i bruk veldig sikre mot å hindre andre i fra å utnytte dataen.

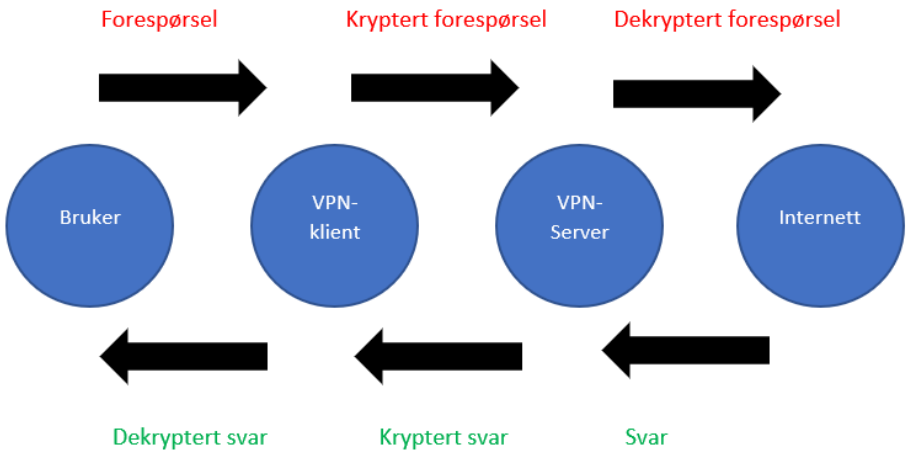
Server
Serverne som blir brukt i VPN er fysiske servere som er plassert i ulike deler av verden. Serveren som en bruker benytter bestemmes av brukeren på VPN-klienten.
VPN-serveren har ulike oppgaver, og de viktigste er:

- Serveren fungerer som en port (en mellomstasjon) til å frakte data inn og ut av enheten.
- Ved å bruke en server vil brukeren bli tildelt IP-adressen og lokalisasjonen til serveren brukeren har valgt. Dette gjør at andre, som for eksempel internettleverandøren til brukeren, ikke vil kunne vite hvilke nettsider og programmer brukeren har vært på via VPN-klienten, da de kun vil kunne se IP adressen og lokalisasjonen til serveren.
- Ved at brukeren velger å ta i bruk en server i et annet land vil det gjøre mulig for brukeren å få tilgang til tjenester i det landet serveren er plassert. Et eksempel på dette kan være hvis brukeren er i Norge og velger en server i Amerika. Ved å benytte

serveren som befinner seg i Amerika kan de få tilgang til visse tjenester som kun tilbys der.

Hvordan VPN fungerer kan deles opp i flere trinn, fra en forespørsel går fra klientenheten til den når internett, får svar på forespørsel fra internett og tilbake til enheten.

- Det begynner ved at en bruker starter VPN-klienten sin. Etter at VPN-klienten er startet vil brukeren kunne sende en forespørsel til internett.
- VPN-klienten vil registrere at brukeren prøver å sende en forespørsel og vil kryptere dataen som blir sendt ved bruk av en protokoll.
- Dataen blir sendt til en forhåndsbestemt server. Når dataen har nådd frem til serven vil den bli dekryptert av serveren og bli sendt til destinasjonen brukeren prøver å nå.
- Når dataen har nådd frem til destinasjonen vil brukeren få et svar tilbake. Da vil dataen som blir sendt tilbake til brukeren reise på samme måte. Den vil bli sendt til serveren som vil kryptere dataen. Den krypterte dataen vil bli sendt til brukeren sin enhet, der den vil bli dekryptert og vist fram til brukeren.



Figur 5: Viser trinn for trinn hvordan data blir behandlet ved bruk av en VPN-klient.

Man skiller vanligvis mellom to typer VPN. *Remote Access* en type VPN-tilkobling som er målrettet en mer privat bruk. En bruker kobler seg til et eksternt privat nettverk, med en kryptert tilkobling som kalles ofte for VPN-tunnel og får tilgang til alle nettverks ressurser. Serveren betegnes som RAS (Remote Access Server), og man trenger å laste ned programvare for å koble seg til denne.

Site-to-Site er en type VPN som benyttes av forretninger og større grupper med mennesker som har for eksempel lokaler i flere områder. Flere ansatte og medlemmer kan få tilgang til nettverket og dele filer med hverandre og likevel holde konfidensiell informasjon utilgjengelig for andre.

3.8. Kommunikasjonsprotokoller

Protokoller er sett med regler og standarder som lar enheter og systemer kommunisere med hverandre. De assisterer med å formatere og overføre data mellom uavhengige systemer til tross for sin særpregete struktur og egne standarder, og fungerer som et felles språk for disse. Brukerstøtten kan implementeres i program- og/eller maskinvare.

Det finnes ulike protokoller til bestemte formål og oppgaver. Større arbeidsprosesser er delt opp i nettverkslag i OSI-modellen, og det er protokoller som er laget for spesifikke aktiviteter innenfor et eller flere av lagene. Noen sett med protokoller samarbeider med hverandre om en felles oppgave på tvers av lagene, for eksempel om å opprettholde en internettforbindelse, og slike sett kalles en *suite*.

Valg av protokoll kan påvirke resten av prosjektet i form av ekstra tiltak som er nødvendig i hver ende av kommunikasjonen for at overføringen skal fungere. Det kan for eksempel være at skytjenesten ikke støtter protokollen eller krever noe form for oppsett før den aksepterer meldinger, og man må på samme måte sjekke at sendingen lar seg gjøre med både valgt ruter, PLS og programvare.

Vi skal se på de mest kjente og relevante kommunikasjonsprotokollene, hvordan de fungerer og avgjøre om de eventuelt passer til vårt formål.

MQTT er kjent for å være en enkel kommunikasjonsprotokoll da den krever lite båndbredde, prosessorkraft og er enkel og ta i bruk. Protokollen ligger på TCP/IP laget, som betyr at den benytter TCP/IP-protokollen for pakkeoverføring.

MQTT bruker *publish/subscribe messaging*, som medfører at klient og server kommuniserer med hverandre igjennom en *broker* (server). Brokeren fungerer som en mellomstasjon mellom de to endepunktene der klienten publiserer data til en broker, mens serveren abonnerer på dataen som *brokeren* mottar. Dette kan sammenlignes med et postkontor som behandler brev og pakker fra og til ulike destinasjoner. Metoden gjør det mer intuitivt å overføre data til flere servere med å opprette flere abonnenter til brokeren (krever lite konfigurasjon).

Med MQTT som overføringsprotokoll kan man gjøre kommunikasjonen sikker ved hjelp av kryptering med TLS og bruk av autentisering, autorisasjon og sertifikater. Til tross for at protokollen er enkel, tilbyr den noen funksjoner tas i bruk om ønsket. Eksempler på dette er to ulike grader av *QoS* (Quality of Service) *Last Will. QoS* går ut på at protokollen tar i bruk to-veis kommunikasjon mellom klient og server for å forsikre at meldinger blir sendt, mens Last Will sørger for at melding blir sendt fra en klient som er i ferd med å miste kontakten med mottaker.

MQTT ble laget for bruk i telemetri, som er årsaken til dens små krav om båndbredde. Telemetri er når man samler måledata fra et fjernt og utilgjengelig sted (for eksempel under havet) og sender dataene trådløst til en lagringsenhet, typisk over GSM.
Denne protokollen er ment for små, lette, mobile og innebygde enheter og systemer, med kostbar båndbredde og krav om et minimum overhead.

AMQP er en mer avansert overføringsprotokoll som tilbyr en rekke nyttige funksjoner. AMQP har mer *overhead* enn MQTT, og er kjent for sin interoperabilitet mellom klienter og brokere. Når vi snakker om overhead, mener vi tilleggsdata som er nødvendig med en melding, men som ikke bidrar til den indre betydningen av denne meldingen. Mellomvare-produkter kan kommunisere til tross for ulike plattformer og forskjellige språk. Protokollen benytter seg av *peer-to-peer*-overføring, altså direkte maskin-til-maskin, men kan også implementeres med en *broker* dersom dette er ønskelig, der rollen til den ene enheten kan konfigureres til å ha en funksjon som en broker.

AMQP defineres av sine egenskaper som meldingsorientering, *queuing* og *routing*. En del av brokeren tar imot meldinger og tilegner dem ulike meldingskøer. Dette elementet kalles *exchange*, og reglene for hvordan meldingene skal fordeles til meldingskøene betegnes som *bindings*. Prinsippet ligner veldig på mailing, men denne protokollen er spesielt rettet mot å levere servicekvalitet, QoS.

Med AMQP blir meldingene levert raskt og leveringen er garantert, da man vil få en bekreftelse på de meldingene som er mottatt. Behovet for protokollen oppstod fra bankindustrien. AMQP ble designet for et mangfold av meldingsapplikasjoner og for overføring av krypterte meldinger mellom disse og diverse forretninger med ulike interne systemdesign.

AMQP er ofte ikke foretrukket for lette og mobile enheter, grunnet at protokollen krever mer prosessering. I tillegg kan AMQP være avansert å sette opp med tanke på mange muligheter og funksjoner protokollen har å by på men ikke nødvendigvis har behov for.

AMQP er en åpen standard (åpen kildekode) protokoll for asynkrone meldingstjenester. Med asynkrone meldinger menes at kommunikasjonen er enveis. Når en melding blir sendt fra en klient, venter den ikke på et svar fra serveren. I synkron meldingsoverføring vil derimot klienten vente på at serveren svarer på at meldingen er mottatt, og klienten sender ingen ny melding før den har fått bekreftelse.

HTTPS er en utvidelse og en sikrere versjon av HTTP (Hypertext Transfer Protocol), en nettverksprotokoll som er grunnlaget (sammen med HTML) for *World Wide Web*. Kommunikasjonen er *request-response*-basert. Det vil si den foregår mellom en klient (vanligvis en nettside) og en tjener, ved at klienten sender noe og tjeneren responderer. Forbindelsen avsluttes vanligvis etter at forespørselen er blitt fullført.

På en måte er HTTPS ingen egen protokoll, men er HTTP kryptert over SSL- og TLS-protokollene. Begge er sikkerhetstiltak som skal bedre integriteten til dataen og motvirke tyvlytting, spesielt fra *man in the middle* angrep. Slike angrep kan komme ved at en noen snikleser meldinger eller får tak i sensitiv informasjon fra ruteren. HTTPS krever mer tid til å prosesseres, som innebærer at det trenger mer hardware og er dermed mer kostbart. Overheaden til HTTPS er større på grunn av SSL/TLS-*handshake*.

SSL står for *Secure Locket Layer*, og er egentlig en utdatert protokoll som er erstattet av TSL (Transport Layer Security). Protokollnavnene brukes fortsatt litt om hverandre, men dagens kryptering foregår egentlig vanligvis over TLS.

Meldingsprotokollen **SOAP** lar programmer som kjører på to ulike operativsystemer, kommunisere sammen med hjelp av HTTP og XML. SOAP støtter også andre transportprotokoller, som SMTP, men den forbindes ofte med disse to. XML (Extensible Markup Language) er et markeringsspråk slik som HTML, men skiller seg fra HTML med at XML beskriver data og hva det representerer der HTML fokuserer på hvordan dataen er fremstilt. På denne måten erstatter ikke XML HTML.

SOAP spesifiserer hvordan en HTTP-header og XML-fil skal kodes slik programmet på den ene maskinen kan kontakte programmet på den andre maskinen sånn at dataen kan sendes videre, og hvordan det kontaktede programmet skal respondere med et svar.

REST (Representational State Transfer) er ikke en protokoll, men en arkitekturisk stil å sette programvare opp på. Det vil si at det er retningslinjer for å etablere kommunikasjon mellom enheter.

Ved bruk av REST som verktøy bruker man en RESTful API til å konfigurere kommunikasjonen, med stort sett HTTP som overføringsprotokoll. API (Application Programming Interface) er et programmeringsgrensesnitt som gjør det mulig å bruke funksjoner fra et annet program i programmet du bruker eller utvikler selv. Et eksempel på dette kan være Google Maps API som gjør at man via programmering kan få tilgang til noen funksjoner fra Google Maps dersom man ønsker å bruke kartdata på en nettside man utvikler. Ved bruk av RESTful API får brukeren tilgang til ulike funksjoner for å sette opp kommunikasjonen. De sentrale funksjonene som tas i bruk ved kommunikasjon med REST er HTTP-forespørslene GET, PUT, POST og DELETE.

RESTful APIet kan brukes ved hjelp av flere ulike programmeringsspråk, og det skal kun trenges et få antall linjer kode for å sette opp ønsket kommunikasjon. Data kan leveres med blant annet ren tekst, HTML, YAML (YAML Ain't Markup Language), XML og JSON (JavaScript Object Notation).

SOAP er eldre og mer moden, har tydelige regler, avanserte sikkerhetsfunksjoner og er generelt en mer kompleks protokoll. Det krever dermed mer båndbredde. REST ble utviklet senere for å forbedre manglene ved SOAP, og er derfor en mer fleksibel måte med løsere regler å sette opp datakommunikasjon på.

XMPP er en åpen standard som er mye brukt i IoT-verdenen, og er også kjent som Jabber. XMPP støtter flere ulike kommunikasjonsløsninger som *request-response* direkte mellom to enheter og *publish-subscribe* for kommunikasjon via en *broker*. Ved hjelp av tilhørende utvidelser (XEP) kan XMPP implementeres for flere ulike kommunikasjonsscenarioer, som for eksempel IoT-løsninger (XEP-0323 mm.). Disse utvidelsene gir også fleksibilitet for brukeren til å tilpasse løsningen til sitt behov.

XMPP bruker XML som meldingsformat. XMPP gir mulighet for flere løsninger for sikring av kommunikasjonen som brukerautentisering, autorisasjon og kryptering med TSL og SASL. SASL står for *Simple Authentication Security Layer* og er et rammeverk som brukes for kommunikasjonssikkerhet. SASL tilbyr applikasjoner for bruk av autentisering, autorisasjon og kryptering som kan implementeres i oppsettet av en forbindelse.

STOMP er en enkel, tekst- og rammebasert protokoll, som ligner på HTTP. Protokollen ble laget for å være simpel og bredt anvendelig. STOMP tillater også overføring av binære meldinger. Den er mindre komplisert enn AMQP, og er også interoperabil mellom flere språk, plattformer og brokere. STOMP-klienter kan kommunisere med nesten alle tilgjengelige STOMP-brokere, og man kan for eksempel koble seg til en STOMP-broker ved å bruke en telnet-klient (programvare for telnet-klient). Telnet er en tekst-basert nettverksprotokoll som gjør det mulig å koble seg til eksterne maskiner over internett eller annet TCP/IP-nettverk, og som regnes som "det første internettet".

Denne protokollen brukes til overføring av data på tvers av applikasjoner, og når det ikke er store krav om komplekse meldingsutvekslinger og køer og kombinasjon av disse. STOMP fungerer best med enkle meldingsapplikasjoner.

I stedet for å håndtere køer og emner, som MQTT og AMQP, bruker STOMP en semantisk SEND med en *string* med destinasjon. (Semantikk håndterer betydningen som blir tildelt ord, karakterer og symboler, på lik måte som syntaks refererer til strukturen til koden til et programmeringsspråk.) Brokeren kartlegger informasjonen, men det er klientene som abonnerer på destinasjonene.

STOMP-servere har en mengde med destinasjoner som meldinger sendes til. Destinasjoner behandles som en *string*. STOMP-protokollen har ikke noe definisjon på hva leveringssematikken skal være for destinasjoner, og den kan variere mellom servere og til og med mellom destinasjonene.

Klienter kan gjøre to ulike ting gjennom to ulike moduser. I produsent-modus kan man sende meldinger til en destinasjon på serveren gjennom en *SEND*-ramme. Som en forbruker kan man abonnere på en eller flere av disse destinasjonene og motta meldinger fra serveren som meldings-ramme. En ramme består av en kommando, et sett med *headers* og en *body*.

Tabell 8
Sammendrag av kommunikasjonsprotokoller vi har undersøkt

	Vurdering			
Protokoll	Brukervennlighet	Sikkerhet	Båndbredde	Datakonsistens
MQTT *Message Queuing Telemetry Transport*	Enkel å implementere, få funksjoner og er godt dokumentert. Ideell for IoT-applikasjoner. Siemens har publisert en MQTT-bibliotek for TIA Portal (ukryptert for s7-1200). Pub/sub-messaging igjennom en broker.	Kan krypteres over TLS/SSL, som gjør overføringen sikker. (Port: 8883) Kan overføres med TCP/IP og gjør det enkelt å implementere med en PLS.	Svært små krav til båndbredde.	Kvitterer hver overføring som gir kontroll over evt. tapt data. MQTT hevder å være pålitelig, men har ikke nødvendigvis garanti for dette i praksis.
AMQP *Advanced Message Queuing Protocol*	Mer avansert da AMQP er designet for å brukes på større applikasjoner. Godt dokumentert. Støtte for peer-to-peer-kommunikasjon, som fjerner kravet om *broker* (men kan også brukes).	Autentisering og kryptering over TLS/SSL eller SASL som opprettholder kravene for sikkerhet. Overføres med TCP.	Stiller litt større krav til båndbredde.	Kvitterer hver overføring som gir kontroll over evt. tapt data. Har ulike redskaper som sørger for mindre tap av data.
REST *Representational State Transfer*	Fleksibel og enkel å forstå, gjenkjennbare konsepter fra HTTP. Kan støtter XML og JSON som meldingsformater. Applikasjoner kan skrives på alle språk.	Kryptering og dataintegritet baserer seg på TLS/SSL. Kan implementere flere sikre løsninger, som brukerautentisering,*t*	Stiller en god del større krav til båndbredde enn MQTT, men regnes fortsatt som en "lett" overføringsmetode.	Begrensningene til HTTP blir også svakheter ved REST. HTTP lagrer ikke state-basert informasjon mellom request-response sykluser.

