Konvergens mellom IT & OT

Et samarbeid med Knowit Sør AS

Torkel Herskedal Ivarsøy
Asbjørn Håland Hestnes

VEILEDER
Geir Inge Hausvik

Universitetet i Agder, 2024
Fakultet for samfunnsvitenskap
Institutt for informasjonssystemer

Forside
IS-304: 2024
Tittel: Konvergens mellom IT & OT

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig:	Hallgeir Nilsen & Geir Inge Hausvik
Veileder	Geir Inge Hausvik
Oppdragsgiver:	Knowit

Studenter:

Etternavn	Fornavn
Herskedal Ivarsøy	Torkel
Håland Hestnes	Asbjørn

	JA	NEI
Jeg/vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	✔	
Kan besvarelsen brukes til undervisningsformål?	✔	
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	✔	

Forord

Denne bacheloroppgaven markerer en milepæl av vår akademiske reise ved Universitetet i Agder, en periode som har blitt fylt med lærerike utfordringer, personlig vekst og faglig utviklingen. Arbeidet med prosjektet Mini Smart Factory Lab i samarbeid med Knowit Sør AS har vært en unik og berikende opplevelse som har gitt oss muligheten til å utforske skjæringspunktet mellom operasjonell teknologi (OT) og informasjonsteknologi (IT) i en industriell kontekst.

Vi ønsker å gi vår dypeste takk til vår veileder Geir Inge Hausvik for hans veiledning, støtte og innsikt gjennom hele prosjektperioden. Dette har vært avgjørende for å få til dette prosjektet gjennom oppmuntring og ekspertise.

En spesiell takk går også til ansatte i Knowit Sør AS, spesielt Arnt Aske, Kristoffer Sand og Trygve Solberg, Parisa Karampour, Erik Mashmann for deres samarbeid, tilgjengelighet og villighet til å dele kunnskap og erfaringer. Dette samarbeidet har vært med på å berike forståelsen av smarte fabrikkløsninger, samtidig understrekt betydningen av industrielt samarbeid i teknologisk innovasjon. Vi er også takknemlig for støtten fra medstudenter, familie og venner, som har vært med på å gi oss styrke, motivasjon og oppmuntret oss gjennom studiet vårt. Troen de har hatt på evnen våres har vært en viktig kilde til motivasjon.

Til slutt, dedikerer vi denne oppgaven til alle som er interessert i å bringe teknologisk innovasjon til industriell produksjon. Det er vårt håp at funnene og anbefalingene presentert i dette arbeide kan bidra til fremtidig forskning og utvikling innen smart fabrikkteknologi.

Abstrakt

Denne rapporten utforsker bachelorprosjektet vårt i samarbeid med Knowit Sør AS. Denne tar stilling til implementeringen og utviklingen av Mini Smart Factory Lab, et innovativt system som er designet for å demonstrere hvordan produksjonsprosesser kan bli automatisert og optimalisert, samtidig som kvalitet blir forbedret.

Utviklingen av Mini Smart Factory Lab demonstrerer hvordan ved å automatisere manuelle arbeidsoppgaver kan skape forbedringer i produksjonseffektivitet og produktkvalitet samtidig som en får datainnsikt i produksjonsprosessene på en konstandseffektiv måte. Prosjektet viser potensialet for IoT, moderne IT- og OT systemer i industrielle sammenhenger, og hvordan en mer datadrevet tilnærming kan transformere produksjonsindustrien.

Dette prosjektet legger grunnlaget for løsninger Knowit kan bruke i sammenheng til eksisterende og nye kunder for å forbedre operasjonell effektivitet, ressurshåndtering og produktkvalitet. Gjennom dette prosjektet bidrar vi med praktiske anbefalinger for integrering av smarte fabrikkløsninger, fremmer innovasjon og teknologisk fremgang innen industriell produksjon.

Gjennomgang av prosjektet kan ses her

Innhold

Figurliste

Tabbeliste

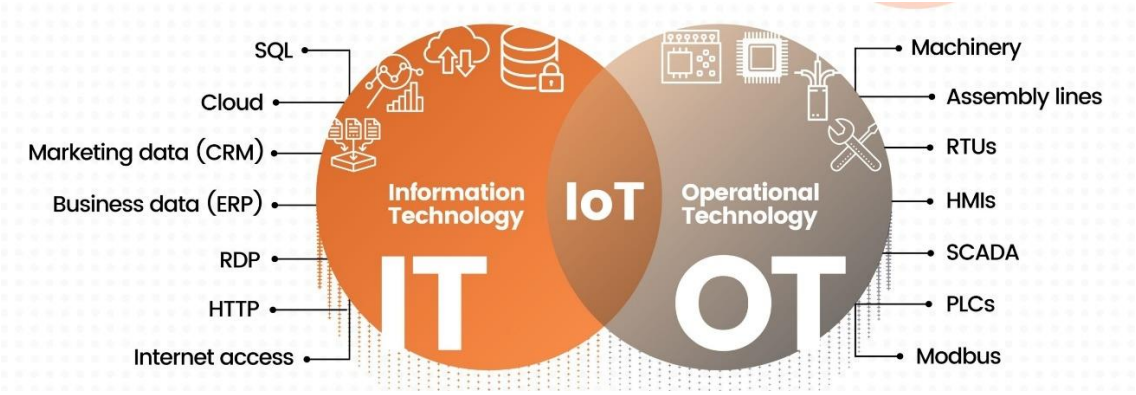
1. Introduksjon

Tradisjonelt sett har det vært slik at informasjonsteknologi (IT) og operasjonell teknologi (OT) har eksistert i to forskjellige domener, som utfører hver sine funksjoner. Men nå er en transformasjon mellom disse verdenene på vei. Derfor er det blitt tydelig at fremtidens industrielle landskap vil bli formet av en konvergering av OT- og IT systemer som er knyttet til industri 4.0. Implementeringen krever ikke bare teknologisk integrasjon, men også en strategisk tilnærming (What Is IT/OT Convergence?, u.å.).

Agder-regionen har en rik historie og et sterkt industrielt fundament og står nå ved et veiskille i den teknologiske revolusjonen. Sysselsettingen i regionen domineres av bygg- og anleggsbransjen og mekanisk industri. Dette spekteret av industrielle aktiviteter, fra informasjons- og kommunikasjonsteknologi (IKT) og elektronikkbransjen til prosessindustrien og leverandørindustrien for olje- og gassvirksomhet, utgjør kjernen i Agders økonomi (Forskningsrådet, 2022). Denne posisjonen som Agder har, gjør regionen til et ideelt utgangspunkt for å utforske og utnytte potensialet som IT og OT konvergensen skaper.

Som et direkte svar på denne konvergensen har vi utviklet Mini Smart Factory Lab. Et prosjekt som er designet for å demonstrere potensialet og hvordan smart fabrikkteknologi blir anvendt i en praktisk setting. Systemet er utviklet for å simulere og demonstrere hvordan konvergensen mellom IT og OT muliggjøre automatiserte produksjonsprosesser ved hjelp av IoT-enheter, se Figur 1, mikrokontrollere, sanntidssystemer for datahåndtering, og algoritmer som styrer produksjonsprosessene. Prosjektet viser hvordan det er mulig å transformere operasjonen av fremtidige fabrikker. Med et slikt system kan fabrikkene operere med større presisjon, mindre ressursbruk og optimalisert produksjonseffektivitet.

Formålet med denne rapporten er å utforske de tekniske aspektene med å implementere informasjons- og operasjonellteknologi i en industriell kontekst. Vi vil diskutere løsninger, utfordringer og fordeler ved bruk av ulik teknologi i dagens industrielle landskap direkte ut mot en kunde som kan dra nytte av denne konvergensen.



Figur 1 - Konvergens mellom IT og OT. Tanand Technology, 2023

Hentet fra https://www.tanand.com.my/it-ot-convergence-how-does-it-benefits-digital-manufacturing/

1.1 Knowit og industrielt samarbeid

Knowit som konsulentselskap spiller en viktig rolle i den digitale transformasjonen av næringslivet. Selskapet, som ble etablert i 1990, har utviklet seg til å bli et ledene selskap for fremtidens digitale løsninger. Selskapet har vokst til å bli ledene i Norden, med over 4000 ansatte fordelt på kontorer i Norge, Sverige, Danmark, Finland, Tyskland og Polen. De har ekspertise innen IT, design, kommunikasjon og management (Om Knowit | Digitaliseringskonsulenter, u.å.).

I en industriell kontekst bidrar selskapet ved å digitalisere og sikre utvikling og integrasjon av programvareteknologi, noe som spesielt er viktig for fremtidige bedrifter om å flytte dem i en bærekraftig retning. Dette fremmer de ved å ha høy grad av kompetanse innen kunstig intelligens (KI), Internet of Things (IoT) og integrerte applikasjoner. De bruker også sin erfaring med automatisering og skyløsninger for å skape en mer sammenkoblet verden (Om Knowit | Digitaliseringskonsulenter, u.å.).

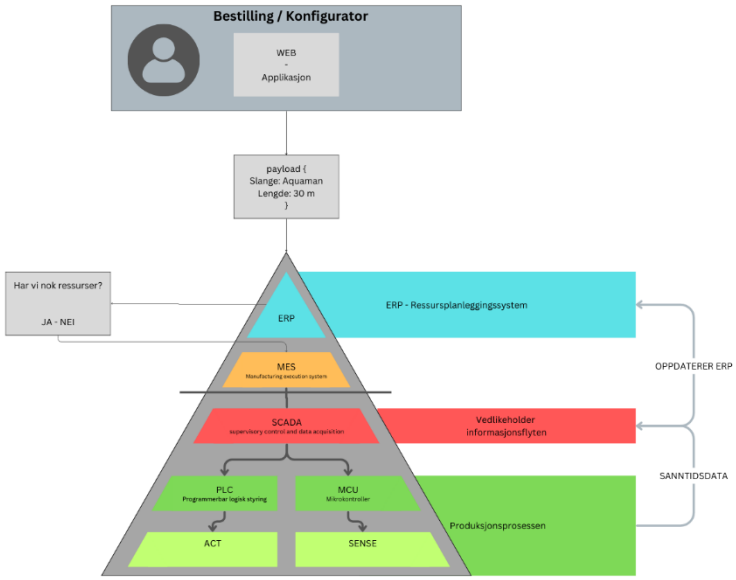
1.2 Om prosjektet: Mini Smart Factory Lab

Mini Smart Factory Lab prosjektet, er en satsning som er designet for å demonstrere fremtidens produksjonsmetoder gjennom automatisering og digitalisering. Dette prosjektet utnytter moderne teknologi, som mikrokontrollere med WiFi-tilkobling og sanntidssystemer for datahåndtering, for å skape sømløse og effektive produksjonsprosesser.

