TEKNISKRAPPORT

# FRA SENSOR TIL SAMFUNN

Valg av teknologier, utfordringer og løsninger

Versjon 1.01

Norkart avd. Kristiansand

**RAPPORT**

**Teknisk rapport – Valg av teknologier, utfordringer og løsninger**

**Emne:** Valg av teknologier, utfordringer og løsninger

**Versjon**: 1.01

**Dato:** 19. August 2022

**Forfattere/redaktør:** Thomas Låg, Joachim Western, Julie Tvergrov, Ida-Marie Solli og Karen Johanna Thorsen, Norkart avd. Kristiansand.

**Kvalitetskontroll:** Alexander Salveson Nossum, Norkart avd. Kristiansand.

# Innhold

# Forkortelser og begrepsavklaring

**API** – Application Programming Interface, et hjelpeverktøy ved programmering. Et grensesnitt mot en eller flere tjenester i et operativsystem.

**AWS** – Amazon Web Services.

**Data Aggregator** – Sammenstiller informasjonen fra en eller flere kilder med hensikt å utarbeide kombinerte datasett for databehandling.

**ERP** – Enterprise Resource Planning. Programvaresystem som støtter **opp** om flere av en bedrifts virksomhetsområder, som; produksjon, lager, salg, innkjøp og økonomi. Målet er å håndtere virksomhetens informasjon og tilfredsstille behov for styring og administrasjon.

**FN** - En global internasjonal organisasjon grunnlagt i 1945. Etterfølger av Folkeforbundet, for å stoppe krig og danne en plattform for dialog. Består av 193 medlemsland.

**GDPR** – Personvernforordningen, engelsk: General Data Protection Regulation. Skal styrke og harmonisere personvernet ved behandling av personopplysninger i Den europeiske union.

**HTTP Endpoint** – Et endepunkt er en URL for webtjeneste, også en distribuert API. SOAP-endepunktet (Simple Object Access Protocol) er en URL. Den identifiserer plasseringen på den innebygde HTTP-tjenesten der nettjenestelytteren lytter etter innkommende forespørsler.

**IDE** – Integrated development environment.

**InfluxDB** – En åpen kildekode-tidsseriedatabase utviklet av selskapet InfluxData.

**IoT** – Internet of Things, kjent som Tingenes internett. Et system av sammenhengende databehandlingsenheter, mekaniske eller digitale maskiner, objekter, som er utstyrt med unike identifikatorer og evnen til å overføre data over et nettverk uten å kreve kontakt mellom mennesker eller mellom menneske og maskin.

**IP** – En unik identifikator eller adresse som tildeles en enhet, for eksempel en PC eller en skriver i et TCP/IP-basert datanettverk.

**KS** – Kommunesektorens Organisasjon, kommunesektorens interesse- og arbeidsgiver-organisasjon i Norge. Arbeider for en effektiv og selvstendig kommunesektor som ivaretar innbyggernes behov.

**Kubernetes** – Et system for programvarekontainere basert på åpen kildekode som er utviklet av Google, med hensikt for automatisk distribusjon, skalering og operasjoner av kontainere på tvers av klynger av datamaskiner som verter.

**LoRa** – LoRaWAN. En type trådløst telekommunikasjons-wide area-nettverk designet for å tillate langdistansekommunikasjon med lav bit hastighet.

**MongoDB** – En dokument basert, open source, såkalt NoSQL-database, skrevet i C++. Det vil si at data ikke lagres i tabeller og rader som i en relasjonell database, som f.eks. MySQL, men i objekter, i MongoDBs tilfelle som BSON-objekter (BSON, binær JSON).

**MQTT** – Message Queuing Telemetry Transport. En åpen nettverksprotokoll for maskin-til-maskin-kommunikasjon med mulighet for å overføre telemetridata, også i nettverk med begrenset funksjon eller høy latens. Protokollen er aktuell for tilkobling av komponenter ved IoT.

**NB-IoT** – Narrowband-IoT. En standard for lavt trådløst strømnettverksradio -teknologi utviklet av 3GPP. En type mobiltelekommunikasjonsstandard.

**STS** – Fra Sensor til Samfunn. Prosjektet Norkart arbeider med.

**UiA** – Universitetet i Agder.

**VScode** – Visual Studio Code. Microsoft sin gratis IDE.

**Webhooks** – En metode for å forsterke eller endre oppførselen til en nettside eller nettapplikasjon med tilpassede tilbakeringinger (engelsk: callback).

**Wifi** – Trådløst nett. Gir trådløs tilgang til Internett i bedrifter, private hjem og offentlige rom.

**4G** – Teknologi for fjerde generasjons mobiltjeneste i mobilnett, etterfølger av 3G og 2G.

**5G** – Teknologi for femte generasjons mobiltjeneste i mobilnett, etterfølger av 4G.

# Figurliste

# 1. Bakgrunn

Prosjektet «Fra Sensor til Samfunn» (heretter omtalt som STS) er et samarbeidsprosjekt mellom flere aktører, gitt i oppdrag fra Kommunesektorens Organisasjon (heretter omtalt som KS) og Kartverket, gjennomført i samarbeid med INTOTO, Bitmesh, Vicotee, Pipelife og Cautus Geo. Prosjektet er utviklet på bakgrunn av «Sensorteknologi i Havn», et studentprosjekt som ble gjennomført våren 2022 ved Universitetet i Agder (heretter UiA) i samarbeid med KS, Oslo Havn, Kystverket og Kartverket. Målet med prosjektet er å bringe frem nåværende byområdeaktiviterer hos ulike byer og kommuner, ulike teknologier, samt tekniske løsninger som kan være med på å digitalisere infrastrukturen i samfunnet. Dette er gjort gjennom kartlegging av status og brukerbehov i norske kommuner. Det har også blitt undersøkt ulike typer teknologier og rammeverk for pilotering av prosjektet.