	Stiller større krav til programmeringsferdigheter enn f.eks. AMQP og MQTT. Finnes en rekke programmer/verktøy som gjør REST-programmering enklere.	*oken passing* og sertifikater.		HTTP kan ikke sende push-varslinger fra server til klient (en-veis-kommunikasjon). *HTTP status code* gir status på overføringene. Frihet til å implementere løsninger som øker sikkerhet. (?)
HTTPS *Hypertext Transfer Protocol Secure*	En utvidelse av HTTP for sikker kommunikasjon over datanettverk. Lite dokumentasjon om bruk i PLS.	Kryptering over TSL/SSL.	Krever lite for å sette opp kommunikasjon mellom to enheter, men krever en del båndbredde ved publisering av meldinger.	Kan generere status-meldinger for dataoverføringene, men har ingen verktøy for å motvirke tap av pakker.
SOAP *Simple Object Access Protocol*	Meldingsprotokoll-spesifikasjon. Bruker XML til å definere innhold av en melding. Avhengig av applikasjonslagsprotokoller som HTTP eller SMTP for overføring.		Krever en del båndbredde pga. sin kompleksitet (i forhold til REST) og ekstra funksjoner.	
STOMP *Streaming Text Oriented Messaging Protocol* *(også kalt TTPM)*	Enkel tekst-basert protokoll. Laget for bruk med MOM. Interoperabelt, klienter kan kommunisere med hvilken som helst meldingsbroker som støtter protokollen.	Fungerer over TCP.		

	Ligner på HTTP.			
XMPP *Extensible Messaging and Presence Protocol* (også kalt Jabber) Åpen standard	Kommunikasjonsprotokoll for meldingsorientert middleware, laget for å være utvidbar (xMOM). Instant Messaging (IM). Meldinger blir sendt *real-time*. XMPP er en åpen standard med mange gratis løsninger for implementering av forskjellige systemer.	Kan bruke HTTPS i stedet for TCP som transportprotokoll for klienter bak brannmurer. XMPP-servere kan være isolert fra det offentlige nettverket (intranett). Bruker SASL og kanalkryptering ved hjelp av TLS.	Stiller større krav til båndbredde enn en del andre IoT-protokoller. Pakkehodet i en overføring kan være opptil 200 bytes, som er betydelig mye mer enn MQTT sitt pakkehode i kan være så lite som 2 bytes. Mulighet for kompresjon minsker størrelsen på en overføring betraktelig.	Har effektiv sanntids push-mekanisme, der eksisterende nettbaserte metoder ofte gjør overflødige forespørsler som innfører nettverksbelastning.

Både MQTT og AMQP er gode kandidater til vårt formål. Vi ønsker at protokollen vi bruker skal ha datakonsistens, være sikker og kreve liten båndbredde (mange overføringer over potensielt lang tid). MQTT er kjent for lite krav til båndbredde. Når det gjelder sikkerhet kan begge krypteres over SSL/TSL, som regnes å være svært sikkert og begge protokollene bruker *stateful* TCP-tilkobling for kommunikasjon. En *stateful* tilkobling et minne om en økt og lagrer midlertidig data, men *stateless* tilkobling har ingen og hver forespørsel til en server blir behandlet på samme måte og med samme respons.

Begge disse protokollene blir mye brukt for overføring av viktige data og skal dermed være robuste nok til å oppfylle vårt formål. AMQP kontrollsjekker både sender og mottaker (server og klient) som sørger for at data ikke går tapt, og gjør at dataene til slutt vil komme frem til mottaker.

Når det gjelder brukervennlighet skal begge protokollene være nokså enkle å håndtere, men MQTT er en enklere protokoll. MQTT krever at man setter opp en broker (mellomstasjon mellom klient og server), mens AMQP har benytter seg av *peer-to-peer*-kommunikasjon. AMQP er mer fleksibel dersom man må endre endepunkter i koblingen (for eksempel dersom en bedrift ønsker å bytte ut eller oppgradere et system).

Alt i alt er AMQP og MQTT svært like i hvordan de er bygd opp, men førstnevnte er betydelig mer avansert med flere funksjoner og muligheter. Dette kan være en stor fordel i mange tilfeller, men ikke nødvendigvis for oss, som ikke har behov for alle mulighetene AMQP tilbyr. Begge protokollene er mye brukt i bransjen, som antyder at de opprettholder kravene til sikkerhet og datakonsistens.

Fordi MQTT-protokollen er spesielt utviklet for bruk til telemetri, gjerne over en GSM-ruter, er det en naturlig beslutning for vårt prosjekt. Samtidig har protokollen små krav til båndbredde, noe som er en av de viktigere vurderingskriteriene. MQTT blir stadig mer brukt til ulike formål og vokser utenfor olje- og gassindustrien. Det er relativt lett å finne dokumentasjon og informasjon om implementering og feilsøking, da denne protokollen er hyppig brukt og har vært tilstede en stund nå.

4. Materialer og metoder

Løsningen blir at PLSen kommuniserer med IBM Cloud. Kommunikasjonen skal skje over MQTT-protokollen, som står for overføring av meldingsdata mellom klient og broker. Meldingen kommer i form av en tekststreng. I dette oppsettet fungerer PLSen som en klient og Watson IoT Platform både som en broker og endelig endepunkt.

4.1. Utstyrsliste

Tabell 9

Oversikt over nødvendig utstyr

Utstyrsliste	
Maskinvare	
Siemens S7-1200	PLS
SCALANCE M874-2	GSM-Ruter
Com4 1GB	Sim-kort til ruter
Programvare	
VMware Player *(VMware Fusion for MAC)*	Virtuell maskin
TIA Portal m/TIA Openness	Applikasjon/programvare
Microsoft Visual Studio	Programvare
Diverse	
IBM Cloud	Skytjeneste
MQTT versjon 3.1 (MQTT_Publisher)	Protokoll

Oppdragsgiver hadde tre ulike rutere tilgjengelig til bruk. Vi besluttet å bruke SCALANCE M874-2 da denne viste seg å være mest aktuell for oss. Sixnet Series RAM 6000 var et godt alternativ, men er en ruter i høy prisklasse grunnet et utvalg av tilleggsfunksjoner vi ikke har behov for i prosjektoppgaven. Ruteren SIMATIC CP 1243-1 er utviklet for TeleControl av SIMATIC-PLSer, som ikke er relevant for oppgaven.

4.2. VMware *bridging* av nettverkskort

Fordi alt i VMware er virtuelt, må vi lage en kobling mellom det fysiske nettverkskortet og den virtuelle maskinen.

1. Åpne VMware player som administrator og kjør Win7 TIA-*image.*
2. Gå inn i *Player -> Manage -> Virtual Machine Settings.*
3. Velg *Network Adapter*. Under *Network connection* velg *bridged* og trykk *Configure Adapters.*
4. Sørg for at det nettverkskortet du ønsker å bruke og koble PLS med er krysset av i *Automatic Bridging Settings.*

OBS. Dersom *Automatic Bridging Settings* ikke viser tilkoblede enheter kan en løsning være å laste ned en tidligere versjon av VMware Player.

4.3. PLS-oppsett

1. Åpne TIA Portal, velg *New Project* eller åpne et ønsket eksisterende prosjekt.
2. Velg *Add New Device*.
3. Under SIMATIC S7-1200 velg *Unspecified CPU 1200*.
4. I *Hardware Detection for PLC*, velg nettverkskortet som er tilkoblet PLSen og trykk *Search*.
5. PLSen skal dukke opp i menyen under. Trykk på den og velg *Detect*.

4.4. Ruter-oppsett og kommunikasjon

For ruteren SCALANCE M874-2, vil oppsettet se slik ut:
1. Koble ruteren til PCen via ethernet.
2. IP-adressen til ruteren finner man ved å nullstille ruteren til *factory settings*. Da vil default-adressen være 192.168.1.1 og subnettmasken 255.255.255.0. (ifølge Siemens sitt oppsettdokument til ruterserien SCALANCE M-800 *"Getting started"*)
3. Gå inn på nettleseren og skriv inn IP-adressen til ruteren. Man vil få opp et log-in-vindu.
4. Når ruteren er nullstilt, så er passord og brukernavn satt til "admin". Fyll ut i de tomme feltene og logg inn.
5. Nå får man tilgang til set up-interfacet, der man kan konfigurere ruterens innstillinger. Første gang man logger inn etter nullstilling, vil man kun få tilgang til *basic wizard* som er det nødvendige man må gjennom for ruteroppsettet. Den består av åtte faner.
6. I første fanen *IP* fyller man ut IP-adressen og subnettmasken som man vil tildele ruteren. Disse kan være det samme som før. Vi fylte inn IP-address: 10.0.0.100, Subnet Mask: 255.255.255.0. Her er det lurt å være bevisst på at adressen man velger har samme nettverk som PLSen man ønsker å bruke. Trykk så *next*.
7. I andre fanen *Device* skal man fylle ut de tre tingene som bestemmer IDen til ruteren. System Name: bachelorGSM, System Location: step, System Contact: service@m800.com. Hva man skriver inn her er kun for å kjenne igjen ruteren. Adressen vi bruker under System Contact er oppført som et eksempel i Siemens sitt oppstartsdokument.
8. I tredje fanen *SIM* fyller vi ut Radio Mode: GSM only og Authentication Method: Auto. PIN-kode behøves ikke med SIM-kortet vi bruker. Kryss også av Enable Mobile Network Interface og Allow Data Roaming (om ønskelig).
9. I fjerde fanen *Operator* settes konfigurasjon for nettverksoperatøren vi bruker. Hvilke innstillinger man legger til her avhenger av operatøren man bruker og finnes for eksempel på hjemmesidene deres. Parameterne våre er som følger: PLMNID: 24209, Operator Name: com4, APN: com4. User Name og Password skal være blankt for operatøren vi bruker. Trykk *create* for å legge til parameterne til lista og huk av *enable*.
10. I femte fanen *Time* velger man innstillinger for ruterens interne klokkeslett og dato. Man kan sette opp tiden manuelt, synkronisere ruteren med PCs satte tid eller bruke en NTP-klient. NTP-klienten står for Network Time Protocol og er en mulighet til å motta informasjon om tid fra en slik server. Vår tidssone er +01.00. Vi krysset av NTP Client og valgte: NTP Server Index: 1, NTP Server Address: 192.53.103.108, NTP Server Port: 123, Poll Interval: 64 (hvor ofte den skal oppdatere seg etter serveren).

Det er verdt å nevne at siden vi bruker en GSM-ruter er databruken viktig å reflektere over. Et pollintervall på 64 henter informasjon hver 64. sekund, som er relativt ofte, men denne dataen er svært liten slik at det ikke vil utgjøre en betydelig forskjell.

11. Dersom man ønsker at ruteren skal nås via et nettdomene, kan man fylle ut DDNS-informasjonen i den sjette fanen "DDNS". Vi bruker ikke dette og lar rutene stå tomme.
12. Den syvende fanen er *SINEMA RC*. SINEMA RC er en server-applikasjon bestående for konfigurering og vedlikehold av fjerntliggende rutere og nettverk. Applikasjonen er laget for større nettverksoppsett i industrien, og er ikke noe vi ønsker å benytte oss av.
13. Siste fanen er oppsummering for kontrollering av informasjon man har fylt ut. Trykk *Set Values* for å anvende konfigurasjonene.
14. Når vi er ferdig med *Basic Wizard*, får vi tilgang til en større konfigureringsmeny i margen til venstre. Under *Security -> Passwords*, har man muligheten til å bytte passord. Vi endret vårt til St3p@step.

Figur 6: Ruterens meny etter at Basic Wizard er gjennomført

15. For å få tilgang til internett må vi endre noen innstillinger for at kommunikasjonen kan passere gjennom brannmuren. Under *Security -> Firewall*, og under fanen *IP Rules* kan man legge til regler/unntak for godkjennelse ved å trykke *create*. Vi har lagt til:
Protocol: IPv4, Action: Accept, From: vlan1, To: ppp0, Source (Range): 192.168.1.0/24, Destination (Range): 0.0.0.0/0, Service: all, Log: info, Precedence: 0
Protocol: IPv4, Action: Accept, From: ppp0, To: vlan1, Source (Range): 0.0.0.0/0, Destination (Range): 192.168.1.0/24, Service: all, Log: info, Precedence: 1

Nå er ruteren konfigurert, kan motta data fra PLS og har internettilkobling. Ulempen er at PLSen S7-1200 ikke kan kryptere dataen den sender, og for sikre meldingene bør vi gjøre noen tiltak.

4.5. Oppsett av Visual Studio med TIA Openness

1. Krav for bruk av TIA Openness

- WinCC og/eller STEP 7 er installert.
- TIA Openness er installert.

For å kunne programmere TIA Openness applikasjoner må kravene under tilfredsstilles:

- Windows Visual Studio må være installert.

2. Krav for innstallering av TIA Openness

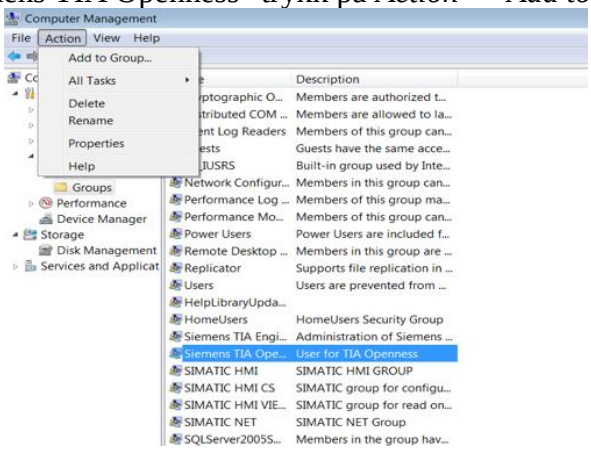
- Datamaskinen som benyttes tilfredsstiller systemkravene.
- Brukeren har administrative rettigheter.
- Eventuelle programmer som kjører blir lukket.
- Autorun blir stoppet.
- WinCC og/eller STEP 7 er installert.
- Versjonen av TIA Openness må stemme overens med versjonen av WinCC og/eller STEP7.

For å kunne installere TIA Openness må TIA Portal installeres. Ved innstallering av TIA Portal må boksen *TIA Openness* krysses av. Dette vil installere TIA Openness på datamaskinen, og vil samtidig gjøre den lokale gruppen *Siemens TIA Openness* tilgjengelig.

3. Legge til brukere til lokale gruppen «Siemens TIA Openness»

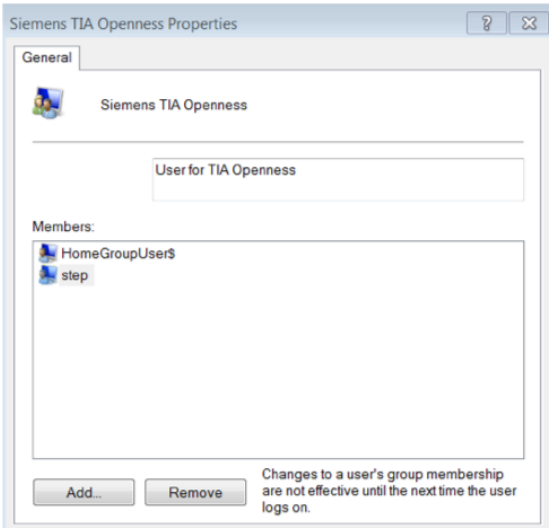
Hver gang en bruker benytter TIA Openness applikasjonen sin, for å få kontakt med TIA Portal, vil TIA Portal sjekke om brukeren er et medlem av den lokale gruppen «Siemens TIA Openness». Hvis brukeren er medlem av gruppen vil de kunne få kontakt med TIA Portal ved bruk av TIA Openness applikasjonen sin.

1. Gå til søkemotoren  , og søk "Kontrollpanel".
2. Trykk på "Administrative verktøy".
3. Trykk deretter på "Datamaskinbehandling" -> "Local Groups and Users" -> "Groups" og velg "Siemens TIA Openness"
4. Etter å ha valgt "Siemens TIA Openness" trykk på *Action -> Add to Group...*



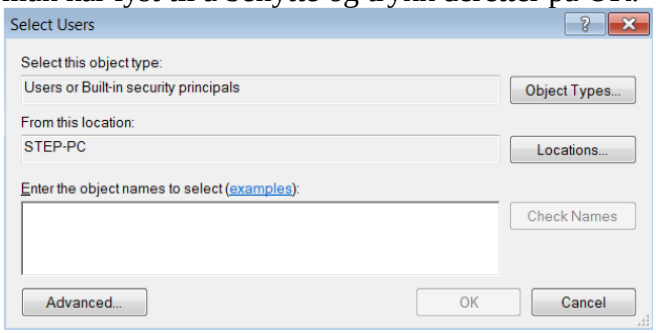
Figur 7: Hvor på datamaskinen lokale gruppen "Siemens TIA Openness" befinner seg.

5. Velg hvilke av medlemmene man har lyst til å bruke og trykk deretter på *Add...*



Figur 8: Vinduet for å velge hvilken medlem man skal inkludere I gruppen "Siemens TIA Openness".

6. Skriv inn brukeren man har lyst til å benytte og trykk deretter på *OK*.



Figur 9: Vinduet som gir brukeren muligheten til å legge til ønsket medlem.

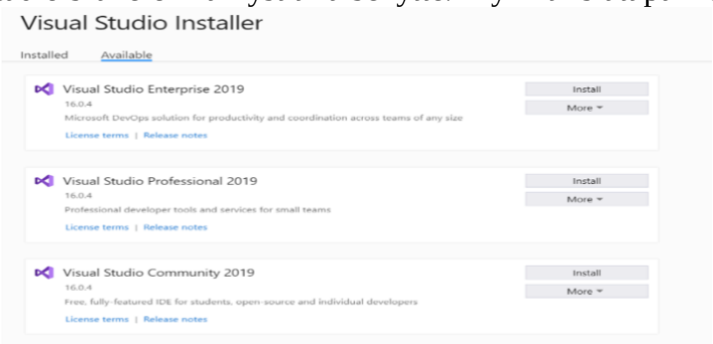
7. Trykk på *OK* for å avslutte.
8. For å bekrefte valgte som er tatt må brukeren logge ut av datamaskinen og logge inn igjen.

4. Installering av Microsoft Visual Studio

1. Last ned *Microsoft Visual Studio 201x* fra hjemmesiden til Microsoft. For denne oppgaven ble Microsoft Visual Studio 2019 brukt.