For å gi en klar forståelse av prosjektet sin anvendelse i praksis, er det viktig å inkludere en beskrivelse av kunden Mandals AS, som er samarbeidspartneren i dette prosjektet. Mandals AS er en ledende produsent av flatliggende slanger av høy kvalitet. Samarbeidet ser på hvordan de kan automatisere og effektivisere manuelle produksjonsprosesser ved bruk av moderne teknologi. Gjennom dette prosjektet har vi anvendt teoretisk kunnskap i en konkret industriell setting, som gir verdifull innsikt i de utfordringene og mulighetene som eksisterer ved smarte fabrikkløsninger. Videre adresserer prosjektet hvordan datainnsikt i produksjonsprosesser kan oppnås ved integrasjon av IoT-enheter som et alternativ til utskiftning av industrielle maskiner.

Som vist i Figur 2 starter prosessen i Mini Smart Factory Lab med en bestilling fra en kunde gjennom et brukergrensesnitt. Selve bestillingen som definerer spesifikke produksjonsparametere som type og lengde på slangen sendes så videre til ERP-systemet som i vårt tilfelle er en lokal database. ERP-systemet sjekker om de nødvendige ressursene er tilgjengelige for å oppfylle bestillingen. Dersom ressursene er tilgjengelige går prosessen videre til Manufacturing Execution system (MES) og SCADA (Supervisory Control And Data Acquisition) som er sentralsystemet og starter produksjonsprosessene. Informasjonsflyten sikrer en strømlinjet håndtering og optimalisering av produksjonen samtidig som det lagrer sanntidsdata for retrospektivt arbeid og oppdaterer ERP-systemet for å beholde dataintegritet.

Prosjektet demonstrere hvordan implementering av IoT-enheter i en industriell kontekst kan gi økt innsikt i produksjonsprosessen, øke effektiviteten, redusere material svinn og øke kundeverdi.



Figur 2 - Automasjonstrekant

1.3 Teamet bak Mini Smart Factory Lab

Prosjektet er et samarbeid mellom studenter, akademikere og eksperter innen industri og kunstig intelligens, som alle har et felles mål om å bringe innovasjon til produksjonsindustrien gjennom smarte fabrikkløsninger. Videre presenteres bakgrunn og rollene de har hatt i prosjektet.

Arnt Aske: Leder for Smart Factory i Knowit Sør. Har over 25 års internasjonal erfaring ved å lede teknisk drevne organisasjoner til kommersiell suksess. Av den grunn er Arnt prosjektleder og har vært en viktig ressurs for å styre prosjektet i riktig retning.

Torkel Herskedal Ivarsøy og Asbjørn Håland Hestnes: Som bachelorstudenter ved Universitet i Agder, utgjør vi kjernen i utviklingsteamet. Begge har en lidenskap for teknologi og innovasjon, hvor vi tar oss av analyse, design, utvikling og testing av Mini Smart Factory Lab.

Kristoffer Sand: Ansatt i Knowit og jobber for avdelingen Smart Factory. Han har bakgrunn i konsulentbransjen og er utdannet sivilingeniør med hovedtyngde i mekatronikk. Kristoffer fungerer som produkteier i prosjektet. Hans ekspertise innen mekatronikk er uvurderlig i prosjektet.

Trygve Solberg: Har også en master innen kunstig intelligens og har god erfaring med programvareutvikling. Trygve deler roller som produkteier med Kristoffer, og sammen sikrer de at prosjektet møter høye standarder.

Parisa Karampour: Har tidligere jobbet som data scientist og har kommet inn på avdelingen for å kartlegge CI/CD, og skal kartlegge hvordan vår løsning alltid er klar for implementering. Dette er essensielt for effektiv utvikling og vedlikehold. Hennes deltakelse har ikke hatt direkte påvirkning på systemet, men vil ha stor betydning for fremtidig utvikling.

Erik Mashmann: Som praktikant fra Danmark, bidrar Eirik med sin kunnskap om nettverk, spesifikt virtuelle private nettverk (VPN). Hans bidrag spiller en viktig rolle i å støtte infrastrukturen og sikre kommunikasjonen mellom klient og server i systemet.

1.4 Begrepsdefinisjoner

I dette kapittelet introduseres nøkkelbegreper, som er grunnleggende for forståelsen av Mini Smart Factory Lab prosjektet. Definisjonene har som mål for å etablere et felles språk for lesere, og bidra til en bedre forståelse av prosjektets tekniske og teoretiske aspekter.

Smart factory
Smart factory eller en smart fabrikk på norsk blir definert i kontekst med den fjerde industrielle revolusjon som et høyt digitalisert og tilkoblet produksjonsanlegg som utnytter cyberfysiske systemer. Systemene bruker avanserte teknologier som kunstig intelligens (KI) for dataanalyse, automatisert prosesskjøring og kontinuerlig datahåndtering (Kumar et al., 2023). Smarte fabrikker har som hensikt å integrere maskiner, mennesker og enorme datavolumer i et sammenhengende, digitalt sammenkoblet økosystem.

Operasjonell teknologi (OT)
Operasjonell teknologi har som formål å monitorere og kontrollere enheter, prosessering og legge til rette for infrastruktur. Operasjonell teknologi er mest brukt i industrielle settinger gjennom maskinvare og programvare på produksjonsanlegg. Operasjonell teknologi skiller seg fra informasjonsteknologi, hvor OT enheter kontrollere på det fysiske nivået, mens IT står for å behandle data å drive applikasjoner (How Is OT Different From IT?, u.å.).

IT/OT-konvergens
IT og OT konvergensen blir referert til integrasjonen av IT systemer, brukt for databehandling, med OT designet for å overvåke og kontrollere produksjonsprosesser. Dette fremmer et helhetlig syn på bedriftsinformasjon og prosessadministrasjon, og sikrer at hver enhet har riktig informasjon til rett tid. Konvergensen av IT og OT gjør det mulig for sømløs interaksjon mellom digitale og fysiske systemer. Dette forbedrer den operasjonelle effektiviteten og muliggjør mer informert beslutningstaking gjennom analyse av sanntidsdata (Ehie & Chilton, 2020).

Mikrokontrollere
En mikrokontroller er en integrert krets som er kompakt laget for å gjøre en spesifikk operasjon i et innebygd system. De tekniske egenskapene til en mikrokontroller er prosesser, minne og input/output pins. De er ofte brukt som små integrerte komponenter til å kontrollere mindre funksjoner i en større komponent uten en form for operativsystem (What Is a Microcontroller and How Does It Work?, u.å.).

MQTT (Message queuing telementry transport)
MQTT er en nettverksprotokoll som krever lite båndbredde for å drive kommunikasjon, som gjør den godt egnet for Internet of Things (IoT) applikasjoner. Denne nettverksprotokollen gjør det mulig for flere enheter å kommunisere med hverandre over trådløst nettverk, uten å vite om hverandres identitet. MQTT kjøres på applikasjonslaget av TCP/IP modellen, som gjør at noder (klienter) kan publisere og subscribe til meldinger gjennom en såkalt broker, som distribuerer meldinger basert på topics (Kashyap et al., 2018).

Node-Red
Node-Red er et åpent kildekode, flytbasert, og hendelsesdrevet programmeringsverktøy. Dette verktøyet blir brukt til å kjøre tilkoblinger direkte med mellom maskinvare, API-er og tjenester på nett. Oppbygningen av et system i Node-Red presenteres gjennom et nettverk av noder som kommuniserer seg imellom. Disse nodene kan bli brukt til å styre og/eller hente data fra maskinvare for å videreføre dette i andre settinger (Node-RED, u.å.).

1.5 Oppgavens struktur

Oppgaven er inndelt i flere deler for å gi en oversikt over hvordan prosjektet ble gjennomført. Første kapittel som har blitt introdusert var selve prosjektet, bakgrunnen og formålet. Kapittel to diskuteres prosjektstyring og metodikk, vektlagt med de agile prinsippene som har blitt anvendt og hvordan dette har påvirket prosjektet. Analysefasen presenteres i kapittel tre, hvor vi går i dybden på systemdefinisjonene og problemområdene som blir adressert. Designvalg og viktige valg av programvare og fastvare tas videre opp i kapittel 4. Kapittel 5 fokuserer på selve prosjektgjennomføringen, hvordan en agil tilnærming med scrum ble tatt i bruk. Det fremhever også hvordan vi har praktisert kvalitetssikring og tidsstyring. I kapittel 6 reflekterer vi over det arbeidet som har blitt gjort, og lærdommer som tas med videre. Avslutningsvis kommer konklusjon, hvor viktige aspekter ved en IT og OT konvergens kommer frem, viktige elementer å tenke på til videre utvikling og hvorfor prosjektet var en suksess.

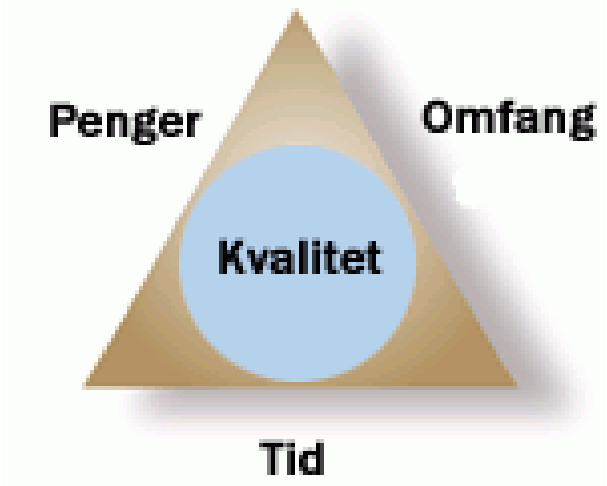
2. Prosjektstyring & Metodikk

Prosjektstyring handler om å etablere mål og planlegge aktiviteter for å nå målet (Rolstadås, 2020) og er avgjørende for gjennomføring av et prosjekt. Et hvert prosjekt gjennomføres i en unik kontekst og valgene man tar og hvilken metode som praktiseres, må tilpasses omgivelsene. Dette kapittelet gir leseren en innsikt i de strategiene, metodene, og verktøyene som vi har valgt for prosjektgjennomføring basert på prosjektets omgivelser. Kapittelet representerer prosjektet i prosjekttrekantens tre dimensjoner. Det presenterer prosjektets suksessfaktorer og suksesskriterier, samt en introduksjon til kvalitetsstyringsaktiviteter som blir brukt. Det argumenterer for hvorfor prosjektet er best tilpasset en agil fremgangsmåte og til slutt argumenterer vi for hvorfor vi har en objekt orientert analyse og design prosess (OOA&D), men ikke praktiserer objekt orientert programmering (OOP).