Den tekniske utviklingen ved piloteringen ble utført av en arbeidsgruppe:

- **Arbeidsgruppen teknisk**; tre sommerstudenter og en ansatt ved Norkart.

Kartlegging av brukerbehov blant kommuner og aktører, består denne arbeidsgruppen av:

- **Arbeidsgruppen kartlegging**; tre sommerstudenter og en ansatt ved Norkart.

## 1.1 Bestilling

Arbeidsgruppen ved prosjektet STS har utviklet en prototype som skal knytte sammen ulike sensordata, og deretter visualisere disse i en eksisterende 3D-modell ved hjelp av JavaScript-biblioteket Cesium. Cesium har blitt benyttet i tidligere prosjekter til utvikling av 3D løsninger i Norkart, og ble et naturlig valg for fremstilling av geografiske 3D-modeller til STS. Formålet med systemet er å samle ulike sensordata på et felles sted hvor data kan hentes ut og visualiseres. Brukergruppen kan også justere ønsket data, noe som er et stort behov blant de ulike kommunene i Norge, da de har ulike behov og ønsker når det kommer til sensorbruk, og befinner seg i ulike stadier ved IoT-satsing. Arbeidsgruppen har også sett på ulike teknologier og eksisterende løsninger, samt undersøkt de mest aktuelle for denne piloteringen.

Arbeidsgruppen skal vurdere nytteverdien av piloteringsprogrammet, beskrive utfordringer og rette fokus mot mulige løsninger. De presenterte teknologiene kan gi mulighet for å fremme digitalisering i kommuners infrastruktur, og vil være et nyttig innsiktsgrunnlag for videre satsing og realisering. Her kan formålet være at det er behov for ytterligere innsiktsarbeid knyttet til fellesløsninger og realiseringer av plattform.

# 2. Eksisterende teknologier

Dette kapittelet tar for seg eksisterende teknologier ved prosjektet. De ulike teknologier og rammeverk som arbeidsgruppene har vurdert, vil her bli beskrevet og presentert. En kort beskrivelse fra funn om teknologiene blant kommunene blir også fremlagt. Denne delen ansees som relevant ved statusen per i dag.

## 2.1 Teknologier og rammeverk

Ettersom arbeidsgruppene skulle dele opp systemet i tre områder, ble det sett på ulike teknologiske løsninger innfor disse. De tre områdene består av input, core og output. Teknologiene blir dermed presentert ved de inndelte områdene.

Input består av MQTT, HTTP Endpoint, API, Data Aggregator, og Webhooks. Arbeidsgruppen har også sett muligheten ved bruk av Edge funksjonalitet dersom brukerne ønsker å ha egne sensorer. Ikke alle kommunene står for egne sensorer, men har ulike sensorleverandører. Med dette er det ikke samme behov for Edge funksjonaliteter. Innen core har arbeidsgruppene sett på lagringsmulighetene i database. Disse består av; MongoDB, InfluxDB og Kubernetes.

MongoDB er en dokumentorientert database med IoT funksjonaliteter. Den kan tilrettelegge store mengder data og gi mulighet for å ha sanntidsanalyse, samt fange opp event-handlinger. Dette er relevant, da hver enkelt kommune har store mengder ulike sensordata. De fleste kommuner har behov for at datastrømmene fungerer i sanntid. Fordeler med MongoDB er at det er kompatibelt med Azure IoT Edge og AWS ++, samt kan knyttes til maskinlæring (heretter forkortet som ML) og gjengir visualisering av grafer.

InfluxDB er en åpen kildekode tidsseriedatabase, som lagrer tidsseriedata knyttet til events og sensorer. Fordelen med dette alternativet, er at den har høy oppdateringsfrekvens slik at store mengder data blir hentet per sekund. Som tidligere nevnt, er dette relevant hos kommunene. Denne typen database kan benytte REST kall. Den gir også valgfrihet i hvilket programmeringsspråk man ønsker å bruke, og det foreligger flere opplæringsmateriale gjennom Influx University. Opplæringsmaterialet gir kommunene mulighet for å lære om hvordan man bruker databasen.

Kubernetes er et system for programvarekontainere basert på åpen kildekode utviklet av Google. Formålet er å sørge for å gi grunnleggende mekanismer i en plattform for automatisk distribusjon, vedlikehold, skalering og operasjoner av kontainere på tvers av klynger av vertsmaskiner. Med andre ord, gir den mulighet til å utvide, samt administrere prosessbelastninger og tjenester.