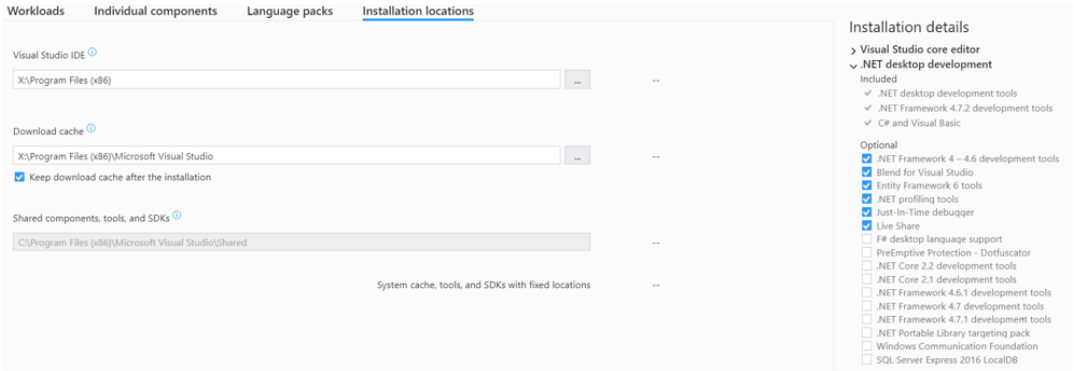
2. Etter at nedlastningen er fullført gå til søkemotoren på datamaskinen  og søk etter *Visual Studio Installer*.

3. Når *Visual Studio Installer* har dukket opp på skjermen velg *Available* og velg hvilken variant av Visual Studio brukeren har lyst til å benytte. Trykk til slutt på *Install*.



Figur 10: De ulike variantene av Visual Studio som kan velges imellom.

4. Når installasjonen er fullført vil en ny skjerm dukke opp, trykk på *Workload*. Brukeren vil nå få ulike kategorier å velge imellom. Finn frem til kategorien *Windows* og velg *.NET desktop development***.**
5. Etter å ha valgt **.***NET desktop development* trykk på *Installation location* for å velge hvor på datamaskinen produktet skal lastes ned. Etter å ha valgt hvor på datamaskinen Visual Studio skal lastes ned, må brukeren på høyre side krysse av *.NET Framework 4-4.6 development tools*.

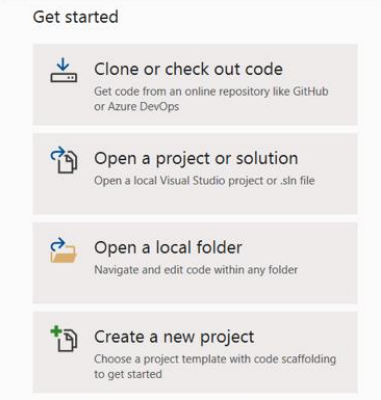


Figur 11: Valg av et egenbestemt lokalisasjon på datamaskinen ved nedlastning av Visual Studio og valg av å inkludere .NET Framework 4-4.6.

6. Trykk til slutt på «Installer» for å starte installasjonen av *Microsoft Visual Studio*.
7. Etter at installasjonen er fullført kan brukeren gå til søkemonitoren, søke opp *Visual Studio* og trykke på den for å starte programmet.

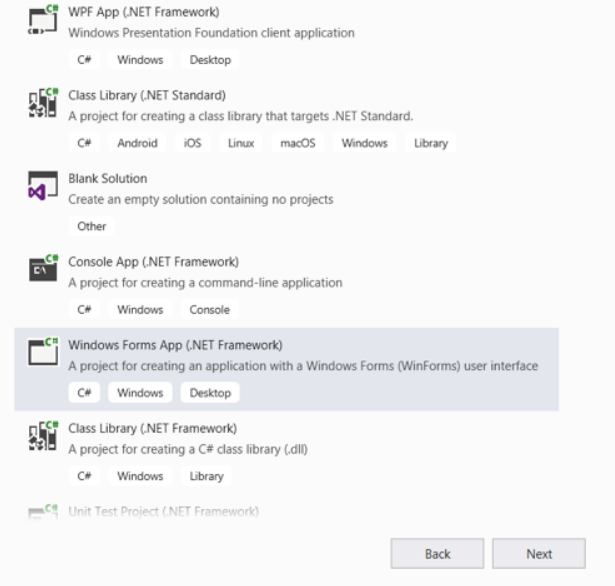
5. Legge til .dll i Visual Studio

DLL (*eng: Dynamic Link Library)* er en fil som benyttes til å kunne få tilgang til visse forhåndsbestemte funksjoner fra en annen programvare. For denne oppgaven ble *Siemens.Engineering.dll* benyttet til å kunne få tilgang til TIA Portal sine funksjoner gjennom Visual Studio.
1. For å legge til *Siemens.Engieering.dll* i Visual studio må brukeren først starte programmet Visual Studio og lage et nytt prosjekt. Trykk på *Create a new project* for å starte.



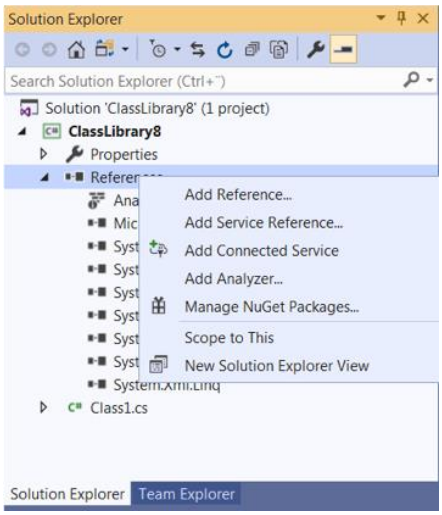
Figur 12: Ulike valgene ved starten av et nytt Visual Studio prosjekt.

2. Skriv deretter inn hva prosjektet skal hete under *Project name* og hvor på dataen prosjektet skal lagres under *Location*. Trykk deretter på *Create* for å gå videre.
3. Siste steget for dannelsen av prosjektet vil brukeren få ulike versjoner av .NET å velge imellom. Her må *Class Library (.NET Framework)* velges og deretter trykk *Next* for å bekrefte valget



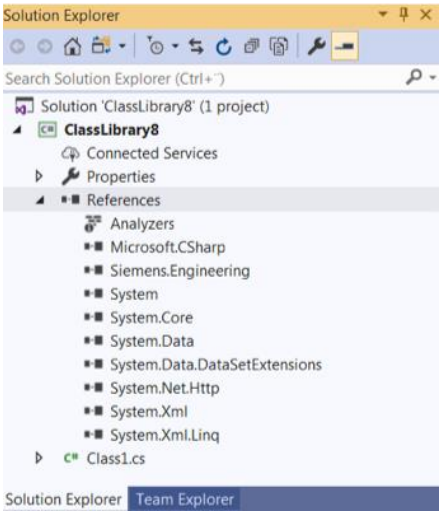
Figur 13: Forskjellige variantene av scripts man kan velge imellom ved programmering i Visual Studio.

4. Når prosjektet er dannet vil det på høyre siden være et felt som heter *Solution Explorer*. Her må brukeren for å legge til *Siemens.Engineering.dll* høyre trykk på kategorien *References* og deretter trykke på *Add Reference*.



Figur 14: Feltet "Solution Explorer" som gjør det mulig å legge til "Siemens.Engieering.dll".

5. For å finne frem til *Siemens.Engineering.dll* må brukeren trykke på *Browse* og deretter gå til mappene; *Siemens → Automation → Portal V15 → PublicAPI → V15.*
6. I mappen *V15* vil det ligge to .dll'er: *Siemens.Engieering.dll og Siemens.Engieering.HMI.dll.* Brukeren må velge *Siemens.Engieering.dll* og deretter trykke på *Add* for å legge til .dll i Visual Studio programmet.
7. Siemens.Engieering.dll vil nå ligge under feltet *Solution Explorer* og kan nå benyttes i *Visual Studio*.

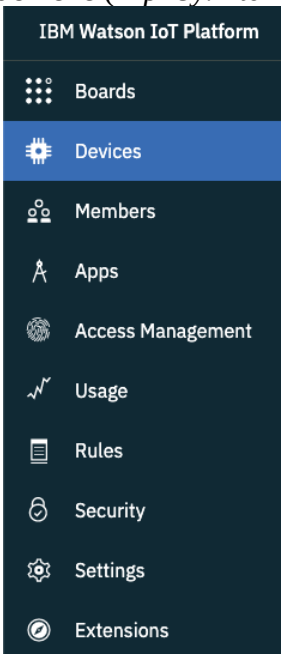


Figur 15: "Siemens.Engineering.dll" lagt til i prosjektet.

4.6. IBM-oppsett og kommunikasjon

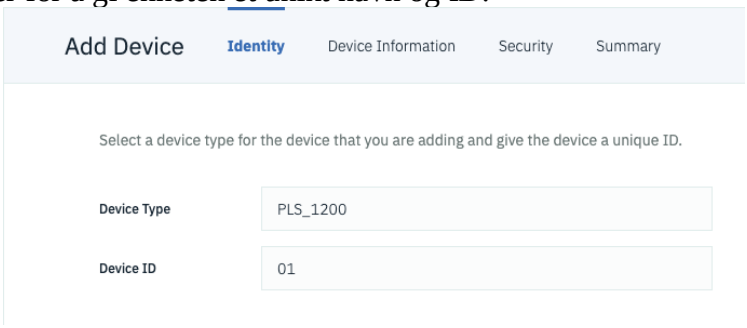
IBM Cloud er en fellesbetegnelse for skylagringstilbudet IBM tilbyr, og er et produkt som inneholder ulike underprodukter designet til spesifikke bruksområder Vi ønsker å koble til en enhet (PLS) til skytjenesten, og velger produktet Watson IoT Platform for dette. Denne plattformen er designet for å koble til IoT-enheter, og PLSen kan brukes i denne sammenhengen.
Plattformen støtter også kommunikasjon over MQTT i form av at den har samme egenskaper som en broker slik at klienten kan koble seg rett på skyen. På denne måten kan man sende beskjeder direkte fra enheten (PLSen) til skyen uten å gå via en fjerntliggende broker.

Når man har laget gratisbruker hos IBM Cloud kan man bruke denne brukeren til å registrere seg i Watson IoT Platform. Når dette er gjort det genereres en organisasjons ID er knyttet til brukeren din som gir tilgang til å bruke plattformen. Vi ble tildelt organisasjons-IDen *1zp28y* og bruker denne for å logge på vårt domene (*1zp28y.internetofthings.ibmcloud.com*).



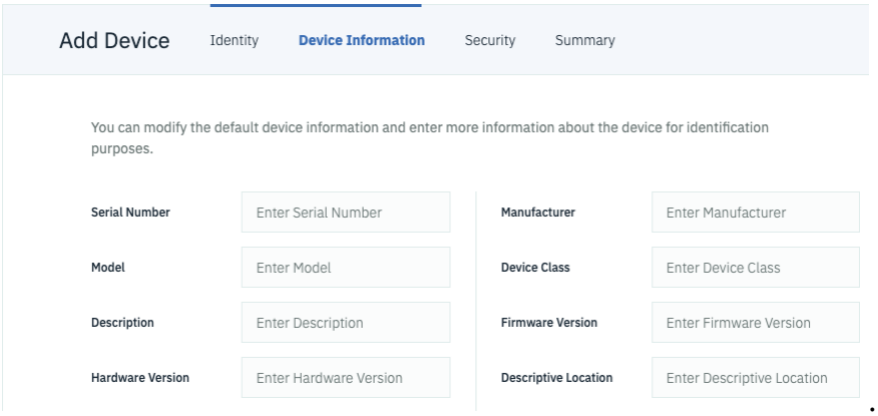
Figur 16: Menyen i Watson IoT Platform som gir tilgang til plattformens ulike funksjoner

For å oppnå kommunikasjon mellom Watson IoT Platform og PLSen, må en enhet settes opp på skyen. Denne enheten blir en virtuell representasjon av enheten vi skal kommunisere med, og settes opp under Devices.
1. For å legge til en ny enhet, velg *Add Device.*
2. Deretter må man definere et navn for enheten i *Device Type* og deretter en ID for enheten i *Device ID*. Dette er for å gi enheten et unikt navn og ID.



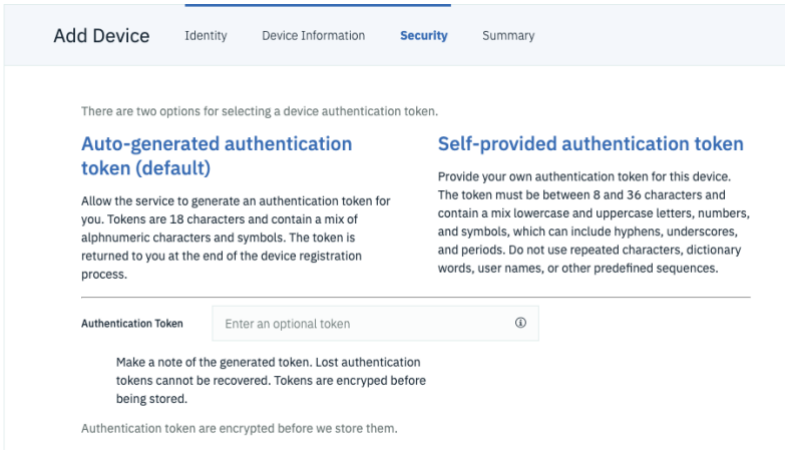
Figur 17: Deklarer enhetstype som skal brukes og ID'en til enheten.

3. Velg *Next.*
4. Deretter har man muligheten til å sette tilleggsinformasjon for enheten. Da vi kun har en enhet tilkoblet, velger vi å ikke fylle ut noen av feltene



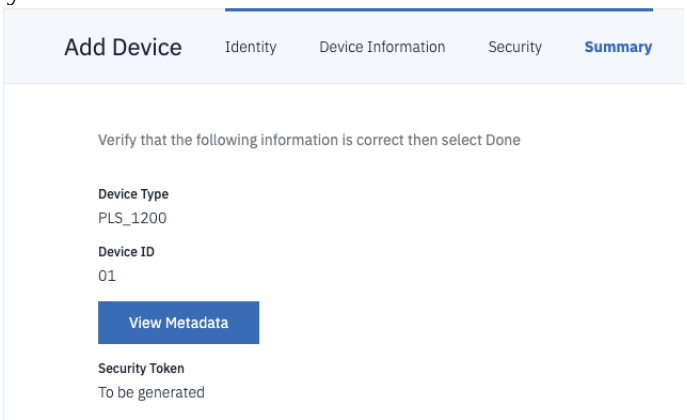
Figur 18: Mulige tilleggsinformasjon som kan inkluderes.

5. Velg *Next.*
6. Under **Security** har man valget mellom å skrive inn enn egen *token* som skal brukes for tilkoblingen, eller få den auto-generert. Et *token* er en unik ID bestående av tilfeldige karakterer. Denne skal fylles inn som passord på enheten man skal kommuniserer med som ekstra sikkerhet.



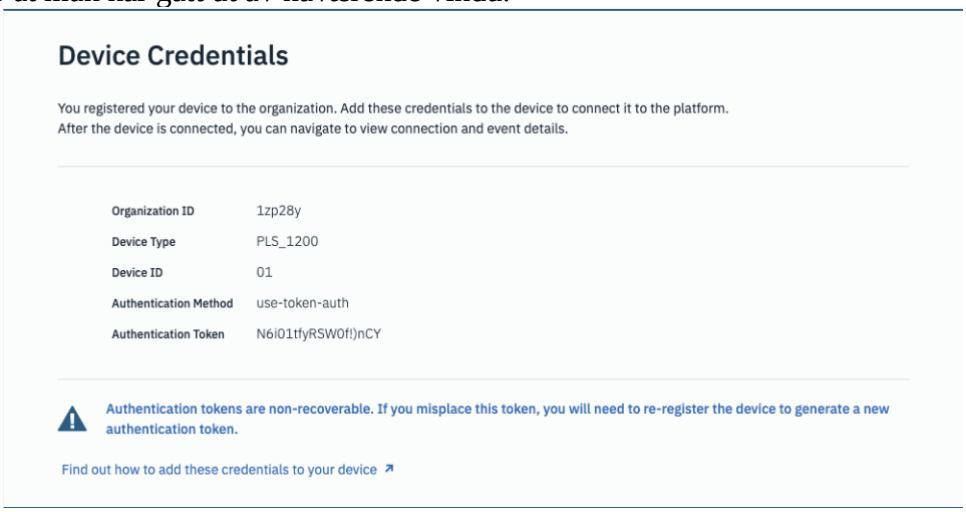
Figur 19: Vi valgte å beholde feltet tomt, slik at plattformen automatisk genererer et token for oss.

7. Velg *Next.*
8. I den siste fanen vil man se et sammendrag av valgene man har gjort. Dersom man er fornøyd med dette, trykker man *Done.*



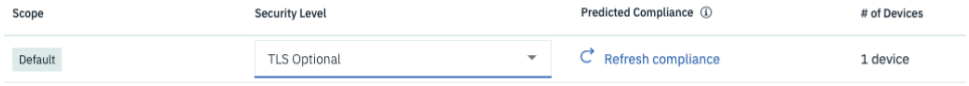
Figur 20: Oppsummering av opplysningene som ble registrert.

9. Når enheten er ferdig konfigurert, vises en liste av parameterne som må defineres på enheten. Det er viktig å notere ned den genererte *tokenen*, da denne ikke er mulig å se den igjen etter at man har gått ut av nåværende vindu.



Figur 21: :rametrene som tas I bruk med enheten.

10. Siden PLSen vi bruker ikke har mulighet til å kryptere data, må IoT-plattformen konfigureres til å også godkjenne ukryptert data. Dette gjøres under *Security* og deretter endre *Security Level* under *Default Rule*. Deretter settes *Security Level* til *TLS Optional.*

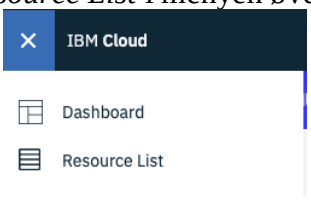


Figur 22: Konfigurerer IoT-plattformen til å kunne godkjenne ukryptert data.

11. Dersom man ønsker å visualisere måledataene man laster opp, kan dette gjøres ved å lage et nytt *board* i **Boards** i menyen. På grunn av at måledata mottas i meldingsformatet JSON, gir det muligheten til å koble datanavn i for eksempel en graf slik at verdien til denne dataen blir visualisert i sanntid.

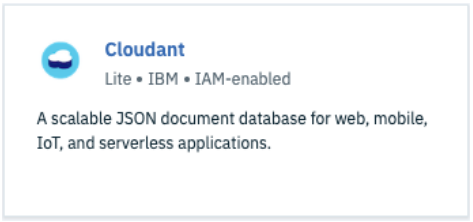
Da er nødvendig oppsett for å kunne motta meldinger i Watson IoT Platform gjort, og det gjenstår å sette opp en database for lagring av den opplastede dataen.