2.1 Prosjekttrekanten

Prosjektet og dets kontekst er sammensatt av tre grunnleggende dimensjoner: tid, penger og omfang, se Figur 3. Ethvert vellykket prosjekt krever en nøye balansering av disse dimensjonene, da dimensjonene er tett koblet og enhver endring i en dimensjon kan få konsekvenser for de andre (Prosjekttrekanten - Støtte for Microsoft, u.å.). Det er derfor avgjørende å forstå grensene og mulighetene som ligger innenfor hver dimensjon.

Dette prosjektet opererer innenfor et fastsatt budsjett som ikke er fleksibelt. Denne begrensningen har implikasjoner for tilgjengeligheten av teknologiske ressurser, både maskinvare og programvare, og setter dermed rammer for implementeringsstrategier. Videre er prosjektet underlagt en fastsatt tidsfrist, satt til 26.04.2024. Dermed må man forsøke å optimalisere tidsbruken for å oppnå prosjektmålene på en effektiv måte. Dette etterlater omfangsdimensjonen som den eneste tilpasningsdyktige faktoren. Omfanget defineres av funksjoner og krav som skal integreres i det ferdige produktet. Prosjektet har rom for endringer for visse funksjonaliteter basert på fremdriften og de målene som er satt.



Figur 3 - Prosjekttrekant (Penger + Omfang + Tid = Kvalitet) (Microsoft, u.å.)

2.2 Suksessfaktorer i prosjektstyring

For å tilrettelegge et godt arbeidsmiljø, har prosjektet definerte suksessfaktorer. «*Suksess faktorer er forhold som må ligge til rette for at målene skal nås*» (Rolstadås, 2023a). I følge Rolstadås vil fokus på suksessfaktorer under prosjektet øke sannsynligheten for et vellykket prosjekt. Det ble etablert suksessfaktorer som skulle øke sannsynligheten for at prosjektet ble vellykket. Teamet diskuterte emner som var relatert til hvorfor prosjekter mislykkes. (Andersen, 2016) forteller at prosjekter mislykkes på grunn av manglende kommunikasjon. Dette stemmer også overens med (Frederik Borgersen, 2018) sine påstander. Han nevner også at svakt eierskap fra toppledelse og uklare målsetninger er store årsaker for mislykkede prosjekter (Frederik Borgersen, 2018). Utfra diskusjon konkluderte man med at kommunikasjon i alle ledd er viktig for å unngå siloer og kommunikasjonssvikt. Basert på diskusjonsemnene utformet man følgende suksessfaktorer.

- God og klar kommunikasjon.
- Tett oppfølging med produkt eiere.
- Et trygt og godt teamforhold.
- God forståelse over det tekniske.
- Engasjement fra toppledelsen.

For å bevare fokus på suksess faktorene er dette alle temaer som ble tatt i betraktning under hele livsløpet av prosjektet for å sikre at teamet opprettholdte arbeidsmiljøet som tillot teamet å nå målet.

2.3 Suksesskriterier og kvalitetssikring

Dette delkapittelet fokuserer på suksesskriterier og kvalitetssikring, som er elementer som er viktige for å sikre prosjektets effektivitet og opprettholdelse av definerte standarder. Her vil vi utforske kriterier som definerer prosjektets suksess samtidig med de metodene og prosessene som er anvendt for å opprettholde høy kvalitet gjennom prosjektet.

2.3.1 Suksesskriterier

Suksesskriterier er indikatorer som er kvantitative og som blir brukt for å evaluere om et prosjekt er en suksess eller ikke (Rolstadås, 2023b). For resultatstyring benyttes SMART-modellen (Specific, Measurable, Achievable, Relevant, Time-bound). Suksesskriteriet skal være spesifikt nok til at man vet nøyaktig hva det er. De skal være målbart, med andre ord skal det skal være kvantifiserbart. Suksesskriteriet skal være oppnåelig og relevant slik at det er motiverende.

Utformingen av suksesskriteriene for dette prosjektet ble ledet av de ulike prinsippene som er i SMART-modellen, som sikrer at hvert punkt er spesifikt, målbart, oppnåelig, relevant og tidsbestemt. Det ble først identifisert at kjernefunksjonalitet var høyt prioritet slik at det grunnleggende for brukeren var dekket. Dette ble fulgt opp sammen med tidsperspektivet og at prosjektet ikke oversteg budsjettet. Vi inkluderte også tilbakemeldinger fra produkteier for å sikre at produktet oppfylte standarder for kvalitet og merverdi, noe som igjen er grunnleggende for en vellykket «Proof of Concept» (PoC).

Prosjektets suksesskriterier har som hensikt at de evaluerer faktorene tid, penger, og omfang fra prosjekttrekanten, se Figur 3, som ofte blir brukt til å evaluere digitaliserings prosjekter (Frederik Borgersen, 2018).

Prosjektets suksess kriterier er følgende:

- 100% av brukerhistorier kategorisert som «MUST HAVE» er implementert i produktet.
- Ingen testscenario relatert til brukerhistorier kategorisert som «MUST HAVE» feiler.
- Prosjektet er ferdigstilt innen fristen 26.04.2024.
- Prosjektet overstiger ikke budsjettet.

Ofte er ikke faktorene: tid, penger, og omfang nok til å evaluere real gevinst (Frederik Borgersen, 2018). For dette prosjektet gjelder det samme. Prosjektet kan levere på tid, inkludere alle brukerhistorier, ikke overstige budsjettet og fortsatt ikke skape merverdi for Knowit. Derfor har vi valgt og evaluerer produktet sammen med Knowit. Produktet skal demonstrere mulighetene ved konvergensen mellom IT og OT slik at Knowit kan selge konseptet og at produktet møter produkteier sine kravspesifikasjoner. Det er ønskelig at koden kan kvalifiseres som høy slik at dette kan gjenbrukes i andre kontekster. Til slutt skal produktet skape merverdi for Knowit.

Suksesskriterier for produktet:

- Produktet skal kunne kvalifiseres som et PoC.
- Produkteier er fornøyd med produktet.
- Kodekvalitet kan kvalifiseres som høy.
- Produktet skal skape merverdi.

2.3.2 Kvalitetssikring

«Kvalitetssikring er planlagte og systematiske aktiviteter som gjøres for å oppnå at et produkt eller en tjeneste vil oppfylle kravene til kvalitet» (Halbo, 2023a). For å sikre kvalitetssikring var det viktig at teamet var innforstått med produksjonsprosessene og preferansene til kunden som skulle leveres til. Noen av de systematiske aktivitetene som ble gjennomført var evaluering av implementerte brukerhistorier under sprint review´s eller utforming av risikoanalyser. Dette blir dypere diskutert i Delkapittel 5.5.

2.4 Samarbeidsmetodikk

Dette prosjektet omfatter temaer som er ukjente for prosjektdeltakerne. Vi har bakgrunn i informasjonsteknologi (IT) og informasjonssystemer (IS) med fokus på analyse, design og programmering av applikasjoner. Ukjente emner er blant annet programmering av maskinvare av operativ teknologi (OT), kabling av digitale- og analoge innganger og ukjente kommunikasjonsprotokoller. I tillegg skal produktet dekke kravspesifikasjonene som produkteier setter. På den andre siden, har vi et godt støtteapparat som består av kompetente fagfolk. Prosjektet krever også tilegning av ny kunnskap og hvordan dette kan anvendes på best mulig måte. Med dette i betraktning er man avhengig av hyppige tester og kontinuerlige tilbakemeldinger. Ifølge Atlassian, kjent programvareselskap av samarbeidsverktøy, velger man agile metodikker for imøtekomme endringer og tilbakemeldinger (Atlassian, u.å.-c). Derfor har vi en agil fremgangsmåte.

Videre utforskes de ulike agile prinsippene som ble praktisert i prosjekt. Agile metodikker er spesielt egnet i prosjekter hvor det skjer raske endringer og hvor det enkelt oppstår utfordringer (Embracing Uncertainty, u.å.). Dette var tydelig i vårt arbeid med ukjent teknologi og skiftene kravspesifikasjoner. Dette støttet for å tilpasse de utfordringene og tilbakemeldingene som kunne komme underveis. Igjen så sikrer dette at produktet utviklet seg i tråd med forventningene og kravene som var satt. Prinsippene ble diskutert i teamet hvor fire agile prinsipper ble valgt som skulle sikre utviklingen av produktet. Tidligere erfaring var den største faktoren for valgene.

1. Kontinuerlige tester og leveringer.
2. Tett samarbeid med produkteiere og resten av teamet.
3. Tilrettelegge for et godt arbeidsmiljø hvor man stoler på hverandre at jobben blir gjort, eksempelvis ved å være tilpasningsdyktig og være ærlig med hverandre.
4. Vær åpne til skiftende krav, selv sent i utviklingen. Smidige prosesser utnytter endringer for kundens konkurransefortrinn.

(12 Principles Behind the Agile Manifesto | Agile Alliance, 2015).

Prinsippene (1) og (2) har bidratt med å sørge for at kvaliteten på produktet og gjennomførelse av prosjektet ikke har redusert underveis. Prinsipp (3) har skapt initiativ for å skape et godt arbeidsmiljø som tilrettelagte for ansvar under frihet. Prinsipp (4) skapte en ønskelig holdning hvor man alltid tenkte på hva kunden ønsket.