Arbeidsgruppene har også sett på Amazon Web Services (heretter kalt AWS). AWS er en skyplattform som tillater bygging og drift alle slags applikasjoner. Her er fordelen raskere,

enklere og mer kostnadseffektiv flytting av eksisterende applikasjoner til skyen. På plattformen kan det bygges kompleks infrastruktur, håndtere databehandling og lagring. Det kan også benyttes ML og kunstig intelligens (AI), noe som er relevant for kommunene, da mange ønsker å utarbeide IoT-systemene med dette. Ved prosessering av datainnhenting har arbeidsgruppene sett på Amazon Kinesis Data Streams. Den har funksjonaliteter som gjør at varierte sensordata i all slags skala kan bli hentet og lagret i sanntid. En annen åpen kildekode plattform for event streaming er Apache Kafka. Den er skreddersydd for store datamengder med events. Denne plattformen er kompatibel med ulike teknologier som for eksempel AWS S3 med hensikt om å lagre objekter gjennom et webtjenestegrensesnitt. Apache Kafka har også diverse APIer man kan ta i bruk.

Mulighetene i Microsoft Azure er store. Azure er hovedtjenesten som ble vurdert og er allerede en intern del av Norkarts Azure IoT Hub sikrer kommunikasjon for sending og mottak av data fra IoT enheter. Det er også mulighet for å sende device-to-cloud meldinger og/eller opplasting av datafiler. Den kan sende meldinger fra skyen til enheter, for eksempel ved å skru av/på sensorer. Fordelen kommer frem ved at det kan kobles videre til andre Azure-tjenester for å gjennomføre analyser eller sette opp funksjoner som gjennomføres basert på innhold. IoT Hub har innebygde endpoints. På denne måten sendes data inn og ut fra sensorer eller via andre APIer. Det er også mulig å sette opp egen message-routing videre til andre tjenester, som for eksempel Data Explorer eller Event Hub.

En metode for å effektivisere databaser med mye aktivitet er å bruke databasecluster. Når det ikke er nok å ha kun én database kan et cluster være en god metode. Databasecluster en sammenslutning av flere databaser samlet i én instans. Flere maskiner jobber sammen som motstrider dataoverflødighet, bidrar til bedre skalerbarhet og gjør databasene lettere tilgjengelig under mange transaksjoner.

Kusto Query Language (heretter kalt KQL) er en enkel og lett tilgjengelig metode for håndtering av databaser og visualisering av sensordata. Denne metoden gjør det enklere å se sammenligninger mellom data fra ulike sensorleverandører. Ved bruk av KQL, kan man også laste opp data direkte i databasen gjennom for eksempel CSV-filer, uten å måtte legge til enheter i IoT Hub. Da vil data håndteres i større mengder uten å måtte gå gjennom sensorleverandørenes API.

Azure Event Hub er en god plattform for flyt av Big Data, med mulighet til å motta og prosessere millioner av events eller handlinger i sekundet. Det er sterkt verktøy for å håndtere store mengder sensordata, den oppfører seg leddet for inntak av data og distribuering.

Azure IoT Edge er hensiktsmessig for å kommunisere direkte med enheter og slipper mellomlagring i databaser. Som er aktuelt dersom egne sensorer blir tatt i bruk. Denne teknologien gjør seg mer relevant ettersom det ble oppdaget i brukerkartleggingen at flere av kommunene står for egne sensorer. Sammen med Azure Data Lake, blir store datamengder lagret. Å velge denne er da naturlig når en sensorhub skal tas i bruk i større skala, med mange ulike sensorer og store mengder data. Knyttet sammen med Azure Functions er det

muligheter for å bygge et API og prosessere IoT datastrømmer. Dette gjør det enklere å sette funksjonene i fokus i stedet for den overordene strukturen.

Aktuelle outputmetoder var hovedsakelig webhooks, WebSockets og REST API. I et system som skal videresende lagret sensordata er det essensielt med en outputmetode. Den mest fleksible prosedyren vil være et API. Det er mange måter å lage et API på, men gjennom Azure Functions blir prosessen lett og effektiv. Dersom dette knyttes med webhooks eller WebSockets, kan det gi mulighet til å sende sanntidsoppdateringer på data som kommer inn i systemet og gi varslinger i output-metoden.

## 2.2 Teknologier hos kommunene

Teknologiene blant kommunene hadde store likheter. Kommunene hadde vært innom LoRaWAN og Nb-IoT, og har flyttet seg gjennom flere generasjons mobiltjenester i mobilnett gjennom årene. Status rundt dette i dag, er at de fleste ønsker å flytte seg fra 4G til 5G. Kommunene vil da ha høyere datahastighet, og lav forsinkelse på mindre enn 1ms i det mobile bakkenettet. Det er viktig å nevne at 5G er basert på User Data Protocol (UDP) / IP-nettverk (IP), og bruker New Radio (NR) som teknologi. Denne typen teknologi er aktuell med tanke på at kommunene ønsker raskere sanntidsdata med så lite så mulig forsinkelse. Det ble nevnt under dybdeintervjuene at overføringer av data også går over både trådløst og kablet nettverk, avhengig av mulighetene på lokasjon. Det ble nevnt under dybdeintervjuene at overføringer av data går over både trådløst og kablet nettverk, avhengig av lokasjon.

Utfordringene rundt sensorleverandører og kommunenes data var størst på datasett og format, da de ulike formatene skaper utfordringer når det kommer til datasettene. Dette gjelder under innhentingsprosessen, men også egne systemer som skal samkjøre med andre ulike sensordata. Dette har noe med at det foreligger ustandardisert data i Norge.