12. Implementering av databasen *IBM Cloudant* gjøres ved å logge på *ibmcloud.com* med IBM IDen. Deretter går man på *Resource List* i menyen øverst til venstre.



Figur 23: Implementerer databasen IBM Cloudant

13. Deretter velges *Create Resource*, og lokaliserer *Cloudant* i listen blant tilgjengelige produkter.



Figur 24: Tar I bruk Cloudant

14. Ved å trykke på *Cloudant*-boksen får man mulighet til å velge parametere for oppsett av databasen. Sørg for at *Choose a region/location to deploy in* er satt til en europeisk server og velg deretter *Use only IAM* under *Available authentication methods.* Trykk deretter på *Create***.** En database er nå laget og ved å velge den i *Resource list* har man muligheten til å logge på databasens grensesnitt for monitorering, modifisering eller sletting av lagrede meldinger.

15. For at databasen skal lagre meldingene som ankommer Watson IoT Platform, må dette konfigureres i plattformen under *Extensions* ved å velge *IBM Cloudant* som *Historian Data Storage.* Når dette er gjort vil alle meldingene som mottas i Watson IoT Platform lagres direkte i databasen.

4.7. Sikkerhet og IPsec

Sikkerhet blir stadig mer viktig og prioritert i alle industrifelt. Det er svært mye verdifull og sensitiv data som blir lastet opp og videreført frem og tilbake. Blant annet på økonomiske og livsviktige grunnlag må vi gjøre tiltak som sikrer at dataen ikke kan misbrukes av uredelige grupper og individer.

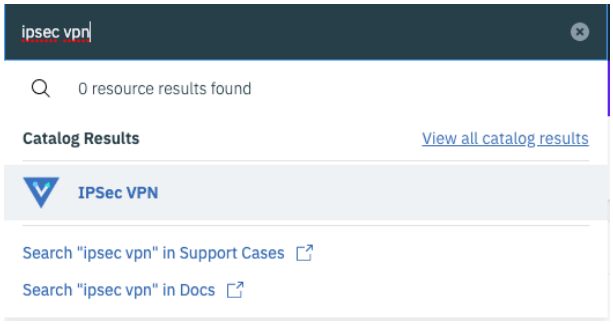
En del av oppgaven vår er å vurdere mulige sikkerhetsløsninger og velge en som er anerkjent sikker og/eller er den rimeligste av disse. Sikring av data kan løses på ulike måter, og kommer ofte i form av å skjule, kryptere, blokkere, eller en kombinasjon av disse.
En av løsningene er å sette opp en VPN-tunnel. I vårt tilfelle kan vi benytte oss av IBM Cloud sin VPN-tjeneste, IPsec VPN. Denne er relativt enkel å sette opp gjennom IBM, og oppfyller kravet om å være trygg nok. Tjenesten koster per dags dato oppunder 900 NOK per måned å abonnere på. Det er typisk at skytjenester krever en viss sum for slike tjenester.
Det er også mulig å bruke Secure Gateway hos IBM. Det er en tjeneste som tillater oss å lage en *gateway* som krypterer dataen mellom ruteren og skytjenesten. Prisen for dette er en engangssum på omtrent 500 NOK per tilkoblede enhet. Denne løsningen er avhengig av at ruteren man bruker kan settes opp med et operativsystem (eks. Linux).

En alternativ løsning er å benytte en S7-1500 PLS istedenfor en S7-1200 som vi i utgangspunktet skal bruke. PLSen konfigureres helt likt, men kan derimot kryptere data. Byttet er altså enkelt og overføringen er sikker. I praksis kan det vise seg å være mer ideelt å ta i bruk en S7-1200 PLS da denne er vanligere i "felten". Selv om å bytte PLS er tilsynelatende den billigste og enkleste løsningen i dette prosjektet, blir ikke dette nødvendigvis billig i praksis dersom en kunde må bytte alle PLSene på et anlegg for å sikre overføringene. Det medfører antatt nedetid på anlegget, ekstra kostnader og gjenoppsett av systemet.

Beste løsning må vurderes ut ifra situasjonen og kunden. Alle er løsninger som vil koste bedriften noe dersom de ønsker å sikre sin data, om den ikke bruker S7-1500 PLSer fra før av. Vi konkluderte med at vi vil enten sette opp IPsec VPN eller bruke en S7-1500 PLS til prosjektet, men det avhenger av hvilken løsning prosjektleder eller ansvarlig favoriserer. En eventuell løsning for oss vil være å levere et prosjekt som er kompatibelt med begge løsningene, så kan bedriften gjøre vurderingen av hvilken av disse de vil bruke avhengig av oppdraget.

IBM Watson og IPsec

For å sette opp en *Site-to-site* VPN-tunnel mellom IBM Cloud og ruter, må dette konfigureres for begge endepunktene.
1. Det første som må gjøres er å legge til tjenesten IPsec VPN i IBM Cloud-kontoen på *ibmcloud.com.*
2. Søk etter “ipsec vpn” i søkefeltet øverst, og velg *IPsec VPN*

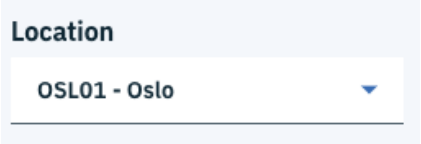


Figur 25: Tar I bruk IPsec VPN

3. På siden man ankommer, velg *Create*

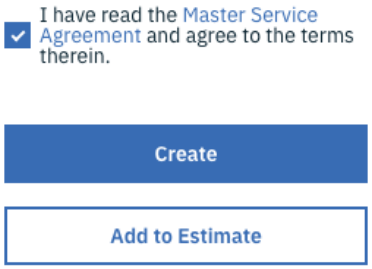
Figur 26:"Create" for å kunne gå videre for å opprette IPSec VPN.

4. Under *Location,* velg *OSLO1 – Oslo*



Figur 27: Lokalisasjonen til brukeren.

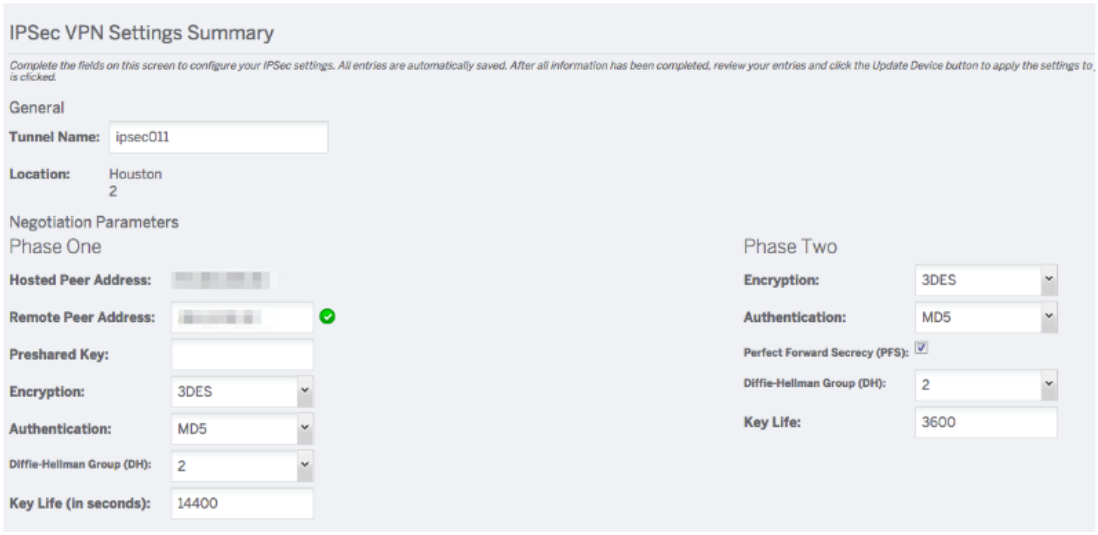
5. Når dette er gjort, les og deretter huk av *I have read the Master Service Agreement and agree to the terms therein.* Velg til slutt *Create* for å betale og opprette produktet.



Figur 28: Samtykker med kravene og oppretter produktet.

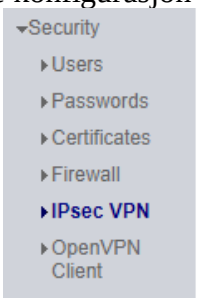
Når produktet er opprettet på IBM Cloud-kontoen er IPsec VPN klar til bruk og kan finnes i *Resource List.* Velg produktet i listen for å begynne konfigurasjon av ny VPN-tilkobling. Vi har ikke gått videre med betalingen og oppsettet deretter fordi vi ikke har de økonomiske ressursene til å fullføre betalingen for prosjektet. Videre konfigurering tyder på å være enkel, og viktig å gjennomføre for å ha et lag med sikkerhet, da vår forbindelse for øyeblikket ikke er kryptert.

Under oppsettet må en *Remote Peer Address* defineres. Her velges den offentlige IP-addressen til ruteren, som kan finnes ved å koble seg til ruteren og deretter gå på nettsiden *whatismyip.com.* Vår ruters adresse med operatøren *com4* er *185.67.16.155/24.* Krypteringsmetode må også settes, her er det lurt å være bevisst over hvor mange bit de ulike krypteringsnøklene bruker. For mest sikker kryptering, velg *AES256 (256 bit)*



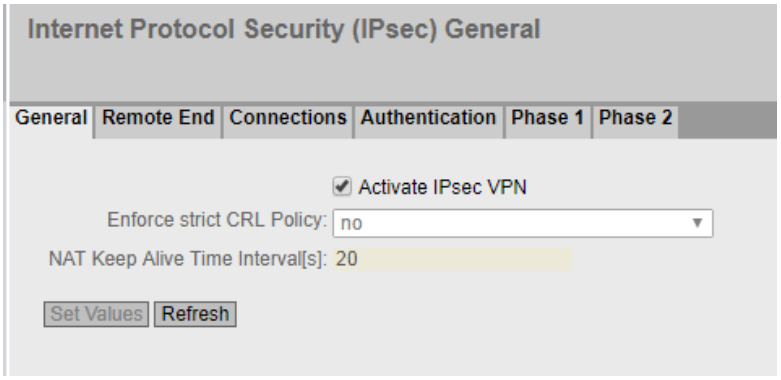
Figur 29: Forslag til konfigurasjon av VPN-tunnel med IPsec VPN i IBM Cloud. Hentet fra https://cloud.ibm.com/docs/infrastructure/iaas-vpn?topic=VPN-setup-ipsec-vpn

Videre må tilsvarende oppsett for IPsec konfigureres på ruteren. Ruteren vi bruker har støtte for IPsec VPN, dette settes opp ved å gå inn på ruteroppsettet via nettleseren.
1. Få tilgang til ruteren ved å skrive *10.0.0.100* i nettleseren. Logg deretter inn ved å benytte tidligere satt brukernavn og passord.
2. Gå til *Security > IPsec* for å begynne konfigurasjon av VPN-tunnel på ruteren



Figur 30: Alternativene som kan velges ved valg av sikkerhet.

3. Oppsett av VPN består av delene *General, Remote End, Connections, Authentication, Phase 1* og *Phase 2.* Start med å huke av *Activate IPsec VPN,* og sørg deretter for at alle stegene blir gjort ved å bla igjennom alle fanene. Oppsettet skal utføres tilsvarende konfigurasjonen som ble gjort for IPsec VPN på skytjenesten.
IP-addressen for *Remote End* skal settes lik IP-adressen til *Watson IoT Platform:*
IP Remote End: 159.8.169.212/24



Figur 31: Stegene som må fylles ut ved oppsett av VPN.

4.8. Testing av MQTT-publisher

På Siemens sine hjemmesider, under *Product Support* ligger det en MQTT Publisher for SIMATIC CPU (PLSen vi bruker). En MQTT Publisher er et program som lar en PLS sende meldinger med MQTT-protokollen. Programmet består hovedsakelig av en funksjonsblokk med all nødvendig kode for kommunikasjon over MQTT. Funksjonsblokka gir oss muligheten til å sette parametere for tilkobling til en *broker* og velge innstillinger for meldingen som skal sendes, for eksempel *topic* og *QoS* (Quality of Service).

Publisheren er et TIA Portal-prosjekt med en enkel funksjonsblokk, og to datablokker. Funksjonsblokka håndterer kommunikasjon med en broker via MQTT-protokollen, mens datablokkene lagrer alle nødvendige parametere som trengs for å oppnå kommunikasjon. Parameterne som settes er avhengig av *brokerens* addresse og meldingens metadata i form av *topic*, *QoS*, port etc.

For å teste denne publisheren etter nedlasting, og samtidig teste om PLSen klarer å kommunisere via GSM-ruteren prøvde vi å sende en enkeltmelding fra PLSen til en broker. I dette tilfellet brukte vi brokeren Mosquitto, som er et anerkjent alternativ.
OBS! Vi må legge til en port for videresending i ruteroppsettet for å få tilgang til PLSens webserver via ruterens adresse. Dette fylles ut under *Layer 3*, deretter *NAT* (Network Address Translation), og under fanen *NAPT* (Network Address Port Translation). Vi legger til en tilgang til PLSen via port 80 ved å fylle ut Source Interface: ppp0, Traffic Type: TCP. Huk av *Use Interface IP from Source Interface*, slik at Destination IP Address blir automatisk fylt ut (den addressen ruteren har fått fra nettverksoperatøren). Deretter fyll ut: Destination Port: 80, Translated Destination IP Address: 10.0.0.10 (PLSen sin IP-adresse), Translated Destination Port: 80.

4.9. Programmering av PLS

Programmet håndterer sampling av måledata basert på filtreringsalgoritme og kjører på en SIMATIC S7-1200 PLS. Det består av ulike moduler som til sammen utgjør en prosess som måler, analyserer, sender og eventuelt bufrer data basert på parametere brukeren har valgt i filtrene. Modulene består av:

Tabell 10
Modulene PLS-programmet inneholder

Funksjonsblokker	
MQTT_Publisher	Håndterer kommunikasjon med broker.
Converter	En konverter som normaliserer og skalerer, og deretter lagrer konvertert data.
Filtre	Ulike filtre bestående av algoritmer for å analysere dataen og gradere viktigheten til hver enkel måling.
Sampler	Trigger sampling basert på en enkel *timer*. Tiden mellom hver sampling er avhengig av en gitt samplingsfrekvens.
"ConstructMsg"	Konstruerer en melding i henhold til JSON-format. Meldingen som blir konstruert inkluderer måledata som er gradert som viktig, høyeste grad av viktighet meldingen inneholder (hvor kritisk er den mest kritiske dataen) og et tidsstempel for når målingene ble gjort.
"BufferHandler"	Håndterer om meldingen skal sendes nå eller lagres i buffer avhengig av kommunikasjonsblokkens (MQTT-publisher) status, i tillegg til å tømme bufferen dersom den inneholder meldinger som kan sendes.
Datablokker	
LMqtt_Data	Inneholder parametere som beskriver brokerens adresse, samt metadata til MQTT-meldingen.
LMqtt_Publish	Inneholder variabler som beskriver nåværende status til MQTT_Publisher
DataStorage	Lagring av konvertert måledata med navn, verdi og kritisk status. Inneholder også en buffer der meldinger som ikke ble sendt mellomlagres, i tillegg til informasjon som samplingsrate og samplingsstatus.

MQTT_Publisher
Denne blokken tar hånd om alt som er nødvendig for kommunikasjon med en MQTT-broker. Alle kommunikasjonsprotokoller krever at dataen sendes på en spesifikk måte, og MQTT skilles seg ikke ut med sine egne retningslinjer for dette. For at brokeren skal forstå meldingen som blir sendt i en datapakke, må den bestå av et pakkehode som inneholder all metadata for kommunikasjonen, i en bestemt rekkefølge.

For å oppnå kommunikasjonen med ønsket broker, må parameterne til funksjonsblokken defineres. Først må *hwIdentifier* og *connectionID* fylles ut. Hva som står her er opptil brukeren da dette hjelper med å identifisere denne forbindelsen og hvilken PLS som kommuniserer dersom man har et nettverk med flere PLSer.

Ved *IpAdressBroker* skriver man inn IP-adressen til brokeren man ønsker å sende meldinger til. PLSen vår skal sende data til vår bruker på Watson IoT Platform, som også fungerer som en broker. Etter å ha laget en bruker hos Watson IoT Platform, får man egen *URL* for tilkobling av enheter; *org_id.messaging.internetofthings.ibmcloud.com*. IP-adressen til denne URLen kan man finne ved å for eksempel pinge den i command-vinduet.

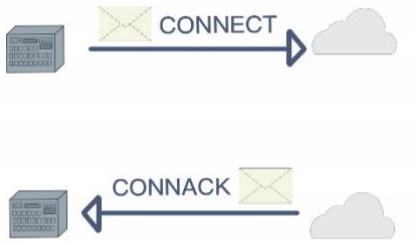
```
C:\Users\pepetter>ping 1zp28y.messaging.internetofthings.ibmcloud.com

Pinging uk.messaging.internetofthings.ibmcloud.com [159.8.169.212] with 32 bytes
 of data:
Reply from 159.8.169.212: bytes=32 time=23ms TTL=53
Reply from 159.8.169.212: bytes=32 time=23ms TTL=53
Reply from 159.8.169.212: bytes=32 time=24ms TTL=53
Reply from 159.8.169.212: bytes=32 time=23ms TTL=53
```

Figur 32: Vi pinger URLen for å finne IP-adressen.

På *localPort* fyller man ut hvilken port PLSen skal bruke for å opprette kommunikasjon over TCP. Her kan man velge port 2000, som er en standard for TCP-forbindelser med PLS. For MQTT-meldinger som skal sendes, velger man den aktuelle porten på *mqttPort*. Dette feltet er avhengig av hva slags mottaker man skal sende pakker til, og om innholdet skal krypteres eller ikke. Dersom man skal sende til en broker, brukes port 1883 for ukryptert data og port 8883 for kryptert data. Hvis man skal overføre til en web-server, skal det gjøres over port 8080 for ukrypterte pakker og port 8081 for krypterte. Meldingene våre må sendes ukryptert dersom S7-1200 PLS ikke støtter kryptering.