2.5 Analyse- og Designmetodikk

Ved dette prosjektet har vi valgt å ha en objektorientert fremgangsmåte ved analyse og design. OOAD er velkjent fremgangsmåte i både modellering og programmering av programvare (Mathiassen et al., 2018). OOA (objekt orientert analyse) lar oss modellere problemet i den virkelige verden gjennom objekter, deres attributter, kommunikasjon og oppførsel. Det tillater for å abstrahere slik at man kan begynne smått og legge på detaljer («Object-Oriented Analysis and Design | OOAD», 2020), som er bra for en agil og iterativ utviklingsprosess. OOD (objekt orientert design) lar oss modulisere systemkomponenter og tilknytte data og arbeidsoppgaver til objekter. Det skaper en felles forståelse for systemet, skaper en semantikk for systemet, og tillater for justeringer og gjør det skalerbart («Object-Oriented Analysis and Design | OOAD», 2020). Begrensninger som OOAD har er at det øker kompleksiteten og det øker de teknologiske kostnadene i form av de totale beregningskapasitetene til systemet på grunn av man er nødt til å skape instanser av objekter. En annen begrensning er at OOAD er at det er tidskonsumerende («Object-Oriented Analysis and Design | OOAD», 2020).

Ettersom dette prosjektet bruker IoT-enheter som har lavere beregningskapasisteter på grunn av lavere prosessorhastighet og minnekapasitet, brukes OOAD bare som en analyserings- og design metode, og der objekt orientert programmering (OOP) blir ikke brukt. Dermed utelukker man dette problemet. For å hindre at OOAD tar mer tid enn nødvendig, har vi en agil fremgangsmetode, dermed jobber vi med det iterativt. I tillegg benyttes scrum rammeverket hvor oppgaver som skal gjennomføres planlegges for hver sprint. På den måten hindres OOAD relaterte oppgaver å ta mer tid enn nødvendig.

Med en god forståelse av de analytiske og designmessige metodene, er vi godt forberedt til å utforske mer spesifikke systemkrav og teknologiske kontekster i det kommende kapittelet. Kapittel tre vil bygge direkte på dette grunnlaget, og etablere videre inn i de spesifikke analysene og teknikkene som er nødvendig for å utforme en robust arkitektur som møter både de nåværende og fremtidige behovene i prosjektet.

3. Analyse

Kapittel tre fokuserer på den initiale analysefasen av prosjektet. Dette kapittelet beskriver metodene og teknikkene som ble anvendt for å analysere systemkravene og den teknologiske konteksten prosjektet omfatter. Denne fasen la grunnlag for beslutninger av design og bidro til å skape en arkitektur som var tilpasset de faktiske forholdene og utfordringene prosjektet stod ovenfor. Det bidro til å skape en robust arkitektur, knyttet opp til faktiske utfordringer og forholde. Emner som blir diskutert er prosjektets systemdefinisjon. Hvilke objekter som representerer problemdomenet (problemet i den virkelig verden). Og kort til slutt applikasjonsdomenet som handler om grensesnittet mellom brukeren og systemet.

3.1 Systemdefinisjon

Som tidligere nevnt er ofte kommunikasjonssvikt årsaken til mislykkete prosjekter. Kommunikasjonssvikt skjer ofte når personer har ulik forståelse av samme emne. Av den grunn er en systemdefinisjon nyttig. En systemdefinisjon skaper et felles synspunkt av systemet (Mathiassen et al., 2018). Systemets kontekst foregår i to domener, problemdomenet og applikasjonsdomenet. Problemdomenet er det systemet skal overvåke og kontrollere, mens applikasjonsdomenet handler om hvordan og hvem som bruker systemet (Mathiassen et al., 2018).

Systemet integrerer IT og OT i en industriell kontekst preget av teknologi som ikke kan erstattes eller oppgraderes. Den primære arbeidsoppgaven av systemet er å starte Mandals sine produksjonsprosesser av slanger under forutsetning av at ressursene er tilgjengelige, forhindre dersom ressurser ikke er tilgjengelige, og stoppe produksjonsprosessen i henhold til lengden definert i bestillingen. I tillegg skal systemet oppdatere tilgjengelige ressurser etter en produksjonsprosess. Systemets sekundæroppgave er å generere sanntidsdata og blokkdata relatert til en bestilling. Systemaktiviteten er todelt, omfattende av både bestillings- og produksjonsfasen. Bestillings- og konfigureringsprosessen er designet for å kunne håndtere varierte miljøer gjennom en pc, mens produksjonen er automatisert og foregår i et statisk miljø preget av høy aktivitet hvor sanntidsdata og blokkdata er tilgjengelig gjennom en Human-Machine Interface (HMI).

Etter å ha definert systemdefinisjonen, benyttet vi FACTOR-metoden for å bryte ned systemdefinisjonen i mer konkrete og håndterbare deler. Hensikten med FACTOR metoden, var for å supplementer systemdefinisjonen slik at det var enklere å få en overordnet forståelse. Ved bruk av FACTOR klarte vi å definere og vurdere hver funksjon av systemet detaljert. Dette fremmet en forbedret kommunikasjon mellom teammedlemmene, som var viktig for utviklingsprosessen. Denne tilnærmingen sikret også at systemkomponentene møtte prosjektets krav og målsettinger, se Tabell 1.

Functionality	Systemet skal kunne motta bestillinger av slanger som selges av Mandals AS og evaluere hvilke og hvor mye ressurser bestillingen krever, og evaluere om man har nok ressurser tilgjengelig. Systemet skal kunne starte produksjonsprosessen og loggføre aktivitetene som skjer underveis og stoppe produksjonsprosessen iht. bestillingen. Etter produksjonsprosessen er ferdig skal systemet oppdatere tilgjengelige ressurser.
Application domain	Administreres av kunde ved bestilling, mens produksjonsdata og sanntidsdata administreres av en ansatt.
Context	Behov for automatisering på produksjonsgulvet og behov for datainnsikt i produksjonsprosessen på en kostnadseffektiv måte.
Technology	Bestillinger skjer via PC. Systemet kjøres på en Raspberry Pi. Generering av sanntidsdata skjer på en mikrokontroller (Arduino Uno Wifi). Monitorering av sanntidsdata og blokkdata visualiseres på en Human Machine Interface (HMI).
Objects	Bestilling (konfigurator), produksjonsdata, event (sanntids-data), device, måling, ressurs (beholdning), brukte ressurs Slange, tpu, vev, og kobling
Responsibility	Iverksette produksjonsprosesser av slanger dersom man har nok tilgjengelige ressurser, dokumentere prosessen og stoppe den iht. bestilling.

Tabell 1 - FACTOR elementer

3.2 Problemdomene

Problemdomene behandler informasjon som systemet overvåke, forvalter og manipulerer (Mathiassen et al., 2018). Denne informasjonen representerer tilstanden til problemdomenet. På den måten kan systemet endre tilstanden til problemdomenet. Informasjonen blir representert gjennom objekter og informasjonen endrer seg gjennom kommunikasjon mellom systemkomponentene og objektene.

3.2.1 Klasser/objekter

Byggeklossene til et system er objekter og et objekt er en entitet med en identitet, tilstand, og oppførsel (Mathiassen et al., 2018). For å avgjøre hvilke objekter som systemet skulle håndtere startet man med å analysere bestillingen fra produkteier med verb/substantiv metoden. Metoden handler om å identifisere objekter (David J. Barnes & Michael Kölling, 2016, s.558), hvor substantivene fra bestillingen blir objekter og verb er handlinger objektene kan gjøre. Fordi systemet ikke bruker OOP blir verb handlinger systemet skal gjøre. Etter de initiale objektene ble lokalisert diskuterte vi med produkteier og sammenlignet det med andre lignende systemer hvor første utkast ble satt. Ettersom prosjektet er agilt har man endret etter behov, eksempelvis er at objektet «Brukt ressurs» ble implementert etter et møte med økonomidirektør av Mandals AS.

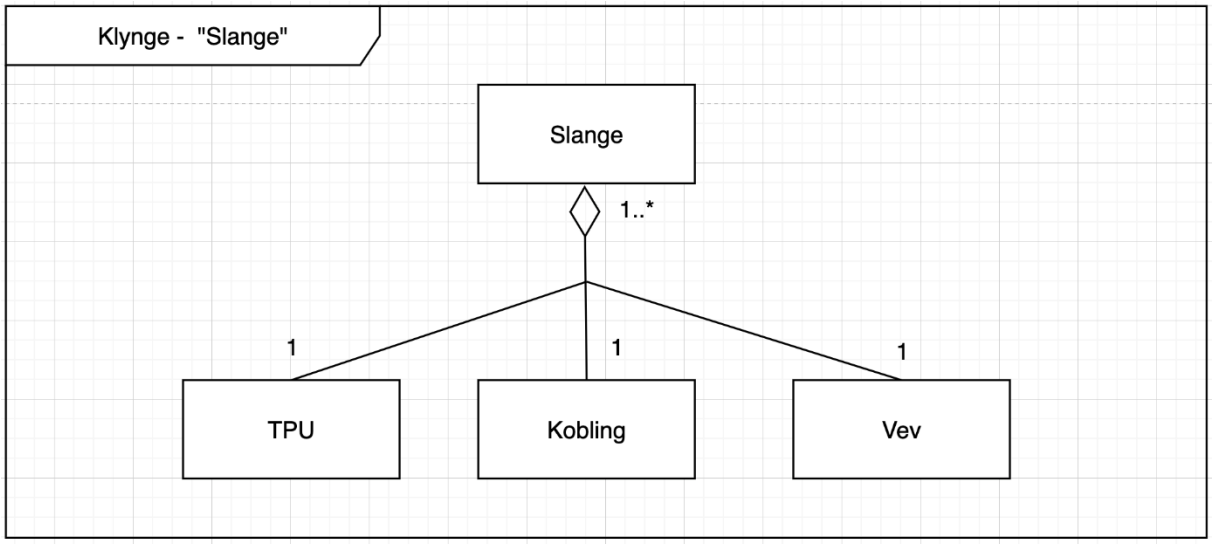
Objektene som representerer problemdomenet, er følgende:

Slange	Produksjonsdata
Vev	Event
TPU	Måling
Kobling	Device
Bestilling	Ressurs
Brukt ressurs	

Tabell 2 - Problemdomenets objekter

Klynger:
Klynger hjelper å skape en helhetlig oversikt over problemdomene ved å fordele det i mindre subdomener. Nedenfor har vi to klynger som representerer håndtering av ressurser og en produksjonsprosess hvor en bestilling er bindeleddet mellom dem. Hensikten med klyngene er for å gi en bedre forståelse over hvilke objekter som kommuniserer med ressurser. Notasjonen er hentet fra boken Databasesystemer skrevet av Kristoffersen 2021 og illustrerer koblingen mellom entitetene og forholdet mellom entitetene. Forholdet som brukes er aggregering som betyr at entiteten er en del av en annen, men kan eksistere uten en direkte kobling. Det er fordi dersom man fjerner en entitet er det ikke ønskelig å fjerne alle entiteter som har et forhold til den.