# 3. Prosess i innledende fase

For å få oversikt over dagens situasjon innen sensorteknologier, har det blitt utført kartlegging blant kommuner, aktører og sensorleverandører. Innledende undersøkelser, dybdeintervjuer og digitale dialoger med både kommuner, aktører og sensorleverandører har gitt et bredt innsyn i dagens status. Videre problemstillinger går ut på bruk av sensordata i Norges kommuner, hvilke funksjonaliteter som er ønsket, hva det er størst behov for, samt samarbeidsvillighet mellom kommuner og aktører. Blant leverandører har det gitt innsikt på hvilke typer sensordata som distribueres.

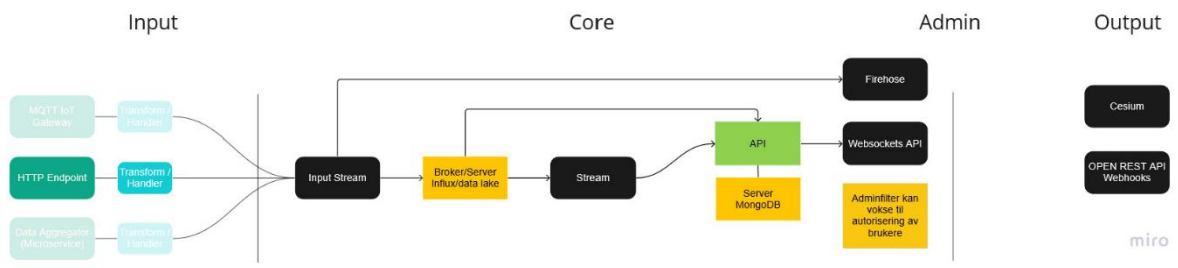
Deltagerne hadde varierende kjennskap til sensorbruk i sin kommune, hvorav noen hadde overblikk over kommunen og distriktet helhetlig, mens andre arbeidet kun med sensorer innenfor sin avdeling eller fagområde. De ulike kompetanseområdene blant deltagerne ga god variasjon i tilbakemeldingene som ble innhentet.

## 3.1 Workshop og brainstorming

Det ble utført en to dagers intensiv med workshop og brainstorming sammen med to av prosjektets samarbeidspartnere, INTOTO og Bitmesh. Dette ble gjort for å se nærmere på hvordan man går frem med å lage en sensorhub på best mulig måte med de ønskede funksjonalitetene av ulik sensordata. Brainstormingen ga arbeidsgruppene et godt overblikk over ulike teknologier som kunne være aktuelle for prosjektet. Utvalget var stort, hvorav flere alternativer blir presentert i **2.1 Teknologier** og rammeverk. Arbeidsgruppen har sett på de ulike delene av innhenting, lagring, prosessering og videresending av sensordata.

## 3.2 Valg av teknologier

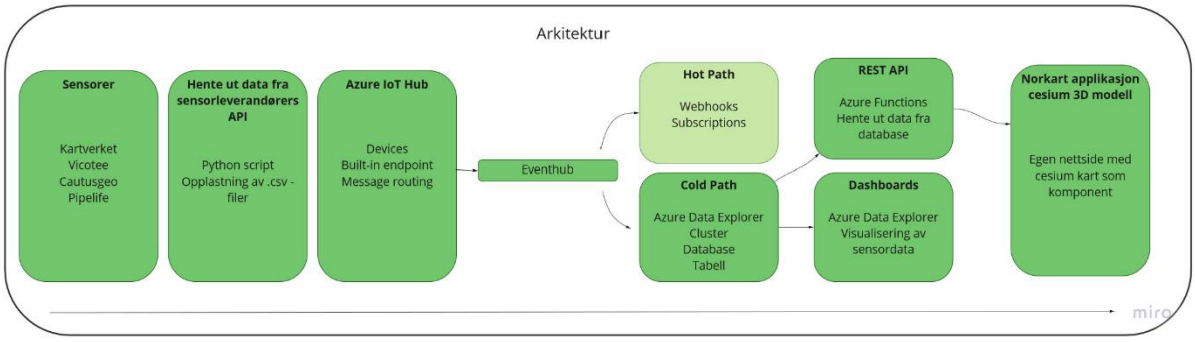
Prosjektet startet med en workshop hvor flere mulige teknologier ble diskutert i henhold til systemarkitekturen. Basert på diskusjoner ble det fastslått at systemet skulle være i tre deler; input, core, og output. Innenfor kategoriene var det mange etablerte teknologier som alle hadde sine egne fordeler og ulemper. Begrensninger med hensyn til tid og andre ressurser påvirket valget av disse teknologiene.




*Figur 1: Første utkast til teknisk arkitektur (v1).*

Workshop, undersøkelser og diskusjoner bistod i å komme frem til de beste valgene basert på kompetansene i arbeidsgruppene, men også i forhold til kompetanser hos veiledere og ressurspersoner.

Første del av arkitekturen bestod av inputmetoder. Siden prosjektet baserer seg på sensordata og IoT, var HTTP og MQTT de mest åpenbare protokollene å bruke. I startfasen ble det også foreslått en tredje metode med Data Aggregator Microservices. Det ble bestemt at fokuset skulle kun ligge på HTTP Endpoints ettersom data fra sensorleverandørene var hovedsakelig historiske. Dermed kunne dataen bli hentet ut gjennom eksisterende API.