MQTT_Publisher har også mulighet til å ta i bruk ytterlige funksjoner som medfølger MQTT-protokollen. Hvis man ønsker å ta de i bruk, kan de aktiveres. En av disse funksjonene er QoS (Quality of Service). Denne kan velges å være av grad én eller av grad to, og regulerer hvor stor grad av bekreftelse klienten krever fra mottaker for å sikre at meldingen ble mottatt. Om QoS aktiveres, vil kommunikasjonsblokken gjøre ytterlige utvekslinger i form av *handshakes* med brokeren for å bekrefte at meldingen som er sendt ble mottatt. Det fører til større bruk av båndbredde, noe som bør tas i betraktning i henhold til kravene man må overholde. Vi velger å sette QoS til nivå én, som går ut på at blokken venter på en bekreftelse fra broker om meldingen er mottatt. På denne måten har vi bedre kontroll på at data ikke går tapt underveis. Det er en nyttig funksjon ved overføring over mobilnett, siden variasjoner i signalstyrke kan oppstå. Samtidig krever første nivå mindre båndbredde enn nivå to.



Figur 33: Handshake mellom klient og server som bekrefter at meldingen er mottatt

MQTT-blokken har en annen funksjon som heter *Will*. Dette er klientens "testament", i betydning av at en melding sendes ut dersom kommunikasjonen er i ferd med å gå tapt. *Clean Session* er en funksjon som går ut på at brokeren får beskjed om å ta vare på meldingene som blir mottatt. Funksjonen er laget for at en vanlig broker-oppsett med klienter som abonnerer på data som brokeren mottar, og er ikke relevant om man skal sende data direkte til skytjenesten.

Om man ønsker å bruke et brukernavn og et passord må det også aktiveres. Dette anbefales å tas i bruk og hjelper med å sikre tilkoblingen fra uvedkommende. Det er derimot ikke nok grunnlag for at kommunikasjonen er sikker. De fleste skytjenester krever at man tar i bruk denne funksjonen. For Watson IoT Platform skal organisasjons-ID settes som brukernavn og

et *token* (et passord bestående a tilfeldig genererte karakterer) som genereres på plattformen skal brukes som passord.

Keep Alive er en funksjon som gjør at klienten regelmessig sender en melding som holder kommunikasjonen åpen. Maksverdi mellom hver *keep alive*-melding er 18 timer, 12 minutter og 15 sekunder (65535s). Settes verdien til 0, vil ikke denne funksjonen tas i bruk. *packetIdentifier* sier noe om hvilken ID første melding har. IDen øker automatisk med 1 for hver ny melding.

Tabell 11
Parameterne vi satte for MQTT-funksjonene:

cleanSession	ja
will	nei
willQoS_1	nei
willQoS_2	nei
willRetain	nei
password	ja
username	ja
qualityOfService	1
retain	nei
keepAlive	3600
packetIdentifier	0

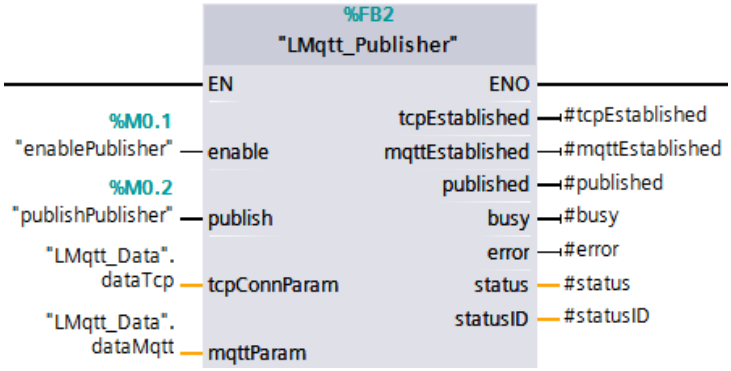
Videre må metadata til meldingen defineres. Her legges til den nødvendige informasjonen for at meldingen skal kunne nås av mottakeren. MQTT fungerer på den måten at hver melding inneholder egen ID og emne, og dette må samsvare med parameterne hos mottaker. Ved bruk av Watson IoT Platform som adressat, kreves det at disse parameterne settes opp i et bestemt format basert på valgene man gjorde under enhetens oppsett i plattformen.

Tabell 12
Parametere for MQTT-metadata.

clientIdentifier	d:1zp28y:PLC_1200:01
willTopic	ingen data
willMessage	ingen data
username	use-token-auth
password	N6i01tfyRSW0f!)nCY
topic	iot-2/evt/PLCdata/fmt/json
message	ingen data (fylles inn av *constructMsg*)

Når dette er gjort har man satt grunnlaget for kommunikasjon mellom PLS og skytjenesten. Videre må meldingstransporten automatiseres av et program som styrer kommunikasjonsblokken i from av hvilken tid meldinger skal publiseres og hva meldingene skal bestå av.

Styring av blokken muliggjøres ved manipulasjon av blokkens inn- og utganger. Den har fire innganger der to av disse er boolske variabler, mens de to mottar parametere for selve forbindelsen med mottaker og meldingenes metadata. Inngangene *enable* og *publish* gir brukeren kontroll over når klienten skal etablere forbindelse og når meldingen er ønsket sendt. Kommunikasjonsblokkens utganger beskriver kommunikasjonsstatus og gir oversikt over blokkens tilstand. Disse er viktige parametere for å implementere blokken i et større PLS-program og for å feilsøke eventuelle forekommende feilmeldinger (*error, status, statusID*). *tcpEstablished* og *mqttEstablished* går høy når forbindelsen med broker er oppnådd, og *published* går høy når blokken mottar en ACK (bekreftelse) fra brokeren om at meldingen er mottatt (gjelder for QoS nivå 1 og 2). Desom blokken er opptatt med å kommunisere, vil *busy* gå høy. Det er en indikasjon på at blokken er i bruk og ingen ytterlige meldinger kan sendes før *busy* går lav igjen.



Figur 34: Bilde av MQTT-blokk fra vårt program

Prosjektets krav innebærer at PLSen skal kommunisere med en skytjeneste og fortløpende laste opp meldinger med måledata. Hvorvidt data skal lastes opp er avhengig av statusen til de ulike målingene, som analyseres og beregnes av algoritmer i filtrene. Det er gjennomførbart ved å utvikle et program som håndterer disse operasjonene sammen med LMqtt_Publisher.

DataStorage

For å behandle måledataene må disse mellomlagres i en datablokk. Denne datablokken inneholder en *struct* (en liste der hver indeks inneholder en gruppe av tre elementer), som består av målingens verdi (*dataValue*), målingens navn som er satt av brukeren (*dataName*) og status på data markert av filtrene (*dataMarked*). Blokken lagrer også informasjon om nåværende samplingsfrekvens og samplingsstatus, slik at de andre funksjonsblokkene i programmet kan følge denne informasjonen og handle deretter. En annen viktig del av datablokken *dataStorage* er en buffer som inneholder alle ferdigproduserte meldinger som ikke kunne bli sendt.

payloadBuffer	Array[0..200] of String[200]	
payloadBuffer[0]	String[200]	''
payloadBuffer[1]	String[200]	''
payloadBuffer[2]	String[200]	''
payloadBuffer[3]	String[200]	''
payloadBuffer[4]	String[200]	''
payloadBuffer[5]	String[200]	''
payloadBuffer[6]	String[200]	''
payloadBuffer[7]	String[200]	''
payloadBuffer[8]	String[200]	''
payloadBuffer[9]	String[200]	''
payloadBuffer[10]	String[200]	''
payloadBuffer[11]	String[200]	''
payloadBuffer[12]	String[200]	''
payloadBuffer[13]	String[200]	''
payloadBuffer[14]	String[200]	''
payloadBuffer[15]	String[200]	''

DATA	Array[0..50] of Struct	
DATA[0]	Struct	
dataValue	Real	0.0
dataName	String	'Distance'
dataMarked	Int	0
DATA[1]	Struct	
dataValue	Real	0.0
dataName	String	'Temperature'
dataMarked	Int	0
DATA[2]	Struct	
dataValue	Real	0.0
dataName	String	'Pressure'
dataMarked	Int	0

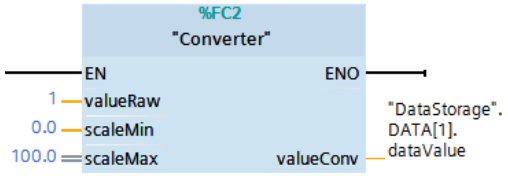
Figur 35: Viser inneholdet i datablokken dataStorage

Konverter

Hensikten med programmer er å analysere målesignaler som er tilkoblet PLSen og eventuelt sende disse videre for monitorering eller behandling. Det første steget er å konvertere målingene PLSen leser av til riktig datatype og verdi med spesifiserte avgrensninger. Konverteringsblokken består av tre innganger og en utgang. Inngangene er der brukeren kobler inn signalet man har i hensikt å konvertere, i tillegg til ønskede maks- og minimumsverdier for signalets skalering. Utgangen består av det ferdig konverterte målesignalet i *real*-format (flyttall).

PLSens analoge innganger leses av i datatypen *WORD*, som er en 16-bit hexadesimal-representasjon av en verdi. Denne datatypen settes på inngangen til converter-blokken, der den blir omgjort til en *int* (integer) og lagret i en midlertidig variabel. Denne variabelen blir så normalisert ved hjelp av *NORM_X*-funksjonen som er tilgjengelig i TIA Portal. Normaliseringen går ut på at signalets verdier nedskaleres til en verdi mellom 0 og 1. Siemens har forhåndsdefinert båndbredden til PLSens A/D (analog til digital) konvertere til å være fra 0 til 27648, som gir en *overhead* av verdier da signalet består av 15 bit (32678 verdier). Siemens har forhåndsdefinert 27648 som maksimalverdi på analoge signaler, som tilsvarer 4mA-20mA eller 0V-10V målebredde. For å sørge for riktig representasjon av målingene må signalet normaliseres, slik at 0 blir definert som minimumsverdi og 27648 blir definert som maksimalverdi.

Deretter må signalet skaleres slik at instrumentets målebredde og måleverdi blir representert riktig. For eksempel en vanntank som blir målt prosentvis, må skaleres slik en tom tank på 0% tilsvarer verdien 0, og full tank på 100% tilsvarer maksverdien 27648, eller en temperatursensor med båndbredde fra -50°C til +120°C skal skaleres slik maks- og minimumsverdier settes til -50 og +120.
Når signalet er konvertert, normalisert og skalert, ligger det lagret på utgangen til blokken. Utgangen kobles sammen med ønsket indeks i datalisten i *DataStorage* for mellomlagring for senere bruk i programmet.



Figur 36: Bilde av funskjonsblokken "Converter".

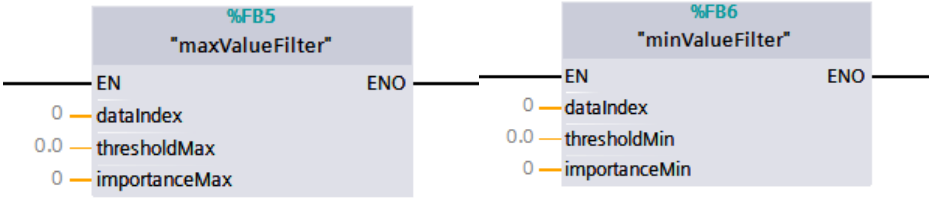
Filtre

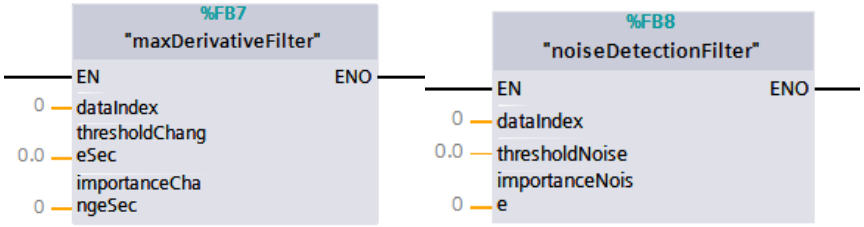
Filtreringen avhenger av hva slags type målesignal vi får inn. Ulike filtreringsmetoder kan være basert på målingens verdi over eller under en gitt *threshold*, dersom målingen havner mellom to bestemte verdier, målingens stigning (derivatet), signal av/på, positivt eller negativt signal, diverse uregelmessigheter i målesignalet og støy.

Etter at måledata er lagret i *DataStorage*, må den filtreres basert på brukeren valg av filtre og tilhørende parametere. Et sentralt krav for programmet er at måledata skal kun lastes opp dersom den er interessant eller kritisk, og derfor er det behov for at målingene går gjennom minst en filtrering for å avgjøre om dataen skal bli sendt til skyen. Det vil si at signalene som ikke har blitt markert som kritiske data i grad en, to eller tre, vil ikke bli lastet opp. Hva som defineres som betydningsfull informasjon er avhengig av hver enkel måleprosess og hva brukeren anser som viktig (*filterImportance*).

Et eksempel på dette kan være en batch-blanding der det er viktig at blandingen ikke faller under en bestemt temperatur. Ved implementering av programmet med dets filtre kan man sette parameterne slik at målt temperatur lastes opp kun når den faller under en fastsatt grense. Det medfører at en bruker kan monitorere prosessen når den oppfører seg merkverdig, og sparer lagringsplass og databruk ved å ikke laste opp målinger dersom verdiene holder seg på et akseptabelt nivå. Man kan dermed gå tilbake og se på lagrede målinger dersom noe har gått galt, og basert på hvordan filtreringen er satt opp og oppståtte avvik vil man muligvis kunne resonere frem til eventuell problemkilde.

Programmet har fire filtre som kan settes opp alene eller med hverandre. Filtrene er *maxValueFilter* som gir målingen kritisk status dersom verdien overstiger satt margin, *minValueFilter* som baserer seg på om målingen er lavere enn satt margin, *maxDerivativeFilter* som markerer målingen som kritisk dersom stigningen per sekund er høyere enn optimalt og *noiseDetectionFilter* som er en algoritme som registrerer hvor stor verdivariasjon målingen har. Filtrene har tre innganger for at brukeren kan definere måleverdien som skal analyseres, ønsket grenseverdi og filterets grad av kritisk tilstand.

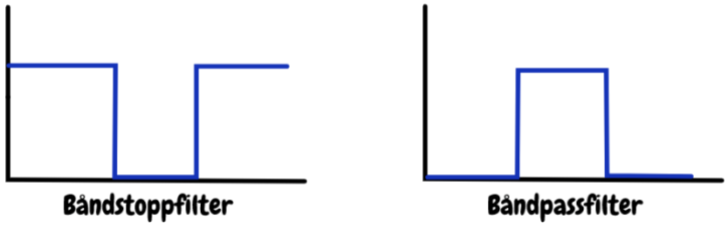




Figur 37: Bilde av funksjonsblokkene for de fire filtrene

maxValueFilter sammenligner det aktuelle målesignalet med grensen brukeren har satt. Dette skjer kontinuerlig hver gang programmet kjøres (hver syklus). Dersom målt verdi er større enn grenseverdien *importanceMax*, blir måledataen markert som kritisk i *DataStorage*. ***minValueFilter*** fungerer på samme måte som filteret beskrevet over, forskjellen er bare at måleverdien sjekkes for om den er lavere enn den valgte grensen. Filteret kan brukes med for eksempel en avstandssensor, slik at en melding blir sendt dersom avstanden er svært liten, på samme måte som en ryggsensor på en bil.

En kombinasjon av *minValueFilter* og *maxValueFilter* vil resultere i et båndpassfilter eller et båndstoppfilter, der man setter marginene over og under et verdiomfang for å kontrollere om signalet holder seg innenfor optimalt område.



Figur 38: Båndstopp- og båndpassfilter

maxDerativeFilter regner ut stigningen til målingen. Algoritmen som beregner dette er basert på formelen for approksimert numerisk derivasjon: $f'(x) = (f(x+h)-f(x)) / h$. Filteret finner stigningen ved å subtrahere nåværende måleverdi $(f(x+h))$ med måleverdien $f(x)$ fra forrige sampling. Deretter divideres differensen med tiden h mellom disse to målingene. Dersom absoluttverdien av denne stigningen er større enn forhåndsvalgt grense, vil denne verdien markeres som kritisk. Et slikt filter vil oppdage om et signal tyder på å være ustabilt eller beveger seg mot en uønsket tilstand.

noiseDetectionFilter er et filter som er laget for å forsøke å eliminere ulempen med at sampling med fast samplingsfrekvens ikke lagrer informasjon om hva som skjer mellom samplene. Filteret legger sammen absoluttverdien av endringene som skjer for hver syklus (omtrent hver 3. ms) i 50 sykluser og deretter trekker fra stigningen mellom første og siste måling. Algoritmen kan da registrere eventuelle hyppige nivåendringer (støy) på signalet. Denne funksjonen er nyttig for å detektere ødelagte instrumenter eller generelt høye støynivåer på signalet.

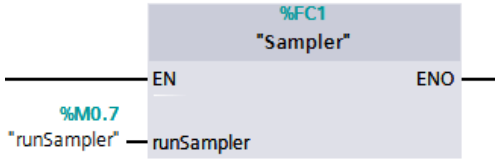
Alle filtrene må ta høyde for at det kan være andre filtre av ulik type som analyserer samme måledata. Grunnen til dette er at det kan oppstå problemer når programmet blir kjørt i en bestemt rekkefølge, som at det siste filteret som analyserer måledataen er det som avgjør hva dataens kritiske tilstand blir. Dette løses ved at alle filtrene har en algoritme som avgjør om den allerede satte tilstanden er høyere eller lavere enn tilstanden filteret ønsket å oppgi.

Dersom den bestemte tilstanden er høyere enn det filteret ønsket å endre den til, vil ikke filteret forandre dette. Om det er starten på en ny syklus av programmet derimot, må det første filteret ta dette med i beregningen og den gjeldende kritiske tilstanden hos det første filteret må overskrive den satte tilstanden, for å kunne oppdatere den hver gang en ny måling blir gjort.

Sampler

Vi vil ha en sampler for å trigge opplastingen av meldingene kontinuerlig. Når målesignalene er konvertert og filtrert, venter programmet på klarsignal fra sampleren, slik at målingene kan prepareres som en tekststreng og deretter sendes. Sampleren har to innganger, *sampleRate* og *runSampler*. Førstnevnte definerer samplingsfrekvensen til sampleren, som bestemmer hvor lang tid det skal gå mellom hvert målepunkt/sample. Sistnevnte er for å ha kontroll på når sampleblokken skal kjøres eller ikke. Selve sampleren baserer seg på en simpel *timer* som trigger variabelen *newSample* i datablokken *DataStorage*. Når timeren har telt til angitt samplingstid, vil den resettes med hjelp av en SR-vippe for å så begynne å telle på nytt.
SR-vippen fungerer slik at når set er på, går utgangen høy, slik at timeren aktiveres. Når set er lav igjen, skjer det ingen forandring før reset går høy, slik at utgangen går lav, altså resettes. En tradisjonell SR-vippes utgang er udefinert når både S=1 og R=1, men TIA Portal tillater denne tilstanden og tolker den som resett.