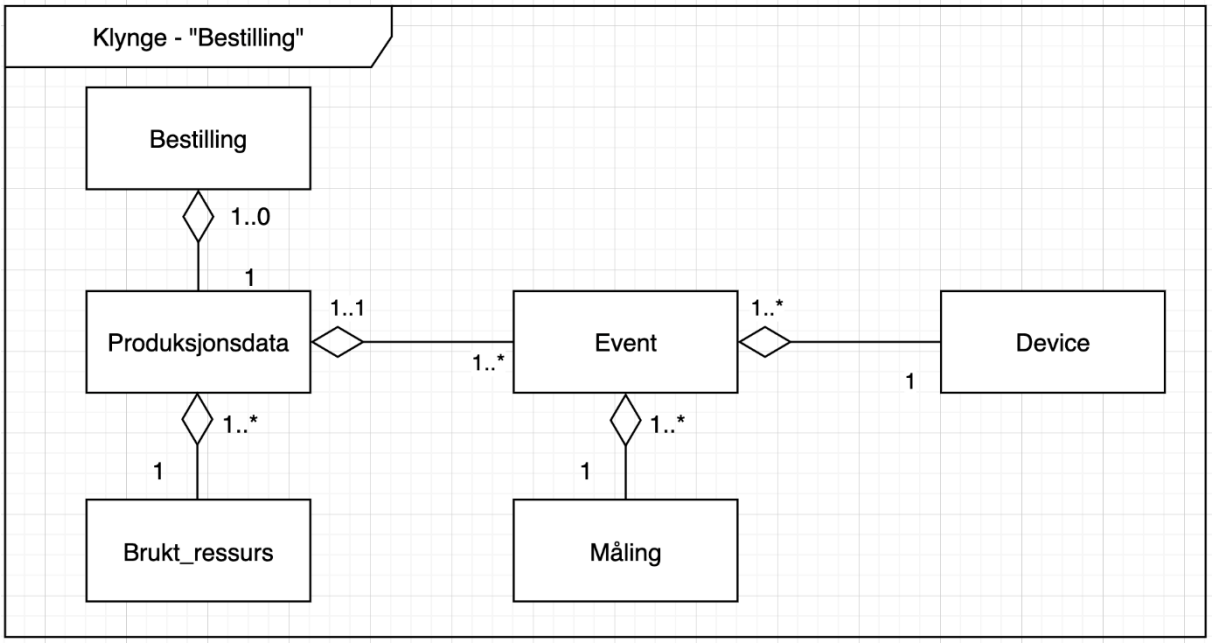
Klynge slange:
Denne klyngen fungerer som en kokebok. En slange består av en TPU, en kobling, og en tråd (vev). Det finnes mange kombinasjoner av TPU, slange, og tråd i en slange og de ulike kombinasjonene krever ulik mengde ressurser. Ved hjelp av klyngen får man oversikt over hvilke kombinasjoner som er mulig, og hvilken mengde ressurser det krever. Klyngen i kombinasjon med entiteten «bestilling» og tabellen «ressurs» kan man sjekke at en bestilling ikke overstiger tilgjengelige ressurser.



Figur 4 - Klynge Slange

Klynge bestilling:

Klyngen representerer en bestilling som er koblet til en produksjonsprosess hvor alle hendelser tilknyttet til produksjonsprosessen finnes. En forekomst av et «produksjonsdata - objekt» skjer ikke før systemet vet at nødvendige ressurser er tilgjengelig (Figur 2). Alle hendelser er også tilknyttet en enhet (device) og forteller hvilke målinger som for eksempel, «lengde» eller «hastighet». Etter produksjonsprosessen er ferdig, tilknyttes ressursene som er brukt til produksjonsdataen.



Figur 5 - Klynge Bestilling

3.2.2 Hendelsestabell

Et «event» blir definert som en hendelse som involverer et eller flere objekter (Mathiassen et al., 2018). Tabell 3 viser hendelsestabellen, som tar for seg arbeidsoppgavene som systemet har og fordeler til de forskjellige enhetene som kommuniserer sammen. Hensikten med hendelsestabellen er å skape en oversikt over hvor hendelsen skjer. Informasjonsflyten er kompleks og strekker seg over flere teknologiske komponenter. Derfor har vi også lagt ved «S1» notasjonen som beskriver hvor startsignalet kommer fra og hvor den faktiske hendelsen skjer. På denne måten gir hendelsestabellen en god oversikt over hvor hendelser både starter og skjer. Hendelsene er tilknyttet til brukerhistoriene som igjen er prioritert i brukerhistoriene, se Tabell 4 og 5, dermed vil ikke alle hendelser være implementert i systemet som gjerne er prioritert som «Should Have».

X = kan skje flere ganger

- = skjer én eller ingen ganger

S1 = start signal

ID	Event	Arduino uno Wifi – Rotary Encoder	RPi Central Server SCADA	GUI
1	Starte en produksjon.		X	S1
2	Stoppe en produksjon.	S1	X	
3	Produsere tidsseriedata.	X		
4	Lagre avvik.		X	
5	Lagre blokkdata for en produksjon.		X	
6	Kunne visuelt se varsling av systemfeil.		S1	X
7	Se blokkdata etter produksjon.		S1	X
8	Se tilstanden til maskinene i sanntid.		S1	X
9	Sjekke hvilke ressurser bestillingen krever.		X	
10	Sjekke tilgjengelig ressurser.		X	
11	Forhindre arbeider å starte en bestilling hvor		X	

	man ikke har nok ressurser.			
12	Oppdatere ressurser etter produksjon		X	
13	Generere rapport			X
14	Lagre tidsseriedata		X	

Tabell 3 - Hendelsestabell

3.3 Applikasjonsdomene

Applikasjonsdomene relaterer til hvordan systemet skal samhandle med problemdomenet og hvem som skal samhandle med systemet (Mathiassen et al., 2018). Dette er grunnlaget for forretningskravene og blir beskrevet gjennom brukerhistorier. Brukerhistoriene tar høyde for hvem som samhandler med systemet, hvordan man samhandler med systemet (brukergrensesnitt), funksjonen systemet gjennomfører, og hvilken merverdi dette medfører brukeren. Nedenfor vises to brukerhistorier fra prosjektet:

Prioritet	Must have Hendelse ID: 9, 10, 11
Brukerhistorie	Som produksjonsarbeider ønsker jeg å sjekke oppskriften til slangen for den aktuelle bestillingen for å sikre at alt er klart.
Funksjon	Systemet skal sjekke hvilke ressurser for å produsere den gitte bestillingen og forhindre start av en produksjon som krever ressurser som ikke er tilgjengelig.
Kriterier:	Stoppe bestillinger som krever ressurser som ikke er tilgjengelige.
Test-scenario	1. Sjekke at systemet ikke starter dersom det ikke er tilstrekkelig av ressurser.
Argumentasjon	Forhindre menneskelig feil som å starte en produksjon uten tilstrekkelige ressurser.
Tilhørende Use-Case	1. Starte en produksjon/bestilling som krever mer ressurser enn det som er tilgjengelig. 2. Se at produksjonen ikke starter.
Funksjon:	READ & SIGNAL
Grad av Kompleksitet:	HIGH

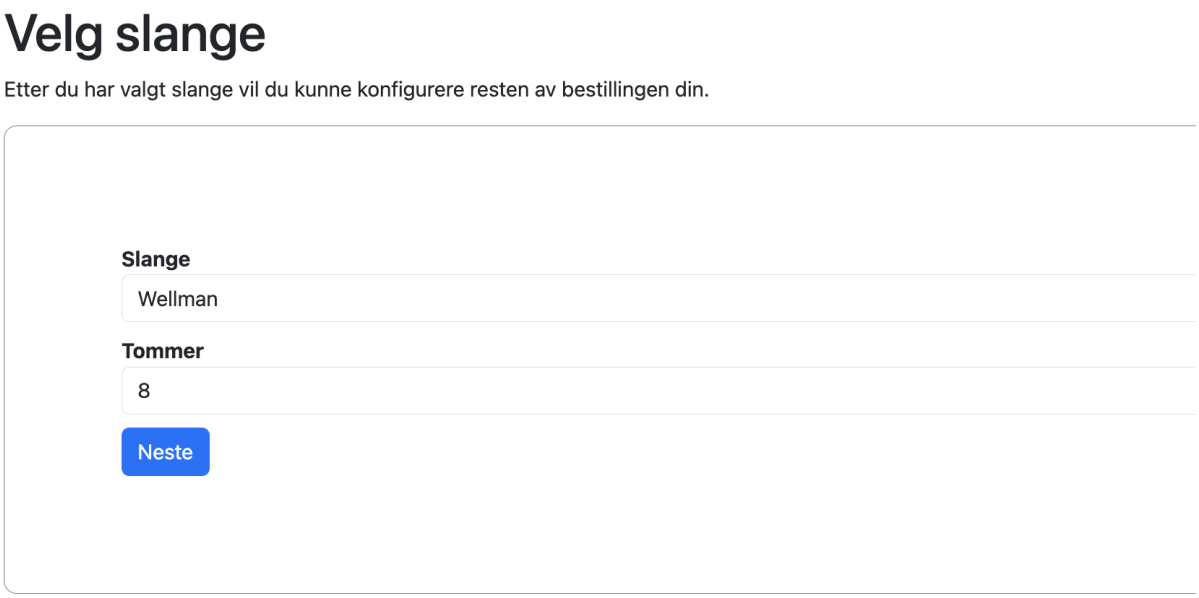
Tabell 4 - Eksempel på brukerhistorie – actor - produksjonsarbeider