*Figur 2: Endelig design av systemarkitektur i piloteringsprosjektet.*

Arkitekturen måtte gjennom flere iterasjoner, som ble tydelig i kjernedelen av systemet. Komponenten som skulle ta imot data ble satt opp som en "Input Stream", hvor teknologier som Kinsesis, Azure Event Hub og datasjøer ble foreslått. Prosjektet landet på Azure som kjerneteknologi hvor dens funksjoner passet perfekt til prosjektets formål. Akkurat hvordan Azure operer i systemet blir utvidet i **4.3 Microsoft Azure**. Videre måtte det være en intern database i systemet som kunne lagre all sensordataen som ble mottatt. Det var snakk om å ha en Broker som kunne lyttes på når data kom inn, det ble senere endret til Azure Event Hub, se **Figur 2.** Valg av database var ikke enkelt ettersom det er et mangfold av gode løsninger der ute. Typer som SQL-databaser, MongoDB, InfluxDB osv. Teamet fant fort ut at store datamengder betydde at en tidsserie-/dokumentdatabase var nødvendig. Ettersom det allerede var bestemt at fokuset lå på Azure ble det, etter undersøkelser ble det oppdaget at den har en funksjon kalt Kusto Query Language (kjent som KQL). Kustodatabasen er en intern database i Azure som tillater god håndtering og manipulering av data.

Fra databasen måtte systemet ha en metode for å hente ut og ta i bruk all dataen, dette ble gjort gjennom et API. Azure hadde også en funksjon for dette gjennom Azure API. I APIet kan data fra hver sensorleverandør bli hentet ut gjennom HTTP. APIet ligger i Azure som en Azure Function og er skrevet i Python. Det er også tenkt at systemet skal ha muligheter for å bruke WebSockets og webhooks, men dette var ikke hovedfokus for piloteringen og dermed ikke

implementert. Som en del av prosjektet ble det også lagd en nettside som bruker APIet for å visualisere output data i CesiumJS.

## 3.3 Avgrensninger

Det kan være utfordrende for arbeidsgruppen å navigere seg blant alle teknologiene og rammeverkene som er tilgjengelig. Mye har blitt utprøvd, mens annet foreligger det lite kunnskap om tilgjengelig på nett. Arbeidsgruppene så dermed på funksjonaliteter som kunne dekke kommunenes behov. Noen av de nevnte teknologiene krevde tilgang, noe som gav arbeidsgruppen en del avgrensninger under prosjektet. Dette diskuteres henholdsvis ved **4.5 Utfordringer**. Drømmen om et større felles sensorhub på nasjonalt nivå bør selvfølgelig utredes, men i den mindre skalaen og med tanke på at prosjektet befinner seg i en piloteringsfase, var det viktig for arbeidsgruppene å innhente de ulike sensordataene som kunne benyttes. Det var viktig å knytte disse til en plattform og få dataen til å fungere. Videre fra dette, utarbeide disse for en visuell fremstilling.

# 4. Utvikling

Dette kapittelet tar for seg utviklingsaspektene ved prosjektet relatert til det grunnleggende konseptet og teknologiene for systemet. Dokumentasjon på systemets funksjoner via Azure og Cesium og hvilken rolle de spiller prosjektet.

## 4.1 Konsept

Systemet går ut på å ta mot sensordata fra diverse aktører, standardisere disse og lage et API som kaller på data for å fremstille dem visuelt i en applikasjon. Data blir hentet inn og lagret i en intern database via Azure, og blir formatert til en strukturert JSON. Deretter har teamet skapt et eget API som tillater henting av disse historiske dataene. Funksjon for sanntidsdata er ikke implementert. Ved å ta i bruk dette grensesnittet ble det skapt en web-applikasjon som fremstiller sensordata for innendørstemperatur, vanntemperatur og vannstand. Visualiseringen ble skapt ved hjelp av JavaScript-biblioteket CesiumJS. Her hentet vi sensordata fra APIet og festet det til diverse 3D modellerte bygg, slik at de endrer farge basert på temperaturmålingene.

## 4.2 Design

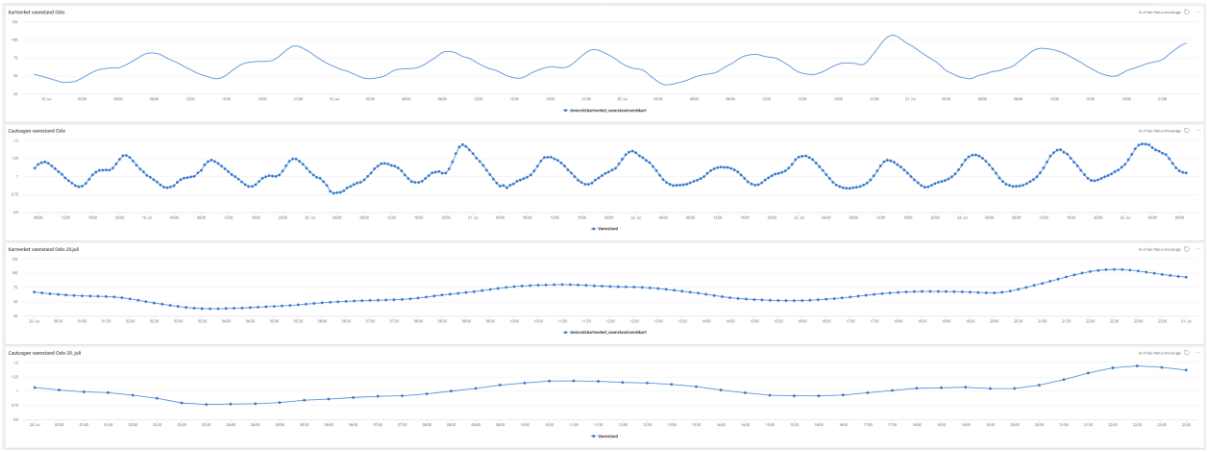
Design av arkitektur startet med en to dagers workshop sammen med representanter fra INTOTO og Bitmesh. Det var mange forslag til potensielle metoder og teknologier for å implementere en arkitektur i henhold til IoT og sensorer. Som resultat landet det på Microsoft Azure som kjerneteknologi. Kapittelet **3.2 Valg av teknologier** inneholder design av arkitektur og dypere beskrivelse av valgene fra designprosessen.