Når SR-vippa går høy, går første timeren på og teller ned tiden sin. Når timeren har telt ferdig, trigger den mellomblokka som resetter SR-vippen og setter i gang en annen timer. Det er behov for den andre timeren fordi vi ønsker å forsikre oss at utgangen av sampleren er høy lenge nok til at den registreres av MQTT-publisher, slik at den får beskjed at meldingen skal sendes.



Figur 39: Bilde av funksjonsblokken "Sampler"

Konstruksjon av melding

Funksjonsblokken *constructMsg* er til for å gjøre om måledata som skal sendes til en tekststreng i henhold til JSON (JavaScript Object Notation), som er meldingsformatet Watson IoT Platform opererer med. JSON strukturerer meldingen i form av at en tekst og data sendes parvis, og vi utnytter det til å sende måleverdiene sammen med deres tilhørende navn. For at skytjenesten skal klare å tolke meldingen er det svært viktig at all data representeres riktig, ved at datanavnet står i anførselstegn etterfulgt av kolon og den relevante verdien. Hele meldingen må også pakkes inn med krøllparenteser. Slik er et eksempel på en melding i JSON-format: {"datanavn":verdi,"datanavn2":verdi2}.

Måten funksjonsblokken former meldingen på er ved å først lete gjennom alle måleverdiene i datablokken *DataStorage*. Verdiene som ikke har blir markert med 0 i grad av viktighet (tilhørende *dataMarked*-grad) blir konvertert til tekstformat og plassert i en midlertidig streng. Konverteringen skjer ved at verdien blir multiplisert med 100, for å så bli omgjort til heltall for å fjerne desimalene og deretter blir dividert med 100 igjen. Det fører til at måledataene blir lagret som tekst med kun to desimaler. To desimaler dekker som regel kravet om målingens nøyaktighet, og ved restriksjon av antall desimaler sørger man for at meldingen ikke blir for

lang og tar begrenset mengde med data (færre desimaler = færre karakterer i meldingen = meldingen bruker mindre data).

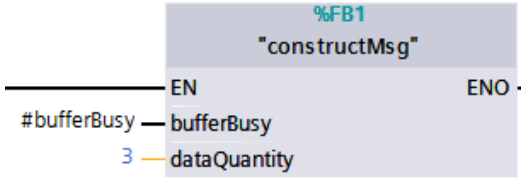
I tillegg til de relevante verdiene med navn, legges det også til en beskjed om den høyeste graden av den høyeste graden av kritisk måledata meldingen inneholder. Det er for å gi en indikasjon om hvor viktig meldingen er for brukeren. Høyeste graden av viktighet bestemmer også hvilken samplingsfrekvens som gjelder, som vil fortelle hvor lang tid det vil ta før neste melding sendes.

Tabell 13
Sammenhengen mellom viktighetsgrad på meldingen og samplingsfrekvens

Grad av viktighet	Samplingsfrekvens
1	20s
2	5s
3	1s

Det er også essensielt å vite til hvilken tid målingene ble samplet. Desom en melding ankommer forsinket eller mellomlagres i bufferen, er det viktig at meldingen inneholder et tidsstempel for når måledataene ble samplet. Tidsstempelet dannes ved bruk av funksjonen *RD_SYS_TIME*, som lagrer PLSens nåværende tid i en variabel av typen *DTL*. Når funksjonen kalles på vil variabelen oppdateres med nåværende data for år, måned, dag, time, minutt og sekund. Denne dataen konverteres dermed til en tekststreng i tidsformatet ISO8601, som er et format Watson IoT Platform støtter. Tidsstempelet legges til i meldingen parvis sammen med navnet *timestamp*. Et eksempel på det: "timestamp":"2019.01.01T13.37.59Z". Selve tidsstempelet kommer også i anførselstegn fordi dataen skal tolkes som en tekst, istedenfor bare en strøm av tallverdier.

Når meldingen er gjort om til tekst, struktureres alle verdiene og beskjedene med tilhørende datanavn parvis i JSON. Prosessen gjentar seg helt til alle måledataene i *DataStorage* er behandlet. Resultatet er en melding med all navngitt måledata gradert av filtre. Sånn ser ut et eksempel på en fullstendig melding med to målesignaler:
{"Temperatur":20.42,"Trykk":0.93,"Message":"Severity:3","timestamp":"2019.01.01T13.37.59Z"}.



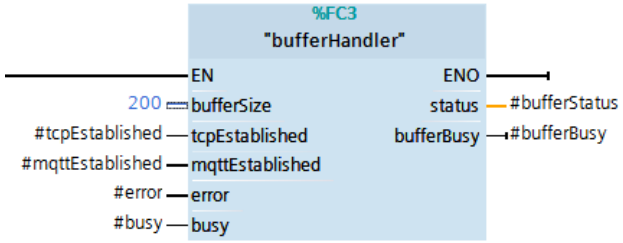
Figur 40: Bilde av funksjonsblokken "constructMsg"

Håndtering av buffer
Den siste funksjonsblokken programmet kjører per syklus er *bufferHandler*. Denne blokken gjør bestemmelser om meldingen skal sendes nå eller bufres, i tillegg til å tømme bufferen dersom det er mulig for øyeblikket. *bufferHandler* avgjør hva som skal skje med den konstruerte meldingen dersom den får beskjed om at den skal sendes, ved at variabelen *newSample* trigges av sampleren eller at samplingsfrekvensen endrer seg. Måten dette skjer på er at blokken gjør valg basert på statusen til utgangene på kommunikasjonsblokken MQTT_Publisher. Dersom kommunikasjonsblokken er tilkoblet brokeren, har ingen

feilmeldinger og er ikke opptatt, vil meldingen bli sendt fortløpende. Hvis *bufferHandler* får beskjed om at en melding skal sendes, men et eller flere av kriteriene for å kunne sende ikke er gyldig eller fraværende, vil meldingen istedenfor lagres i en buffer i *DataStorage*.
Når det er mulig, vil *bufferHandler* tømme bufferen. Det er viktig at bufferen ikke tømmes samtidig som det allerede foregår trafikk mellom kommunikasjonsblokken og brokeren/skyen. Måten dette er løst på er at *bufferHandler* sjekker bufferen hver gang det ikke skjer en sampling og opplastning. Hvis ingenting skjer, sjekkes hver indeks i bufferen. Dersom det eksisterer en melding i indeksen som blir sjekket, blir denne sendt til brokeren og deretter slettet fra bufferlisten. Så fort meldingen er sendt og ryddet, vil neste melding i bufferen sendes og så videre.

På denne måten klarer man å ivareta målinger man ønsker å sende dersom kommunikasjonsblokken ikke er i stand til å overlevere grunnet dårlig dekning, tekniske feil og lignende. Hvor mange meldinger som man er begrenset til å lagre samtidig er avhengig av størrelsen på bufferen, som igjen er avhengig av PLSens minne. En stor prosess med mange målesignaler krever også mange konvertere, mange filtre og mer kapasitet i *DataStorage*.

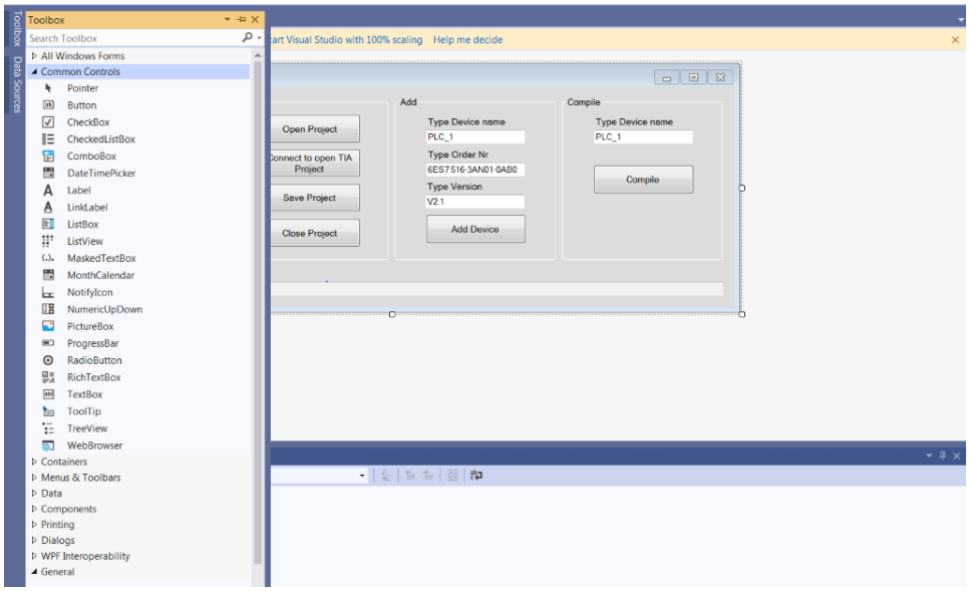
Dette er måten PLSen håndterer oppgavene på i en bestemt rekkefølge. PLSen arbeider gjennom programmet syklisk, som betyr at alle leddene i programmet blir kjørt om og om igjen i en løkke i svært høy hastighet. Hva som defineres som rask nok syklustid er avhengig av programmets krav. Dersom PLSen håndterer svært raske prosesser, kan syklustiden påvirke nøyaktigheten på resultatene negativt. Prosjektet vårt stiller ikke store krav til rask syklustid, da det ikke er behov for å oppdatere målesignalene oftere enn hvert 20-20ms. En forsinkelse på opplastingen av data til sky er heller ikke kritisk, med mindre den er på flere minutter. Programmets størrelse resulterer i at PLSen vi bruker har en syklustid på gjennomsnittlig 3ms, som er normal hastighet for et program på denne størrelsen.



Figur 41: Bilde av funksjonsblokken "bufferHandler"

4.10. Visual Studio og TIA Openness

Formålet med bruken av TIA Openness for denne oppgaven var for å finne ut hvorvidt programmet kan benyttes for å effektivisere bruken av TIA Portal. Ved å anvende TIA Openness gjorde vi en vurdering av hvor vanskelighetsgraden ligger ved utvikling av et program som utnytter APIets funksjoner. Vi hadde fokus på hvor intuitive API-funksjonene er å ta i bruk, og hvor tidskrevende det er å utvikle funksjonelle programløsninger.
Vi har utviklet et *interface* i Visual Studio for at brukeren enkelt kan få tilgang programmets funksjoner. Grensesnitt er en funksjon i Visual Studio av formen *Windows Application*, og gir oss muligheten til å utvikle et grensesnitt ut ifra redskaper som programmeringsmiljøet tilbyr.



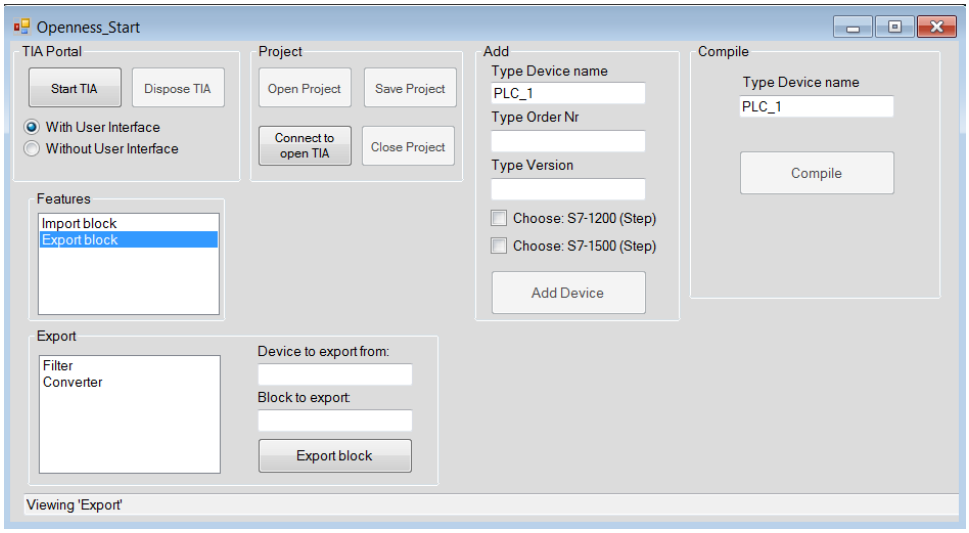
Figur 42: Figuren viser redskapene i Visual Studio som brukes til å utvikle grensesnittet.

Grunnmuren for vårt grensesnitt er basert ut ifra et demoprosjekt utviklet av Siemens. Dette demoprosjektet inneholder et grensesnitt med APIets grunnleggende funksjonaliteter, som blant annet å åpne allerede eksisterende PLS-prosjekter i TIA Portal, og starte og avslutte programmet.

Figur 43: Grensesnittet i demoprosjektet til Siemens

Vi tok utgangspunkt i grensenittet som kom med demoprosjektet til *Siemens*, og ekspanderte dette ved å inkludere flere funksjonaliteter. Vi endte opp med å utvikle funksjonene:

1. Mulighet til å enkelt velge s7-1200 PLSen vi fikk utdelt av oppdragsgiver som ny enhet.
2. Eksportering av allerede eksisterende funksjonsblokker fra en TIA Portal-prosjekt til en ønsket mappe på PCen.
3. Importering av forhåndsbestemte blokker fra en mappe til et TIA Portal-prosjekt.



Figur 44: Figuren viser grensesnittet vi endte opp med og dens funksjonaliteter.

Programstrukturen i Visual Studio-prosjektet vi utviklet består av flere scripts som til sammen utgjør det helhetlige programmet. Scriptene har sine unike funksjoner og fokuserer på ulike oppgaver.

Form1.cs er prosjektets hovedskript. Scriptet består av alle funksjonene som utføres i grensesnittet, og er i praksis "det som skjer under panseret" til det visuelle grensesnittet. Dersom brukeren for eksempel trykker på knappen *Start TIA*, vil tilhørende funksjon i scriptet utføres. Da scriptet tar i bruk de ulike funksjonene TIA Openness APIet tilbyr, må APIet inkluderes øverst i scriptet.

```
using Siemens.Engineering;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Hmi;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
```

Figur 45: Inkludering av TIA Openness' API-funksjoner i Form1.cs.

Form1.Designer.cs er skriptet lagrer informasjonen om det som blir laget i grensesnittet. Under utvikling av grensesnittet vil skriptet generere verdier som beskriver elementer som en knapps størrelse, plassering, farge, navn osv. Scriptet inneholder også direkte link mellom elementene i *interfacet* og de deres tilhørende funksjon i *Form1.cs*. Et eksempel på dette er dersom knappen *Example_Button* klikkes vil funksjonen *Example()* i *Form1.cs* kjøres ved hjelp av kodelinjen:

```
this.ExampleButton.Click += new System.EventHandler(this.Example)
```