Prioritet	Must have Hendelse ID: 7
Brukerhistorie	Som en produksjonsleder ønsker jeg å kunne se blokkdataen som har blitt laget knyttet opp mot det som har blitt produsert
Funksjon	Systemet skal vise data som har blitt generert til en bestilling
Kriterier:	Fremvisning av blokkdata medgått i produksjon
Test-scenario	1. Sjekke at blokkdataen stemmer overens med bestilling
Argumentasjon	Å kunne se blokkdata som er knyttet opp mot det som er blitt produsert, er essensielt for effektiv kvalitetskontroll i en SmartFactory sammenheng.
Tilhørende Use-Case:	1. Produksjonslederen sjekker opp fullførte produksjoner 2. Systemet viser en liste over fullførte produksjoner 3. Produksjonslederen velger en spesifikk produksjon 4. Systemet presenterer blokkdata knyttet til den valgte produksjonen; tidsstempler, forbrukte ressurser og produksjonsparametere
Funksjon	READ
Grad av kompleksitet:	MEDIUM

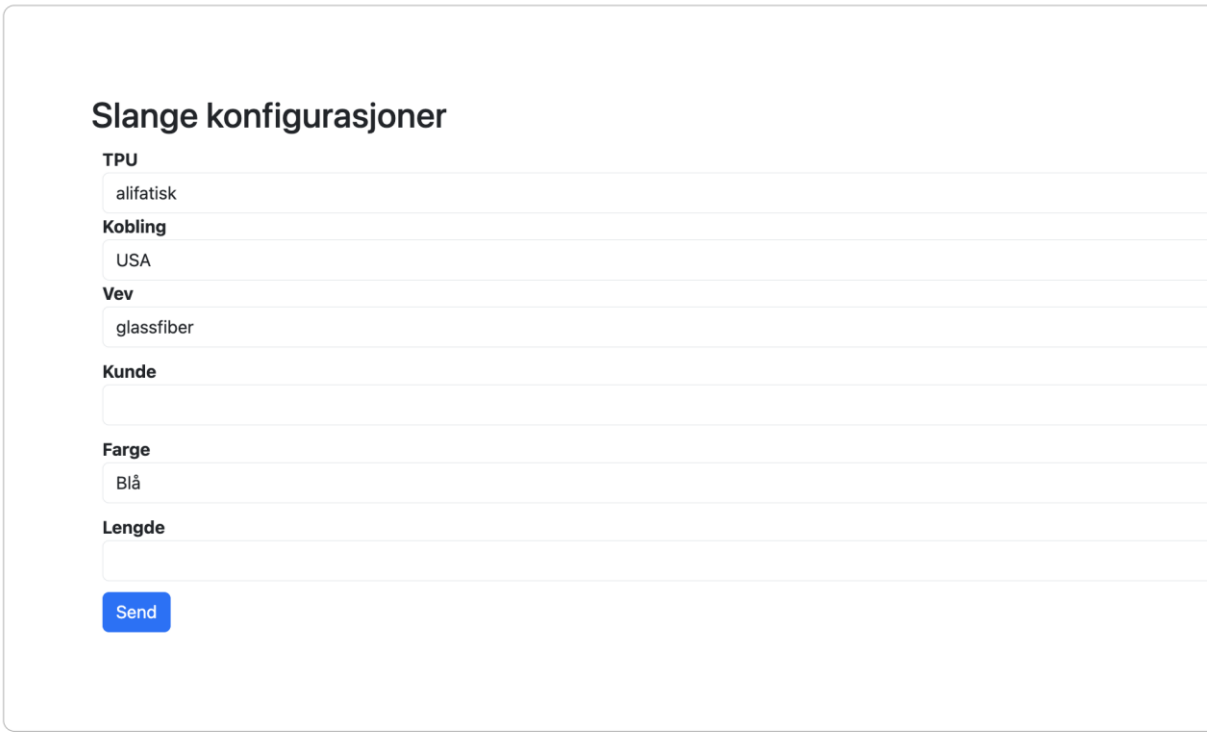
Tabell 5 - Eksempel på brukerhistorie – actor - produksjonsleder

Dette la grunnlaget for hvilke brukergrensesnitt som var nødvendige. Man var avhengig av to brukergrensnitt. Et brukergrensnitt som tillot en bruker å sende inn en bestilling. I tillegg var man avhengig av et brukergrensnitt for å overvåke sanntidsdataen og blokkdataen av produksjonen. Med hensyn til at design og brukervennlighet ikke er viktig i dette prosjektet har ikke dette vært hovedfokuset, men et godt design har ingen store mangler (Mathiassen et al., 2018), dermed har man gjort det etter best evne og tid. Valgene er basert på standarder og kravspesifikasjoner av kunde samtidig som det er blitt evaluert kontinuerlig gjennom sprint reviews.

Det første brukergrensnittet tillater for at man kan sende inn en bestilling. Prosjektet har valgt «form fill-in pattern» som er velkjent og simplifiserer innsetting av riktig data (Mathiassen et al., 2018) illustrert i Figur 6 og Figur 7.

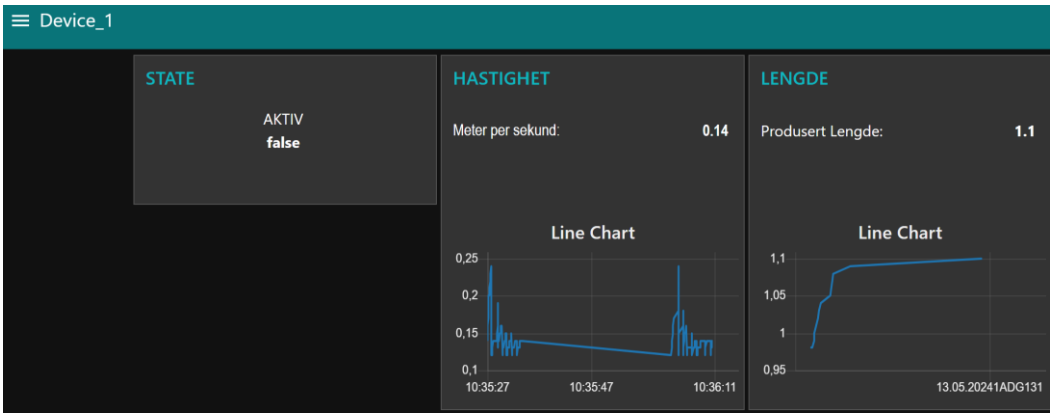


Figur 6 - Brukergrensnitt for bestilling - vindu 1

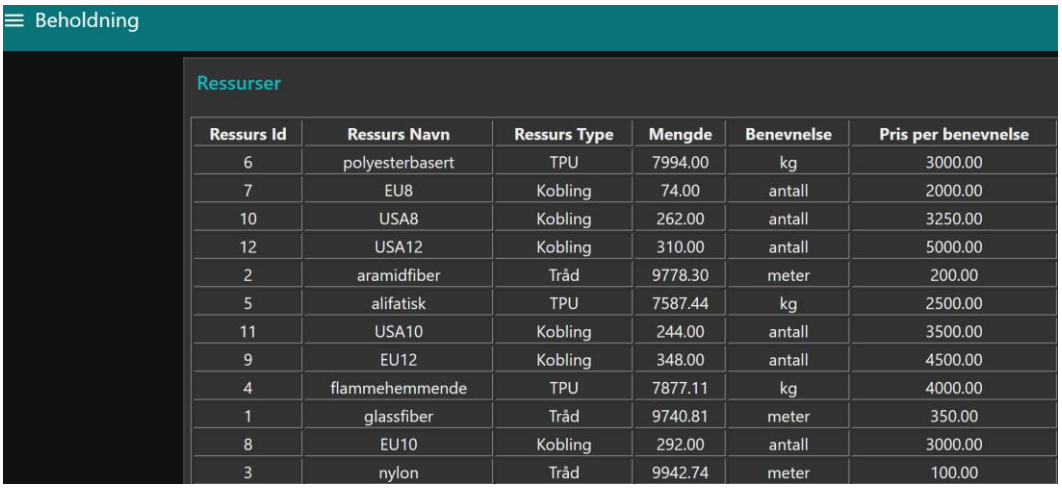


Figur 7 - Brukergrensnitt for bestilling - vindu 2

Det andre brukergrensesnittet var for å visualisere sanntidsdataen, beholdning og produksjonsdataen. For sanntidsdataen valgte vi linjediagrammer i kombinasjon med tekst for å visualisere dataen, se Figur 8. Dette tillatte for at man kunne se alle datapunktene under produksjonsprosessen (*Lage, forstå og presentere statistikk - Diagrammer - Medie- og informasjonskunnskap 1 - NDLA - NDLA*, u.å.). Beholdningen ble visualisert gjennom en tabell, hvor en har oversikt over de ressursene som er tilgjengelige, se Figur 9. For produksjonsdata valgte man å visualisere dataen i tabeller slik at man fikk et helhetlig bilde av hva som har gått med av ressurser og kostnader i produksjonen, se Figur 10.



Figur 8 - Brukergrensesnitt status Device_1



Ressurs Id	Ressurs Navn	Ressurs Type	Mengde	Benevnelse	Pris per benevnelse
6	polyesterbasert	TPU	7994.00	kg	3000.00
7	EU8	Kobling	74.00	antall	2000.00
10	USA8	Kobling	262.00	antall	3250.00
12	USA12	Kobling	310.00	antall	5000.00
2	aramidfiber	Tråd	9778.30	meter	200.00
5	alifatisk	TPU	7587.44	kg	2500.00
11	USA10	Kobling	244.00	antall	3500.00
9	EU12	Kobling	348.00	antall	4500.00
4	flammehemmende	TPU	7877.11	kg	4000.00
1	glassfiber	Tråd	9740.81	meter	350.00
8	EU10	Kobling	292.00	antall	3000.00
3	nylon	Tråd	9942.74	meter	100.00

Figur 9 - Brukergrensesnitt beholdning av ressurser

Figur 10 - Brukergrensesnitt produksjonsdata tilhørende produksjon

Etter en grundig gjennomgang av analysemetodene og identifikasjon av nøkkelområder innen systemdefinisjonene i kapittel tre, er vi nå klare for å anvende disse innsiktene i de praktiske designbeslutningene som presenteres i kapittel 4. Dette kapittelet tar for seg designvalg som vil bidra de teoretiske analysene til funksjonelle løsninger og legge til rette for realiseringen av prosjektet.