## 4.3 Microsoft Azure

Hovedsystemet er implementert i Microsoft Azure, som gir en rekke med funksjonaliteter relevant til prosjektet for å håndtere IoT og sensordata. Under kommer en detaljert dokumentasjon for hvordan prosjektets Azure system fungerer.

Teamet begynte med å lage en Azure IoT Hub som kunne registrere en *device per sensor* for å koble seg til Azure systemet. Teamet hadde ikke tilgang til de fysiske sensorene og tilhørende førsteparts programvare, av den grunn ble alternative metoder undersøkt. Løsningen var å lage en *virtual device* i IoT Hubben for å ta imot data og bruke som en kanal til selve hubben. Teamet lagde skript for å hente ut data fra sensorleverandørenes eksisterende APIer. For å hente data fra sensorleverandøren Vicotee, trenger skriptet en *fersk API* nøkkel som blir autorisert av et teammedlem sin brukerkonto. API-nøklene har kun

gyldighet i et par timer, slik at det må hentes ut en ny for hvert API kall. Vicotee sin data var formatert i et ryddig JSON-format, hvor det ikke var behov for justeringer i formatet. Derimot, kom data fra Kartverket i et suboptimalt XML-format som vi konverterte til et strukturert og gyldig JSON-format. Cautus Geo og Pipelife sin data ble lastet ned i CSV-format og manuelt opplastet til databasen i Azure.



*Figur 3: Sammenligning av vannstandsmålinger fra Kartverket og Cautus Geo, visualisert i Azure Data Explorer.*

For å få god oversikt og mulighet til konfigurasjon av Azure IoT Hub, ble det brukt en utvidelse i Visual Studio Code. Det beste med denne framgangsmåten var at utvidelsen kan generere et eksempelprogram som sender inn testdata til IoT Hubben. Koden er fleksibel og kan genereres på diverse programmeringsspråk. Når koden er generert, er det enkelt å modifisere den for å møte spesifikke formål. I prosjektet ble koden konfigurert for å videresende den innhentede dataen.

Azure har en annen funksjon kalt Event Hub. Denne funksjonen kan prosessere og videresende store mengder data. Dette ble brukt som en kobling mellom IoT Hub og databasen. Når data befinner seg i IoT Hubben blir den plukket opp av Event Hubben og sendt til databasen. I dette stadiet er det muligheter for å filtrere og sende ut datamålinger i sanntid, se "Hot Path" i **Figur 2**.

Systemets database er satt opp i Azure Data Explorer. For at data fra Event Hub skal bli satt inn i databasen, må du først definere hvilken tabell og kolonne det har som destinasjon i en *mapping*. Det oppstod et problem med denne framgangsmåten ved at muligheten for å skille mellom datakilder forsvinner. Grunnet all data ble samlet da det var sendt fra separate IoT enheter til Event Hub. Løsningen til problemet lå i beskjeden fra enhet til IoT Hub. I beskjeden er det mulig å sende med metadata som inneholder hvilken *mapping* eller kartlegging det skal bruke når beskjeden ankommer databasen.

Azure Data Explorer har en visualiseringsløsning som heter Insights. Denne funksjonen ble brukt til å lage diverse grafer som visualiserer sensordata fra databasen.

For å sende data ut av systemet ble det bygget et REST API. Det er skrevet i Python ved hjelp av FastAPI og lastet opp i skyen som en Azure Function. For å hente ut data via APIet tar det imot to parametere: *'table'* og *'query'.* I mappen API-endpointet ligger i, ligger det også forskjellige konfigurasjonsfiler for hver tabell i databasen. For at APIet skal bruke den riktige konfigurasjonen må det defineres av brukeren som kaller APIet. Dette blir gjort ved å skrive navnet på konfigurasjonsfilen i *table*-parameteren. Eksempelvis "table=Vicotee_config.json". *Query*-parameteren er valgfritt. Hvis *query* ikke er definert, henter APIet ut all dataen som ligger i tabellen konfigurert i *table*-parameteren. Under er det eksempelkode som henter ut sensormålinger for temperatur og lys fra Vicotee den 19. og 20. Juli sortert etter dato:

```
http://sts2022-
api.azurewebsites.net/api/securehttp?table=Vicotee_config.json&query=|
where (key == "voc_temperature" or key =="ambient_light") and (timestamp
startswith "2022-07-19" or timestamp startswith "2022-07-20") | sort by
timestamp asc
```

*Figur 4: Eksempel på HTTP-kall fra API. Meldingen henter ut temperatur- og lysmålinger fra Vicotee den 19. og 20. Juli sortert etter dato.*