Program.cs representerer det øverste laget i Visual Studio-prosjektet. Det er dette skriptet som blir *executed* når programmet kjøres, og kan sammenlignes med *main-funksjonen* andre programmeringsmiljøer. Scriptets er programmert slik at *Form1.cs* blir igangsatt ved oppstart, i tillegg til funksjoner for korrekt fremvisning av grensesnitt.

```
namespace StartOpenness
{
    0 references
    static class Program
    {

        [STAThread]
        0 references
        static void Main()
        {

            //Kjører applikasjonene ved oppstart
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Figur 46: Scriptet Program.cs har ansvar for å kjøre hovedscriptet Form1.cs

Tabell 14
Funksjonene vi har utviklet i prosjektet og hva de består av.

Funksjon	**Hva den gjør**
My Resolver	Gjør det mulig å starte TIA Portal ved å bruke TIA Openness.
Start TIA	Starter opp programmet TIA Portal.
Connect TIA	Oppretter tilkobling med et åpent prosjekt i TIA Portal.
Dispose TIA	Eliminerer tilkoblingen med det åpne TIA Portal-prosjektet.
Search Project	Gir bruker muligheten til å finne eksisterende prosjekt som er lagret på datamaskinen.
Open Project	Åpner eksisterende prosjekt.
Save Project	Lagrer TIA Portal prosjekt.
Close Project	Lukker TIA Portal prosjekt.
Get PLC Software	Brukes til å finne enheten som brukeren har lyst til å legge til på TIA Portal.
Compile	Kompilerer et TIA Portal prosjekt.
Add Device	Legger til en bestemt enhet i et TIA Portal prosjekt.
Export Blocks	Eksporterer blokker fra et TIA Portal prosjekt til en forhåndsbestemt mappe på datamaskinen.
Import System Blocks	Importerer blokker fra en forhåndsbestemt mappe på datamaskinen til et TIA Portal prosjekt.

My Resolver
Denne funksjonen er nødvendig for at TIA Openness APIet fungerer sammen med TIA Portal-programmet installert på PCen. Funksjonen sørger for at APIet og TIA Portal linkes sammen ved bruk av en egen nøkkel som ble generert under installasjon av TIA Portal. Nøkkelen befinner seg i operativsystemets registre og dens adresse er definert i funksjonen *My Resolver*.

Når funksjonen kjøres leter først programmet etter nøkkelen i registrene. Dersom nøkkelen ble funnet lagres den i en variabel som brukes i .NET-funksjonen *Assembly.loadFrom()* som aktiverer nøkkelen og oppretter en link mellom APIet og TIA Portal.

Start TIA
Gir brukeren mulighet til å starte *TIA Portal* via grensesnittet. Funksjonen gir brukeren valget til å starte *TIA Portal* med eller uten et grensesnitt. Når knappen *Start TIA* klikkes, vil API-funksjonen *new TiaPortal()* kjøres. Denne funksjonen starter og åpner et nytt TIA Portal-vindu.

Connect TIA
For å utføre operasjoner i et TIA Portal-prosjekt må Visual Studio-prosjektet tilkobles. På grunn av dette må brukeren alltid kjøre Connect TIA-funksjonen før ytterlige funksjoner som *Add Device, Import/Export Block* og *Save Project* kan utføres. *Connect TIA* utfører API-funksjonen *GetProcesses()* for å hente informasjon om det åpne TIA Portal-prosjektet. Dersom et prosjekt ble funnet kjøres så API-funksjonen *tiaProcess.Attach()* for at applikasjonen skal tilkobles projektet som ble funnet.

Dispose TIA
Funksjonen tas i bruk når brukeren har lyst til å avslutte tilkoblingen med en TIA Portal prosjektet som er i bruk, dette gjøres ved at API-funksjonen *MyTiaPortal.Dispose()* kjøres.

Search Project
Når knappen *Open Project* på grensesnittet blir klikket, blir *Search Project* kjørt. Funksjonen åpner et nytt *dialog-vindu,* der brukeren kan velge ønsket prosjektfil for TIA Portal. Dialog-vinduet filtrerer filene slik at man kun ser TIA Portals prosjektfiler (.ap15-filer). Dersom brukeren velger en fil, lagres filnavnet og dens adresse til en variabel for bruk i funksjonen *Open Project.*

Open Project
Funksjonen kalles i *Search Project* ved at den åpner opp prosjektet som ble valgt. Variabelen som ble lagret i *Search Project* brukes her slik at ønsket prosjektfil åpnes. I API-funksjonen *MyTiaPortal.Projects.Open()* utføres operasjonen som åpner den valgte prosjektfilten i TIA Portal.

Save Project
Funksjonen tas i bruk dersom brukeren velger å lagre prosjektet som er i bruk i *Tia Portal*. Da vil api-funskjonen *MyProject.Save()* kjøres, som vil lagre prosjektet i en forhåndsbestemt mappe på datamaskinen, der *MyProject* er definert som prosjektet som skal lagres.

Close Project
Funksjonen tas i bruk dersom brukeren velger å lukke prosjektet som er i bruk i *Tia Portal*. Da vil api-funksjonen *MyProject.Close()* kjøres, som vil lukke prosjektet.

Get PLC Software
For å legge til eller endre elementer i en enhet i et TIA Portal-prosjekt, må det defineres hvilken enhet (PLS) dette er i form av produkttype, prosessortype osv. De forskjellige enhetene kan ha ulike strukturer og virkemåter må defineres før endringer kan forekomme. Funksjonen *Get PLC Software* henter informasjon fra PLSens *softwareContainer* for valgt enhet slik at de API-funksjonene som krever dette kan utføres. Informasjonen i *softwareContaineren* lagres i en global variabel for bruk i andre funksjoner i applikasjonen.

Compile
Funksjon for å kompilere TIA Portal-prosjektet til ønsket enhet, slik at programmet er klart til å lastes opp og kjøres på enheten. *Compile*-funksjonen forteller *Get PLC Software* hvilken enhet som skal kompileres, og mottar den nødvendige informasjonen om enheten slik at programmet blir kompatibelt med enheten etter kompileringen.
Funksjonen benytter API-funksjonen *compiler.Compile()* for å utføre kompileringen i TIA Portal.

Add Device
Via grensesnittet har brukeren mulighet til å legge til flere enheter. Ved å fylle inn enhetens navn, unike ID og versjon-nummer kan ønsket PLS fra SIMATIC (undermerke av Siemens) legges til i det åpne TIA Portal-prosjektet. Funksjonen sjekker henter navnene til de allerede eksisterende enhetene i prosjektet, og legger til den ønskede enheten dersom den ikke har samme navn som de andre enhetene.
API-funksjonen som benyttes her er *Devices.CreateWithItem()* som mottar valgt parametere for navn, ID og versjon-nummer og oppretter en enhet av denne typen i TIA Portal-prosjektet.

Export Blocks
Dette er en funksjon for eksportering av funksjons- og datablokker i et åpent TIA Portal-prosjekt. Funksjonen består av API-funksjonen *plcBlock.Export()* som mottar navnet på blokken som skal eksporteres. Grensesnittet vi har laget er utstyrt med et tekstfelt der brukeren kan skrive inn navnet på blokken som skal eksporteres.
API-funksjonen eksporterer blokken som en fil i *.xml*-format, som er et markeringsspråk som lagrer all den informasjonen som trengs for å kunne rekonstruere blokken. .xml-filen lagres i en valgfri mappe på PCen.