4. Design

Dette kapitelet tar for seg hvordan systemet er designet. Det fremhever systemets begrensninger og hvilke designkriterier som er viktige for dette prosjektet. Deretter tar det for seg hvordan databasen er strukturert, hvilken arkitektur systemet har og viktige valg av programvare, teknologiske komponenter, og kommunikasjonsprotokoller. Kapittelet tydeliggjør også hvordan design valgene henger sammen med designkriteriene.

4.1 Designkriterier

Før man prioriterer designkriteriene er det viktig at man har en oversikt over hvilke begrensninger systemet har. Mathiassen et al. definerer dimensjoner som kan begrense systemet, den tekniske, organisatoriske og menneskelige. Den tekniske dimensjonen hvor maskinvare og programvare begrenser systemets muligheter. Den organisatoriske dimensjonen fremhever hvordan kravspesifikasjoner fra organisasjonen, planer for videreutvikling, og hvilke arbeidsmetodikker deltakere i prosjektet bruker begrenser valgmulighetene, som for eksempel systemets arkitektur (Mathiassen et al., 2018). Den siste dimensjonen Mathiassen et al. nevner er den menneskelige dimensjonen hvor deltakernes kompetanse og erfaring begrenser hva man kan oppnå basert på tilgjengelig tid. Innsikt i systemets begrensninger har vært viktig for dette prosjektet slik at man kan ta bedre valg som balanserer disse. Tabell 6 gir en oversikt over hvilke begrensninger som er relevante for systemet.

Dimensjon	**Begrensninger**
Teknisk	Maskinvare begrensninger. Programvare begrensninger.
Organisatorisk	Planer for videreutvikling. Arbeidsfordeling mellom avdelinger.
Menneskelig	Erfaring med lignende systemer og tekniske plattformer.

Tabell 6 - Systemets dimensjoner og relevante begrensninger

Ifølge Mathiassen et al. er designkriterier ønskelige attributter som systemet skal ha og det er viktig å prioritere designkriteriene for å styre designvalg mot ønskelig slutt produkt. Ut ifra forretningskravene har vi fordelt de i funksjonelle- og ikke-funksjonelle krav. Funksjonelle kravspesifikasjoner beskriver hva systemet er nødt til å gjøre, mens ikke-funksjonelle kravspesifikasjoner beskriver hvordan systemet skal oppføre seg (Dennis et al., 2015), eksempelvis at systemet skal ha en grad av sikkerhet.

Vi har valgt ut noen designkriterier som er viktig for dette prosjektet. Kriterier som er valgt er basert på Mathiassen et al., (2018) sin liste av klassiske design kriterier, forretningskravene av oppdragsgiver (funksjonelle og ikke funksjonelle kravspesifikasjoner), systemets begrensninger, og prosjektets tre dimensjoner (tid, penger, omfang), se Figur 3 - prosjekttrekanten. Tabell 7 viser designkriteriene relevante for dette prosjektet som er valgt fra listen av klassiske designkriterier laget av Mathiassen et al. I tillegg legger vi ved to designkriterier som ikke blir fremhevet i Tabell 8, men som er like viktige styringsfaktorer for prosjektet, se Tabell 8.

Kriterier definert av Mathiassen et al., (2018):

Kriterier	Mål av
Usable	Systemets tilpasningsdyktighet iht. den organisatoriske konteksten.
Secure	Systemets evne til å sikre kommunikasjonen
Correct	Systemets evne til å møte forretningskravene.
Reliable	Presisjon av systemets funksjonelle krav.
Maintainable	System evne til å lokalisere og håndtere bugs.
Flexible	Systemets evne til å modifiseres.
Comprehensible	Anstrengelse av å få en helhetlig forståelse av systemet.
Reuseable	Potensialet for at deler av systemet kan bli brukt i andre relaterte kontekster.
Interoperable	Kostnadene/muligheten til å koble seg til andre systemer.

Tabell 7 - Definisjon av design kriterier - Mathiassen et al., 2018

Selvdefinerte Kriterier:

Kriterer	Mål av
Technical Cost	Den teknologiske kostnaden av å kjøre systemet.
Scalability	Systemets evne til å skalere.

Tabell 8 - Selvdefinerte design kriterier

Hvordan man balanserer kriteriene er viktig. Kriteriene er avhengige av hverandre. En prioritering av et kriterium vil påvirke muligheten til å prioritere en annen. Hvordan man balanserer kriteriene er avhengig av konteksten av systemet, men ifølge Mathiassen et al., (2018) balanserer et godt design flere kriterier. Balansering av designkriterier er basert på kunde sine kravspesifikasjoner, prosjektets omfang, se Figur 3 prosjekttrekanten,

Kriterier	Veldig viktig	Viktig	Mindre viktig	Lett oppfylt
Usable	X			
Secure			X	
Correct		X		
Reliable	X			
Maintainable		X		
Flexible		X		
Comprehensible		X		
Reuseable		X		
Interoperable				X
Techincal Cost		X		
Scalability	X			

Tabell 9 - Prioritering av design kriterier

Som Tabell 9 illustrerer er Usable, Reliable, og Scalability prioritert som veldig viktig. Med andre ord prioriterer man systemets evne til å tilpasse seg den organisatoriske konteksten, systemets presisjon av funksjonelle krav, og systemets evne til å skalere (ref. Tabell 8). Kriteriene skiller seg fra viktig, ved at veldig viktig er kritisk for systemet suksess. Det er viktig at systemet kan kunne anvendes i flere forskjellige organisatoriske kontekster fordi deler av systemet skal kan bli gjenbrukt av flere kunder. I tillegg kan den organisatoriske konteksten endres over tid, dermed er det viktig at systemet ikke er en flaskehals for det. Presisjon i OT er veldig viktig. Det er helt nødvendig at hendelser skjer ved angitt tidspunkt fordi konsekvensene av at det ikke skjer, kan være stor. For eksempel hvis produksjonen av en slange ikke stopper. Det vil medføre store økonomiske tap av ressurser. Som nevnt, kan den organisatoriske konteksten endres og den er forskjellig avhengig av kunden. Dermed er det viktig at systemet kan skaleres dersom det er behov.

Designkriteriene Correct, Maintainable, Flexible, Comprehensible, Reusable, og Techincal Cost er prioritert som viktig. Systemet skal møte forretningskravene (Correct), det skal være mulig å håndtere bugs (Maintainable), dermed skal det også være mulig å foreta endringer (Flexible). Ettersom systemet er grunnlaget for videreutvikling er det viktig at systemet er enkelt å forstå (Comprehensible). Systemet skal kunne gjenbrukes hos flere kunder (Reusable). Til slutt skal systemet være en kostnadseffektivt alternativ for kunder, derfor bruker vi IoT-enheter. Med dette i betraktning er det viktig at systemet ikke overstiger beregningskapasiteten til enhetene som brukes (Techincal Cost).

Designkriteriet Secure er prioritert som mindre viktig fordi det var ikke hovedfokus for leveringen av prosjektet på grunn av tid og omfang, men sikkerhet i en IT og OT kontekst er viktig. Derfor er dette tatt med i betraktning under prosjektet livsløp. Eksempelvis er implementering av VPN for sikker kommunikasjon mellom server og klient eller tilrettelegging for autentisering gjennom x509 sertifikater for maskin til maskin (M2M) kommunikasjon.

Siste designkriteriet er Interoperable som angår systemets evne til å koble seg til andre systemer og tredjeparter. Dette er viktig fordi kunder kan ha nødvendig data lagret i forskjellige systemer. Derfor må systemet ha mulighet å tilkoble koble seg til systemet. Dette er prioritert som lett oppfylt fordi Node-red, som er den tekniske plattformen som styrer kommunikasjonsflyten, tilrettelegger for nettopp dette.

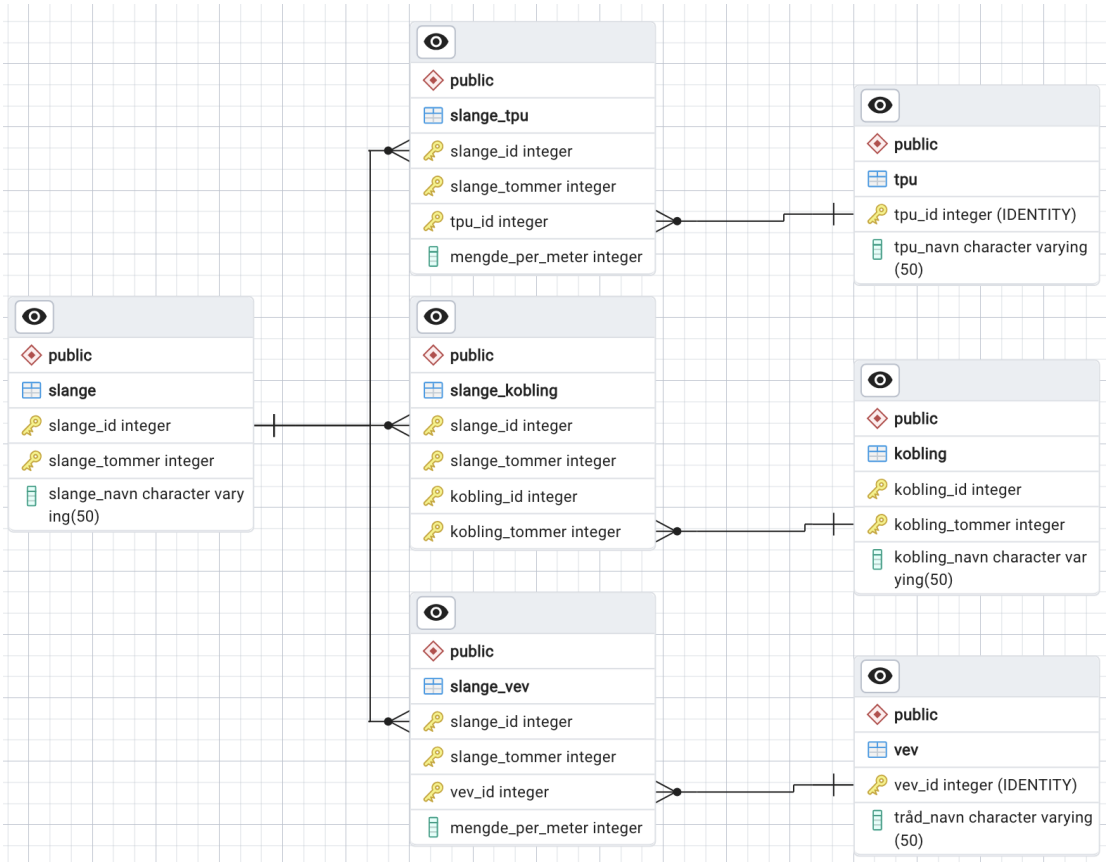
4.2 Database

Strukturen av databasen er viktig dersom man ønsker en hensiktsmessig database. En hensiktsmessig database tilrettelegger for skalerbarhet og dataintegritet. Høy grad av redundans fører til unødvendig lagring av data slik at dataen blir inkonsistent (Bjørn Kristoffersen, 2021). Dette er fordi det fører til forskjellige anomalier. Eksempelvis kan det forekomme oppdateringsanomalier ved at data er lagret flere steder og at man ikke oppdaterer alle lagringsplassene. Dette svekker databasens dataintegritet. For å unngå dette kan man normalisere databasen. Normalisering er en måte man kan dele opp databasen i tabeller og attributter for å fjerne redundans (Kristoffersen, 2021) slik at databasen kan håndtere dataen uten anomalier.