For at APIet skal ha tilgang til databasen krever det en autentisering. Det er mange metoder for å oppnå riktig autentisering. For testing ble den enkleste metoden benyttet, der brukeren får opp et vindu i nettleseren og må logge på Azure Portal. For at denne metoden skal fungere kreves det at brukeren som kaller APIet har tilgang til databasen. Denne metoden fungerer derimot dårlig når brukeren er et program, for eksempel uthenting i JavaScript. Teamet bestemte da at autentiseringen blir gjort hos APIet i stedet for hos brukeren. Løsningen var en annen Azure funksjonalitet kalt Azure Active Directory, hvor det er satt opp en ID som kobler APIet til databasen gjennom en API-nøkkel. På denne måten kreves ingen konfigurering for autentisering av brukeren. Dette er en midlertidig løsning som ikke legger stor vekt på sikkerhet, men gjør livet enkelt for utviklere. Videre i prosjektet vil en løsning for å forbedre denne autentiseringsmetoden være å la autentisering skje gjennom *bearer tokens.*

Når APIet henter ut data fra databasen inneholder det opprinnelig kun verdiene i en array, men ingen kolonnenavn. Kolonnenavnene henter APIet fra konfigurasjonsfilen til tabellen. Deretter går den gjennom en loop av databasens output for å generere et JSON-objekt med kolonnenavn i form av *keys. Keys* er satt sammen med korresponderende *values* fra databasen. Dette objektet blir sendt som en *return* til brukeren.

## 4.4 Cesium

Web-applikasjonen som visualiserer dataen, er skapt i React med CesiumJS. I React er det lagd en komponent som henter inn dataen med et REST kall til Azure API funksjonen. Den henter data fra databasen i Azure. Dataen er i et 24 timers tidsrom og blir separert i enkelttimer. Komponenten er sammenslått med en skyveknapp som har 24 markeringer. Når skyveknappen endrer markering i form av en time vil verdien bli sendt til en funksjon som bestemmer hvilken timesdata som skal brukes. Deretter videresendes det til Cesium for å oppdatere fargene. Cesium består av en modell med 3D bygg (tileset). Når Cesium mottar temperaturdata, vil et skript konvertere det til et RGB-fargeformat, og kalle en ny komponent som oppdaterer tileset og de nye fargene blir lastet inn. Tileset har en rekke funksjoner man kan benytte for å endre på modellene, som bygningskategorier, farge, skygge osv. Spesifikt er det brukt Cesium3DTileStyle for å endre fargene på byggene. TileStyle tar inn et tileset som parameter og ved å bruke style funksjonen vil et JSON-format tillate bygningsspesifikke endringer. Fargen kan endres med for eksempel RGB eller Hex-verdier, som er grunnen til at dataene blir konvertert til et RGB-format Cesium kan motta i tileset. Data for vannstand og vanntemperatur blir hentet fra samme Azure database og verdiene blir sendt til Cesium som visualiserer det som en LabelEntity, i form av en tekstboks/merkelapp. Her blir dataen kun konvertert til en tekststreng.


*Figur 5: Slider styrer tidsrom for temperatur- og vannmålinger visualisert på bygg og hav.*

## 4.5 Utfordringer

Utfordringene i prosjektet bestod hovedsakelig av lite tilgjengelig dokumentasjon på programvare og bibliotek. Under prosjektets oppstart hadde arbeidsgruppen minimalt med kunnskap til Microsoft Azure, hvor det var vanskelig å sette seg inn i økosystemet og finne relevant informasjon om fremgangsmåte, da vi hadde lite erfaring med cloud-computing. Dette forårsaket at mye tid ble brukt i prosjektets oppstart med opplæring i programmet og inkluderte funksjonaliteter. Lignende utfordring oppstod i Cesium, som er et bibliotek svært lite utnyttet på global skala og inneholder svak dokumentering av diverse funksjoner og problemstillinger. Problemet førte til at enkle implementeringer og idéer kunne gå langt over estimert tid.

Et kjent problem i sensorverdenen er mangel på standardisering. Dette viste seg i prosjektet da sensorata skulle inn i systemet. Utfordringen ved å dele data mellom aktører er grunnet de fleste har en intern metode for håndtering av sensordata, som andre ikke er kjent med eller ikke har mulighet til å motta. For eksempel kan noen returnere en JSON-fil som er formatert på en sær måte, eller i et helt annet filformat som få har hørt om, og enda færre faktisk bruker. Lite dataflyt mellom partene kan føre til at flere sitter på verdifulle data andre hadde dratt nytte av. Delingen er også hindret av manglende dokumentasjon på eksisterende APIer, noe som skapte frustrasjon og problemer i prosjektet da data skulle først hentes ut. Et formål med dette prosjektet var å se på denne problemstillingen og finne eventuelle metoder for å standardisere disse sensordataene, utdypet **5.2.1 Backend og teknisk arbeid.**

# 5. Veien videre

Pilotprosjektet for STS har stort potensiale når det kommer til videreutvikling. Videreutvikling av prototype fra pilotprosjektet kan deles inn i forskjellige fokusområder, hvor tre viktige områder er kort beskrevet under i **5.2.1 Backend og teknisk arbeid**, **5.2.2 Frontend og Cesium**, og **5.2.3 UX- og grafisk design**. Funksjonalitetene som integreres i piloteringen er viktige for å undersøke mulighetene, men i neste steg anbefales det også å rette mer fokus mot brukeren og deres behov for design av en prototype som kan benyttes i ulike test scenarioer.