```
<Access Scope="LocalVariable" UId="26">
  <Symbol>
    <Component Name="reset" />
  </Symbol>
</Access>
<Access Scope="LocalVariable" UId="27">
  <Symbol>
    <Component Name="srvippe" />
  </Symbol>
</Access>
<Access Scope="LocalVariable" UId="28">
  <Symbol>
    <Component Name="sampleRate" />
  </Symbol>
</Access>
<Part Name="Coil" UId="29" />
<Part Name="TP" Version="1.0" UId="30">
  <Instance Scope="GlobalVariable" UId="31">
    <Component Name="IEC_Timer_0_DB_1" />
  </Instance>
  <TemplateValue Name="time_type" Type="Type">Time</TemplateValue>
</Part>
<Part Name="Coil" UId="32" />
<Part Name="Sr" UId="33" />
<Part Name="TON" Version="1.0" UId="34">
```

Figur 47 - Utdrag fra en .xml-fil som inneholder data for en funksjonsblokk

Import Blocks
Dersom funksjons- eller datablokker er eksportert med Export Blocks-funksjonen, kan de genererte .xml-filene brukes for å importere blokkene inn i et TIA Portal-prosjekt. API-funksjonen *Block.Import()* mottar addresse og navn til .xml-fila som skal importeres, og importerer denne inn som enn blokk på valgt enhet i TIA Portal-prosjektet.
Det er også implementert en liste i grensesnittet for at brukeren kan se hvilke .xml-filer som er tilgjengelig for importering. Listen har en tilhørende *Refresh-knapp* som oppdaterer listen ved å hente navn og adresse til .xml-filene som befinner seg valgt mappe på PCen.

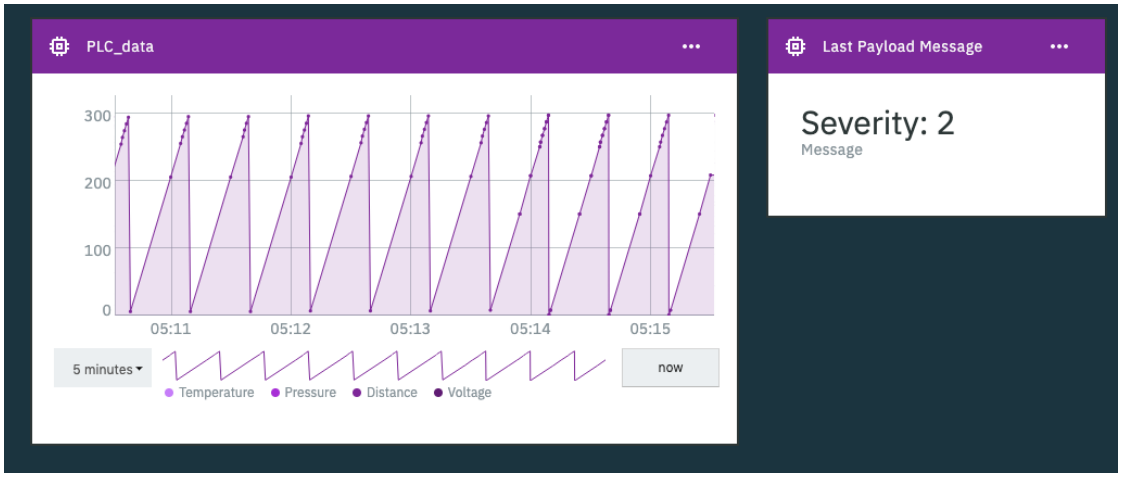
5. Resultater
Produktet vårt er en IT-løsning for overføring av utvalgt måledata fra en S7-1200 PLS til en skytjenesteleverandøren IBM.

Vi har valgt ut en kommunikasjonsprotokoll og en skytjeneste for overføringen av måledata i stor grad basert på brukervennlighet. MQTT-protokollen er kjent for bruk i IoT-løsninger og telemetri, og fordi IBM-teamet har utviklet MQTT, er skytjenesten skreddersydd for kommunikasjon over denne protokollen. IBM Cloud (IBM Watson IoT Platform) fungerer også som en broker, noe som forenkler oppsettet.

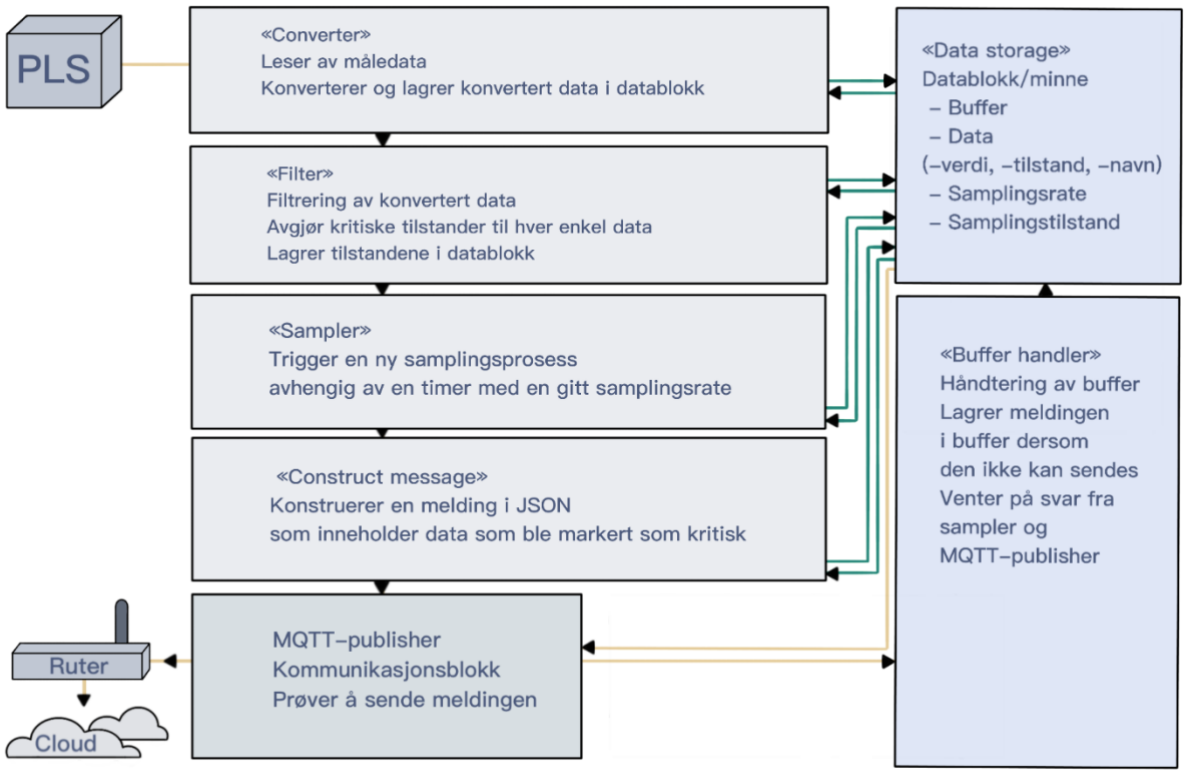
Løsningen innebærer kontoopprettelse og PLS-oppsett hos skytjenesteleverandøren, ruterkonfigurasjon av SCALANCE M874-2 og et PLS-program.

Selve PLS-programmet er laget i TIA Portal, og sørger for kontinuerlig opplasting til en broker. Programmet består av Siemens sin MQTT-publisher, en funksjonsblokk som konverterer målesignalene fra analoge inngangene og skalerer de, en sampler som trigger opplastning basert på gitt samplingsfrekvens og *av/på*-funksjon, en datablokk som inneholder blant annet buffer og metadata til målingssignalene som skal lastes opp, en funksjonsblokk som håndterer *buffering* av data, en funksjonsblokk som gjør målesignalene om til melding som skal sendes i en *string*-format og et (eller flere) filter som bestemmer hva slags data skal overføres til skyen.

Filtrene man har tilgjengelig er maxValueFilter (høypass), minValueFilter (lavpass), maxDerivativeFilter og noiseDetectionFilter. Disse kan brukes i kombinasjon med hverandre. Fordi PLS-programmet er laget i form av flere moduler, er det relativt enkelt å modifisere hvis det er behov for det og tilpasning til et annet prosjekt.



Figur 48: Visualisering av innkommende data fra PLSen i Watson IoT Platform.



Figur 49: Bildet visualiserer PLS-programmet i sin helhet

Ved bruk av Visual Studio utviklet vi et grensesnitt som gjør operasjoner i TIA Portal med TIA Openness. Grensesnittet inneholder elementære funksjoner som benytter noen av API-funksjonene TIA Openness tilbyr, og gir brukeren mulighet til å utføre enkle oppgaver i TIA Portal, som generell håndtering av TIA Portal-prosjekter, importering og eksportering av blokkdata.

6. Diskusjon

Filteret til PLS-programmet inneholder flere parametere som man kan fylle ut med ønskede grenseverdier. Slik oppfyller filteret kravet om at det skal trigges når målesignalet beveger seg utenfor definert område, og resten av programmet sørger for å sende opp en melding med måledata og tidsstempel til skyplattformen. Dette fortsetter programmet å gjøre så lenge målingene tyder på at noe uønsket eller unormalt skjer.

Systemløsningen sjekker måledataen avhengig av samplingsrate. Det vil si at den fastsatte samplingsraten bestemmer oppløsningen på programmet og dets evne til å lese av, analysere og behandle måledataen. Programmet har en funksjon for dynamisk sampling som lar brukeren sette parametere, slik at samplingsraten øker automatisk avhengig av den kritiske graderingen av målingene. Kritisk måledata krever ofte høy samplingsrate for å analyseres tilstrekkelig, da disse kan være ustabile og ha raske endringer. Dynamisk samplingsrate gir programmet en evne til å forstå når flere punktprøver ønskes, som vi anser som en nyttig egenskap for brukeren å ha tilgang til.

Når en skal se på dataen som er blitt lastet opp, vil man kunne se når signalene beveger seg mot og i kritiske tilstander og tidspunktet på når de er målt, noe som er optimalt hvis formålet er overvåkning av fjerntliggende anlegg og måleinstrumenter. Ulempen med filteret er at hvis en uønsket endring skjer på kortere tid enn gitt samlingsrate, for eksempel en rask *peak* mellom to stikkprøver, vil ikke programmet kunne detektere hendelsen. Dette er prinsippet i Nyquists teorem, som går ut på at samplingsraten bør være det dobbelte av frekvensen på signalet som punktprøves.

Et alternativt og mer dynamisk design på filteret ville være å lagre måledata som baserer seg på endring avhengig av forrige måling, i stedet for en gitt samplingsrate. Hvis endringen er spesielt stor mellom to målinger, en grenseverdi som defineres av brukeren, vil filteret trigges og ta et antall stikkprøver som er avhengig av hvor stor endringen (stigningen eller synkingen) er. Et eksempel kan være at vi skal begynne å sample en temperaturmåler hver gang verdien endrer seg med 0,5° grader eller mer. Om man måler 20,3° ute, vil ikke neste måling skje før temperaturen har endret seg til enten 19,8° eller 20,8°. Slik forsikrer man seg at all endring blir notert, og er ideelt dersom man er ute etter å kun se etter unormal oppførsel. Det vil også være fordelaktig å forsikre seg at all interessant data blir målt og at ingenting går tapt mellom stikkprøvene. Til tross for dette er dette begrenset av maksimum samplerate som er bestemt av PLSens syklustid. Syklustiden er avhengig av prosessortype og størrelsen på programmet, men ligger som regel mellom 2-10ms.

Det betyr at når skyen mottar alle måledata så kan man i utgangspunktet rekonstruere en kurve som kan være nyttig i en feilsøkingsprosess. For å unngå at PLSen på et visst punkt vil *spamme* (sende en enorm mengde data over kort tid) skyen med meldinger på grunn av hyppig variasjon i målingene og føre til et stort forbruk av data, må programmet kunne lagre en samling med målinger innen en bestemt tid og strukturere disse i én melding som deretter sendes på samme måte. En løsning som dette krever at hver eneste måling har inkludert et tidsstempel som beskriver tidspunktet målingen ble gjort, og øker databruken drastisk da meldingene kan bli opptil dobbelt så lange ved å inneholde tidsstempler.

Et annet problem med en slik filtrering er at dersom et målesignal har mye støy vil dette sannsynligvis bli fanget opp av algoritmen og igjen føre til en ekstrem mengde med opplasting av målesignalet. Dette kan også resultere i en kostbar faktura fra nettleverandøren og eventuelt skytjenesten. En annen mulig komplikasjon kan være at PLSen ikke klarer å sende meldingene raskt nok ved veldig stor pågang at kritisk data, som vil føre til at bufferen fylles helt opp slik at viktig data kan gå tapt.

Samplingsmetoden programmet bruker er basert på en gitt samplingsfrekvens. Samplingsfrekvens er en fast tid mellom hver måling/opplastning, som fører til at programmet kan gå glipp av interessant informasjon som skjer mellom hvert sample. For å minimere denne ulempen har vi valgt å implementere et filter som oppdager store variasjoner uavhengig av samplingstid, i tillegg til at vi har laget funksjonen dynamisk samplerate. Ulempen med filteret *noiseDetectionFilter* er at dersom det oppdager støy eller lignende på signalet vil ikke nødvendigvis dette vises på måledataene i skytjenesten i form av hvilken algoritme og filter som har analysert det.

Dersom kommunikasjonen brått termineres eller lignende, har ikke programmet noen funksjon for å enten koble seg på eller opplyse brukeren om dette. Dersom programmet brukes for å måle viktig informasjon kan dette være et stort problem, da brukeren forventer at alt fungerer som det skal. Programmets buffer kan ta høyde for slike problemer ved å lagre usendte meldinger, men bufferens størrelse er begrenset av minnet til PLSen. En løsning på dette kan være redundans i form av å ta i bruk en ekstra ruter, som kan opprettholde kommunikasjon med skytjenesten dersom den andre ruteren feiler.

Kort vurdering av TIA Openness
Etter å ha opparbeidet noe erfaring med TIA Openness og dens virkemåte har vi fått et innblikk i APIets potensiale, brukervennlighet og hvor mye det krever av utvikleren. Vi erfarte at APIet er tidskrevende å sette seg inn i og krever grundig forståelse av både programmering i C# og relativt mye erfaring med TIA Portal. Dersom man ønsker å ta i bruk de mindre avanserte API-funksjonene som *Start TIA* og *Open Project*, er TIA Openness lite krevende ta i bruk, men om man ønsker å implementere mer avanserte løsninger ble APIet fort både innviklet og programmeringstungt.

Vi mener at TIA Openness kan være et svært nyttig verktøy som gir utvikleren mulighet til å skape interessante løsninger for TIA Portal som effektiviserer tidskrevende gjøremål. Dessverre krever dette også mye tid, kunnskap og programmeringserfaring å utvikle.
TIA Openness er for mange et ukjent produkt av Siemens og har en relativt liten brukergruppe, som merkes i form av begrensningen av nyttig informasjon som finnes på nettet i form av tidligere løsninger og demonstrasjoner. Grensesnittet er godt dokumentert av Siemens, men mangler flere konkrete forklaringer på APIets funksjoner. Dokumentasjonen som finnes omhandler for det meste eksempelkode, som ikke gir godt nok grunnlag for forståelse av APIets funksjoner. Dette kan gjøre det problematisk å tilpasse API-funksjonene til bruk i egne applikasjoner.

7. Konklusjon

7.1 Konklusjon

Prosjektoppgaven gikk ut på å vurdere skytjenester og protokoller, sette opp kommunikasjon mellom skytjeneste og PLS og utvikle et program som bruker filtreringsmetoder som grunnlag for opplastning av måledata til skytjenesten. I tillegg skulle vi forsøke å utvikle et program med TIA Openness for generering av prosjekter i TIA Portal for å gi en vurdering av TIA Openness.

Kravene som ble stilt til prosjektet er opprettholdt i form av vurderinger og oppsett. Vi har foreslått flere mulige og aktuelle sikkerhetstiltak på dataoverføringen. PLSen kommuniserer vellykket med valgt skytjeneste over en kommunikasjonsprotokoll som krever svært lite båndbredde, som er viktig da vi kommuniserer over mobilnett.

7.2. Videre arbeid

Det er et behov for gode filteralgoritmer i bransjen. Som nevnt tidligere, er en mer dynamisk og kompleks sampling av måledata svært interessant for feilsøking og rekonstruering av en kurve som kan tyde på hva som har kan ha skjedd og hvorfor, eller analysere måledata for å forutse hva som kommer til å skje. For et målesignal som stiger i verdi, og over en ønsket terskel, er det aktuelt å øke samplingsraten akkurat da og i takt med stigningen for å få med så mangle punkter som er fordelaktig til å skape en detaljert akse. Samtidig som dette skjer, må

ikke systemet overkjøres med meldinger, og bør samle stikkprøver av målesignalene og sende de samlet i én melding i en lavere sendefrekvens. Dette er også mer krevende for sensitive målere hvor veldig raske endringer kan skje og forsvinne.

8. Kildeliste

PLS

AMCI, Advanced Micro Controls Inc. (u.å.) What is a PLC? Hentet fra https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/what-plc/

Stephen Gates (2017, 28. november) A Beginner's PLC Overview. Hentet fra https://www.automation.com/automation-news/article/a-beginners-plc-overview-part-1-of-4-introduction-to-plcs

Kevin Cope (2018, 22. februar) Who is the father of the PLC and why was it invented? Hentet fra https://realpars.com/father-of-the-plc/

Camille Gilbert (2014, 18. august) Advantages of a Modular PLC over a Fixed PLC? Hentet fra https://blog.cetrain.isu.edu/blog/bid/353287/Advantages-of-a-Modular-PLC-Over-a-Fixed-PLC

VPN

Levent Sapci (2016, 7. juni) Site-to-Site and Remote-Access VPN: What's The Difference https://blog.hotspotshield.com/2016/06/07/site-to-site-remote-access-vpn-whats-difference/

John Mason (2019, 26. Februar) VPN Beginner's Guide. Hentet fra https://thebestvpn.com/what-is-vpn-beginners-guide/

Jeff Tyson, Stephanie Crawford (2018, 28. Juni) How VPNs Work. Hentet fra https://computer.howstuffworks.com/vpn.htm

Andrew Tarantola (2013, 16. Mars) VPNs: What They Do, How They Work, and Why You're Dumb for Not Using One. Hentet fra https://gizmodo.com/vpns-what-they-do-how-they-work-and-why-youre-dumb-f-5990192

VPN Protokoller

VPN One Click (2016, 28. Mai) Types of VPN and types of VPN Protocols. Hentet fra https://www.vpnoneclick.com/types-of-vpn-and-types-of-vpn-protocols/

Gvain Phillips (2017, 12. Oktober) The 5 Major VPN Protocols Explained. Hentet fra https://www.makeuseof.com/tag/major-vpn-protocols-explained/

Kees Friesland (2018, 15. Juli) Which VPN Protocol You Should Use – Five of The Most Common VPN Protocols Compared. Hentet fra https://www.technadu.com/vpn-protocols/8436/

Tim Mocan (2019, 18. Januar) What Is A VPN Protocol & What Is the Best VPN Protocol? Hentet fra https://www.cactusvpn.com/beginners-guide-to-vpn/vpn-protocol/

IBM Cloud (2019, 26. April) Set Up IPSec VPN. Hentet fra https://cloud.ibm.com/docs/infrastructure/iaas-vpn?topic=VPN-setup-ipsec-vpn

API

Mulesoft (2018, 13. Februar) What is an API? (Application Programming Interface) Hentet fra https://www.mulesoft.com/resources/api/what-is-an-api

IBM Cloud

IBM Cloud (2019, 14. Mai) IBM Cloud Products. Hentet fra

https://www.ibm.com/cloud/products
IBM Cloud (2019, 19. Mai) IBM Cloud pricing. Hentet fra
https://www.ibm.com/cloud/pricing
IBM Cloud (2019, 12. April) Global locations and resiliency options for IBM Cloud Object Storage. Hentet fra
https://www.ibm.com/cloud/object-storage/resiliency
IBM Cloud (2019, 29. Mars) Security in the IBM Cloud. Hentet fra
https://www.ibm.com/cloud/security
IBM Cloud (2018, 1. Oktober) IBM Cloud support. Hentet fra
https://www.ibm.com/cloud/support
IBM Cloud (2017, 28. Juli) Global locations for your business. Hentet fra
https://www.ibm.com/cloud/data-centers/

Azure
Azure (u.å.) Azure regions. Hentet fra
https://azure.microsoft.com/en-us/global-infrastructure/regions/
Azure (u.å.) Storage. Hentet fra
https://azure.microsoft.com/en-us/product-categories/storage/
Azure (u.å.) Azure Security Center. Hentet fra
https://azure.microsoft.com/en-us/services/security-center/
Azure (u.å.) Azure Pricing. Hentet fra
https://azure.microsoft.com/en-us/pricing/
Azure (u.å.) Compare support plans. Hentet fra
https://azure.microsoft.com/en-us/support/plans/

Amazon Web Services
Amazon (2019, 17. Mai) AWS Pricing. Hentet fra
https://aws.amazon.com/pricing/?nc2=h_ql_pr
Amazon (2019, 17. Mai) Global Infrastructure. Hentet fra
https://aws.amazon.com/about-aws/global-infrastructure/
Amazon (2019, 2. Mai) Cloud Storage with AWS. Hentet fra
https://aws.amazon.com/products/storage/?nc2=h_m1
Amazon (2019, 2. Mai) Cloud security, Identity & Compliance with AWS. Hentet fra
https://aws.amazon.com/products/security/?nc2=h_m1
Amazon (2019, 2. Mai) Compare AWS Support Plans. Hentet fra
https://aws.amazon.com/premiumsupport/plans/

Google Cloud
Google Gloud (2019, Januar) Pricing. Hentet fra
https://cloud.google.com/pricing/
Google Gloud (u.å) Cloud locations. Hentet fra
https://cloud.google.com/about/locations/
Google Gloud (u.å) Security. Hentet fra
https://cloud.google.com/security/
Google Gloud (u.å) Cloud storage products. Hentet fra
https://cloud.google.com/products/storage/
Google Gloud (u.å) Google Cloud Platform support. Hentet fra
https://cloud.google.com/support/

Linker til privat, offentlig og hybrid skytjeneste
Lasse Nordvik Wedø (2016, 7. November) Public, private eller hybrid – hva er i skyen? Hentet fra https://blogg.telenorinpli.no/public-private-eller-hybrid-hva-er-hva-i-skyen
Azure (u.å.) What are public, private and hybrid clouds? Hentet fra https://azure.microsoft.com/en-gb/overview/what-are-private-public-hybrid-clouds/
Muhammad Raza (2018, 12. September) Public Cloud vs Private Cloud vs Hybrid Cloud: What's the Differnce? Hentet fra https://www.bmc.com/blogs/public-private-hybrid-cloud/

Linker til IaaS, SaaS og PaaS
Stephen Watts (2017, 22. September) SaaS vs PaaS vs IaaS: What's The Difference and How To Choose. Hentet fra https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/
Atos (2017, 7. November) Iaas, PaaS, SaaS (Explained and Compared). Hentet fra https://apprenda.com/library/paas/iaas-paas-saas-explained-compared/
Datatilsynet (2018, 23. Juni) Skytjenester. Hentet fra https://www.datatilsynet.no/personvern-pa-ulike-omrader/internett-og-apper/skytjenester/

Kryptering
Bryan Clark (2015, 9. Mars) How Does Encryption Work, and Is It Really Safe? Hentet fra https://www.makeuseof.com/tag/encryption-care/
Mohit Arora (2015, 24. September) How Safe is AES Encryption? | Advanced Encryption Standard. Hentet fra https://www.kryptall.com/index.php/2015-09-24-06-28-54/how-safe-is-safe-is-aes-encryption-safe
Search Encrypt (2017, 27. November) What is Encryption & How Does It Work? Hentet fra https://medium.com/searchencrypt/what-is-encryption-how-does-it-work-e8f20e340537

Protokoller
Margaret Rouse (2018, November) Network protocol. Hentet fra https://searchnetworking.techtarget.com/definition/protocol
Michael Yuan (2017, 7. august) Getting to know MQTT. Hentet fra https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/
Sean Dague (2018, 7. juni) Using MQTT to send receive data for your next project. Hentet fra https://opensource.com/article/18/6/mqtt
The HiveMQ Team (2015, 11. mai) MQTT Security Fundamentals: TLS/SSL. Hentet fra https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl/
The HiveMQ Team (2015, 26. januar) MQTT Essentials Part 3: Client, Broker and Connection Establishment. Hentet fra https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/
Rabbitmq (2006, juni) AMQP Protocol Specification. Hentet fra https://www.rabbitmq.com/resources/specs/amqp0-8.pdf
AMQP (u.å.) AMQP is the Internet Protocol for Business Messaging. Hentet fra https://www.amqp.org/about/what
Margaret Rouse (2018, januar) Advanced Message Queuing Protocol. Hentet fra https://whatis.techtarget.com/definition/Advanced-Message-Queuing-Protocol-AMQP

Peter Ledbrook (2010, 14. juni) Understanding AMQP. Hentet fra https://spring.io/blog/2010/06/14/understanding-amqp-the-protocol-used-by-rabbitmq/
Margaret Rouse (2017, januar) Representational State Transfer. Hentet fra https://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer
Margaret Rouse (2016, desember) RESTful API. Hentet fra https://searchmicroservices.techtarget.com/definition/RESTful-API
Claire Hunsaker (2015, 18. mai) REST vs SOAP: When is REST better? Hentet fra https://stormpath.com/blog/rest-vs-soap
The Secret Securiry Wiki (u.å.) Simple Object Access Protocol. Hentet fra https://doubleoctopus.com/security-wiki/protocol/simple-object-access-protocol/
Anna Monus (2019, januar) SOAP vs REST vs JASON comparison. Hentet fra https://raygun.com/blog/soap-vs-rest-vs-json/
XMPP (u.å.) An Overview of XMPP. Hentet fra https://xmpp.org/about/technology-overview.html
Isode (2018, September). Operating XMPP over HF Radio and Constrained Networks. Hentet fra https://www.isode.com/whitepapers/low-bandwidth-xmpp.html
Security Compass (2016, 31. Mai) XMPP: Swiss Army Knife for Internet of Things. Hentet fra https://blog.securitycompass.com/xmpp-swiss-army-knife-for-internet-of-things-iot-9eff783c44ba
Chris Love (2019, 11. mai) How HTTPS Works to Keep You Secure and How it Differs From HTTP. Hentet fra https://love2dev.com/blog/how-https-works/
Margaret Rouse (2010, August) XML File Format. Hentet fra https://whatis.techtarget.com/fileformat/XML-eXtensible-markup-language
XMLObjective (2013, 11. juli) What is the difference between XML and HTML? Hentet fra http://www.xmlobjective.com/what-is-the-difference-between-xml-and-html/
Oracle (2012, 1. Juni) Introduction to Simple Authentication Security Layer. Hentet fra https://docs.oracle.com/cd/E23824_01/html/819-2145/sasl.intro.20.html
CloidAMQP (u.å.) Stomp. Hentet fra https://www.cloudamqp.com/docs/stomp.html
Stomp (u.å.) Stomp Protocol Specification, v1.1. Hentet fra https://stomp.github.io/stomp-specification-1.1.html
Harshvardhan Mishra (2019, 29. januar) STOMP, Simple/Streaming Text Oriented Messaging Protocol. Hentet fra https://iotbyhvm.ooo/stomp/
Margaret Rouse (2014, desember) Simple Object Access Protocol. Hentet fra https://searchmicroservices.techtarget.com/definition/SOAP-Simple-Object-Access-Protocol
Guy Levin (2018, 5. april) Introduction to REST API Security. Hentet fra https://blog.restcase.com/introduction-to-rest-api-security/
Ranjith Kumar (2018, 11. april) MQTT vs REST from IoT Implementation perspective. Hentet fra https://www.bevywise.com/blog/mqtt-vs-rest-iot-implementation/
Marina Serozhenko (2017, 20. mars) MQTT vs HTTP: Which one is the best for IoT? Hentet fra https://medium.com/mqtt-buddy/mqtt-vs-http-which-one-is-the-best-for-iot-c868169b3105
Charlie Wang (2018, 26. november) HTTP vs MQTT: A Tale of two IoT protocols. Hentet fra

https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols
Kristopher Sandoval (2017, 7. Desember) Defining Steteful vs Stateless Web Services. Hentet fra
https://nordicapis.com/defining-stateful-vs-stateless-web-services/

Implementering og diverse

GTAI (2017, 7. April) Industrie 4.0. Hentet fra
https://www.gtai.de/GTAI/Navigation/EN/Invest/Industries/Industrie-4-0/Industrie-4-0/industrie-4-0-what-is-it.html
IBM Cloud Docs, Internet of Things Platform (2019, 16. april) Getting started tutorial. Hentet fra
https://cloud.ibm.com/docs/services/IoT?topic=iot-platform-getting-started
IBM Knowledge Center (2019, 14. Mai) Adding the TIMESTAMP format attribute to a logical interface property. Hentet fra
https://www.ibm.com/support/knowledgecenter/en/SSQP8H/iot/troubleshooting/li_timestamp_property.html
IBM Knowledge Center (2019, 14. Mai) Connecting devices. Hentet fra
https://www.ibm.com/support/knowledgecenter/en/SSQP8H/iot/platform/iotplatform_task.html
Brian Innes (2018, 30. januar) Use MQTT to send data to Watson IoT Platform. Hentet fra
https://developer.ibm.com/tv/mqtt-client-library-watson-iot-platform/
Wolfram MathWorld (oppdatert 2019, 15. mai) Numerical Differentiation. Hentet fra
http://mathworld.wolfram.com/NumericalDifferentiation.html
JSON (u.å.) Introducing JSON. Hentet fra
https://www.json.org
Siemens (2018, 16. august) MQTT Publisher for SIMATIC CPU. Hentet fra
https://support.industry.siemens.com/cs/document/109748872/mqtt-publisher-for-simatic-cpu?dti=0&lc=en-WW
Siemens (2018, August) MQTT Publisher (unencrypted) for the S7-1500/S7-1200 and S7-300. Hentet fra
https://support.industry.siemens.com/cs/attachments/109748872/109748872_MqttClient_Publish_Unsecure_DOKU_V1_1_en.pdf
Kevin Purdy (2011, 13. April) What Is the .NET Framework, and Why Wo I Need It? Hentet fra
https://lifehacker.com/what-is-the-net-framework-and-why-do-i-need-it-5791578
Siemens (2019, januar) Industrial Remote Communication, Remote Networks, Configuring VPN tunnel. Hentet fra
https://support.industry.siemens.com/cs/attachments/109764618/GS_SCALANCE-M800-VPN-Tunnel_76.pdf
SIMENS (2016, 1. APRIL) FOR AN S7-1200/S7-1500 CONTROLLER IN STEP 7 (TIA PORTAL), HOW DO YOU SCALE INTEGER VALUES IN REAL NUMBERS AND VICE VERSA FOR ANALOG INPUTS AND OUTPUTS? HENTET FRA:

https://support.industry.siemens.com/cs/document/39334504/for-an-s7-1200-s7-1500-controller-in-step-7-(tia-portal)-how-do-you-scale-integer-values-in-real-numbers-and-vice-versa-for-analog-inputs-and-outputs-?dti=0&lc=en-ES

Simens TIA Openness

Simens (2019, 3. mai) TIA Portal Openness: Introduction and Demo Application. Hentet fra: https://support.industry.siemens.com/cs/document/108716692/tia-portal-openness%3A-introduction-and-demo-application?dti=0&lc=en-WW
SIMENS (2016, 3. NOVEMBER) OPENNESS - EFFICIENT GENERATION OF PROGRAM CODE USING CODE GENERATORS. HENTET FRA:

https://www.youtube.com/watch?v=Ki12pLbEcxs
Siemens (2018, 20. Desember) Openness: Automating creation of projects. Hentet fra https://cache.industry.siemens.com/dl/dl-media/163/109477163/att_929953/v7/117337971723/en-US/index.html#treeId=d1f9123e56ff885898b3412a040f4065

9. Vedlegg

Vedlegg 1: FDS (Functional Design Specification)
Vedlegg 2: WBS (Work Breakdown Structure)
Vedlegg 3: Gantt-diagram
Vedlegg 4: Flytskjema: Sampler
Vedlegg 5: Flytskjema: Filtre
Vedlegg 6: Flytskjema: BufferHandler
Vedlegg 7: Flytskjema: Converter
Vedlegg 8: Flytskjema: ConstructMsg
Vedlegg 9: PLS-programmets kode i strukturert tekst
Vedlegg 10: TIA Openness-kode