## 5.1 Vurdering

Demonstrasjon av prototype til samarbeidspartnere vil bli et nyttig ledd for vurderingen. De er å anse som eksperter innen feltet, og kan bidra med gode tilbakemeldinger og forslag for videre fokusområder for piloteringen.

Teste og evaluere prototypen som har blitt utviklet er et viktig steg for å få god forståelse for videre valg av fokusområder. Med hensyn på dette vil neste steg i utvikling innebære å tilpasse prototypen slik at den kan blir enklere og mer praktisk å ta i bruk for testing, samt design av scenarioer som er ønskelig å teste.

## 5.2 Videre arbeid

Videre arbeid for prosjektet kan deles inn i flere faser som til dels kan samkjøres under videre utvikling. Det er stort potensiale for prototypen når det kommer til å være et verktøy som kan forenkle enkelte arbeidsprosesser når det kommer til IoT og sensorteknologi, men her må funksjonalitet kombineres med brukbarhet.

### 5.2.1 Backend og teknisk arbeid

En viktig problemstilling diskutert i startfasen av prosjektet var å få en universal metode for å standardisere sensordata som kommer inn i systemet. Slik at systemet tillater målinger fra alle typer brukere. Piloteringsprosjektet hadde kun fire forskjellige kilder til sensorata, som ikke representerer alle de diverse brukerne for systemet. Videre arbeid innebærer å se på en løsning for mottak av flere kildetyper. Eventuelle metoder innebærer bruk av AI, nye skript, andre teknologier etc.

Det er flere forbedringer som må til i Azure for å sikre skalerbarhet videre i prosjektet. Utvidelse av inntak for sensorkilder medfører at kjernedelen av systemet også må bli

tilrettelagt nye formater ved å legge til ny kartlegging for inntakskilden. I databasen kan det implementeres metoder for automatisk opprettelse av tabeller for sensordataen.

Ideelt sett så burde sensorer kunne kobles direkte til ett Norkart program på en brukervennlig måte. Etter dette kunne programvaren satt opp ny device på IoT hubben og koblet sensoren direkte til den. Ny tabell og mapping i databasen kan bli automatisk generert, samt viderekobling med exit API-et for cold storage. En brukervennlig meny kan tenkes som genererer hot paths med subscription muligheter for spesifikke datastrømmer som utfyller brukersatte kriterier.

Med tanke på fremtidig bruk hos kommuner, bør det også tilrettelegges for å kunne bruke egne sensorer og at brukeren selv kan velge hvem som kan se eller endre sensorer og datastrømmer i deres område.

### 5.2.2 Frontend og Cesium

Gjennom pilotprosjektet ble det vektlagt å få et funksjonelt konseptbevis i form av en prototype. I forbindelse med videre utvikling av pilotprosjektet anbefales det at Cesium blir mer vektlagt, og tilgjengelige funksjonaliteter som kan løfte produktet blir utnyttet slik at funksjonsverdien til prototypen blir økt i takt med økt kompleksitet i backend utviklingen av prototypen. Ved å legge inn visualisering av flere tilgjengelige datasett vil konseptet blir tydeligere fremstilt. Per nå er det ikke en tilgjengelig metode i Cesium for visualisering av flo og fjære da terreng modellering ikke er tilgjengelig under havnivå. Alternative visualiseringsmetoder kan testes for å få dette også visuelt i modellen, da prototypen fra pilotprosjektet kun viser tallverdier fra målingene her.

Pilotprosjektet fikk også tilgang til flere målinger som det ikke var tilgjengelig tid for å bearbeide over prosjektperioden. Å inkludere disse målingene, flere temperatur-, lys-, og luftmålinger for å nevne noen, kan bistå til å fremstille et mer helhetlig bilde av potensialet for å samle forskjellige formater og typer sensordata i en visualisert applikasjon.

### 5.2.3 UX- og grafisk design

Ved videre utvikling fra pilotering bør et større fokus rettes mot brukere av løsningen. Gjennom piloteringen ble det rettet lite fokus mot brukeropplevelsen, men i neste fase av utvikling bør brukertester og design med tanke på ulike brukere vektlegges i større grad.

Det er flere områder som kan arbeides med for å sikre en god brukeropplevelse. Blant de mer åpenbare områdene er det å gi brukeren valgmuligheter, som eksempelvis å filtrere hvilke data som er ønskelig å se til enhver tid. Et oversiktlig brukergrensesnitt som er enkelt å ta i bruk, men som også dekker brukerens grunnleggende behov under interaksjon med

systemet, er viktig for den helhetlige opplevelsen. Her vil en naturlig start være oppsett av wireframes for ulike scenarioer basert på personas.

Gjennom dybdeintervjuene som ble gjennomført under brukerkartleggingen ble mye varierende informasjon innsamlet om aktuelle brukere av sensorteknologi i de ulike arbeidsleddene hos kommunene. Å benytte denne informasjonen til å utvikle personas er anbefalt, men også er relativt enkel oppgave som vil ha mye å si for UX perspektivet for videre utvikling av pilotprosjektet. Dette vil legge grunnlag for storyboard design, valg av datafremstillinger, layout til grensesnittet, bruk av design elementer, for å nevne et par eksempler.