Databasen følger normaliseringslovene opp til tredje normalform (3NF). Det vil si at tabellene har atomære attributter (1NF), ingen partielle avhengigheter (2NF), eller transitive avhengigheter (3NF). Det finnes flere normaliseringslover, men i noen tilfeller vil det ikke være hensiktsmessig å følge alle normaliseringsformene fordi flere tabeller resulterer i et tregere system (Kristoffersen, 2021). Med hensyn til designkriteriene Scalability, Techincal Cost, Comprehensible, Flexible, og Usable følger ikke databasen normaliseringslovene utover 3NF. Se Tabell 7 (Definisjon av design kriterier), Tabell 8 (Selvdefinerte design kriterier), og Tabell 9 (Prioritering av design kriterier).

På bakgrunn av systemets tekniske begrensninger, se Tabell 4 (Systemets dimensjoner og relevante begrensninger), er designkriteriet Technical Cost prioritert som viktig. Ved å tillate for kontrollert redundans, hindrer man å dele opp databasen i flere tabeller og attributter som vil øke kompleksiteten av systemet. Det øker effektiviteten og reduserer anstrengelsen av å få en helhetlig forståelse av systemet (designkriteriet – Comprehensible). Ved å øke antall tabeller og attributter fastslår man seg til et design. Dette er ikke ønskelig, med hensyn til designkriteriet Flexible og Usable hvor tilpasningsdyktighet iht. organisatorisk kontekst er viktig.

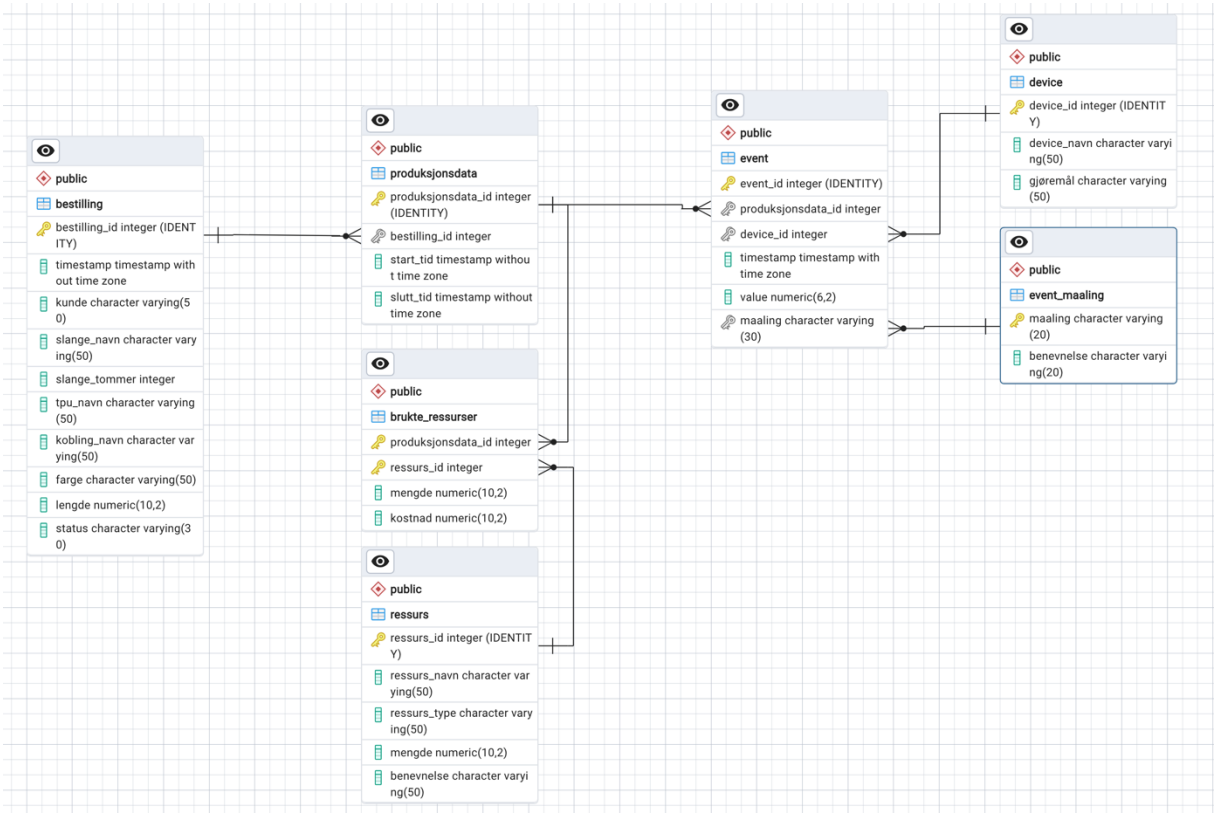
Databasen skal være skalerbar og gjenbrukbar i andre kontekster (designkriteriet Scalability og Reusable). For å besvare dette har man brukt metoder som å modellere abstrakt ved bruk av klynger og generalisering. Klyngen Slange, representert i Figur 4 er skalerbar ved at man kan enkelt legge til flere attributter til entitetene (slange, tpu, vev, kobling) og definere forholdet mellom entitetene ved hjelp av koblingstabellene (slange_tpu, slange_vev, og slange_kobling), se Figur 11. Dersom man legger til attributter er man også nødt til å refaktorisere SQL-spørringer i systemet. Denne klyngen er ikke gjenbrukbar fordi den er spesifikt designet for Mandals AS.



Figur 11 - ERD av klyngen slange

Klyngen Bestilling, representert i Figur 5 er også skalerbar ved at man kan skalere attributtene til entitetene, hvor man er også avhengig av å refaktorisere SQL-spørringer i systemet.

Databasen legger til rette for at man kan skalere med flere enheter som tar forskjellige målinger, se figur 12 (tabellene event, device, og event_maaling). Klyngen Bestilling er gjenbrukbar i motsetning til klyngen Slange. Dette oppnår man ved at entitetene er generalisert slik at det kan bli gjenbrukt flere steder.



Figur 12 - ERD av klyngen bestilling

Selv om systemet ikke normaliserer utover 3NF, som kan påvirke dataintegriteten til systemet (Kristoffersen, 2021), velger man å ikke normalisere utover 3NF. Dette er på bakgrunn av tester som viser at systemet møter dataintegritet kravene og designkriteriene Reliable og Correct, se Tabell 7. Til slutt ønsker vi å nevne at datatypene er valgt med hensyn til design kriteriet Techincal Cost. Det er ønskelig å bruke så lite minne som mulig, men ikke hindre innsetting av data. Dermed bruker man datatypen integer hvor det er mulig som alternativ for datatypen numeric fordi den krever større minneplass. Numeric er brukt hvor det er nødvendig for presisjon som for eksempel ved raden value i tabellen event. Datatypen varchar(50) definerer en variabel lengde som ikke kan overstige 50 karakterer. På den måten kan vi begrense minne allokeringen slik at systemet er fleksibelt frem til 50 karakterer.

4.3 Teknisk arkitektur

Vedlegg A viser kommunikasjonsflyten fra når en bestilling blir gjennomført frem til produksjonsdataen blir visualisert. Vedlegget er en kombinasjon av et flytdiagram og den tekniske arkitekturen for å fortelle hvor funksjonene skjer. Den tekniske arkitekturen er designet slik at det er enkelt å skalere med flere enheter som kommuniserer med den sentrale serveren. Kommunikasjonen er gjennom et lukket lokalt nettverk som øker sikkerheten, se

Tabell 7 med designkriteriet Security. Kommunikasjonen baserer seg på pub/sub arkitektur over MQTT protokollen som reduserer den tekniske kostnaden, se Tabell 8 designkriteriet Technical Cost. Kommunikasjonen mellom klient og server er også adskilt og skjer gjennom et Virtuelt Privat Nettverk (VPN) for å øke sikkerheten, se tabell 7 designkriteriet Security og Vedlegg J – Automasjonspyramiden med VPN.

4.4 Komponent arkitektur

Dette delkapittelet tar for seg de ulike arkitekturene som har vært sentrale i utformingen av systemet. Her blir client/server-, pub/sub- og hendelsesdrevet arkitektur lagt frem. Dette vil diskutere hvordan hver av disse bidrar til systemets funksjonalitet og samhandling.

4.4.1 Client/Server arkitektur

Klient – Server arkitektur, eller med en andre, to lags arkitektur er en av de mest vanlige arkitekturene som brukes. De fleste web-applikasjoner bruker arkitekturen to lags arkitektur (Dias et al., 2022). Klient-server arkitekturen er en løsning for å distribuere systemfunksjoner til flere geografiske lokasjoner eller klienter av definisjon (Mathiassen et al., 2018). Arkitekturen tillater for klienter å kommunisere med systemet over Hyper Text Transfer Protokoll (HTTP). Fordelene med dette er at man kan spesifisere hvilke systemfunksjoner klientene har tilgang til og klientvolumet er skalerbart (Computer Network | Client and Server Model - Javatpoint, u.å.) (designkriteriet Scalability). Det skaper også mulighet å fraskrive ansvar fra serveren over på klient slik at serveren bruker mindre krefter (Mathiassen et al., 2018) (se tabell 8 designkriteriet Technical Cost). Det skiller klienten og serveren slik at man kan foreta endringer hos en av de uten å påvirke hverandre (designkriteriene Usable og Flexible). Skille gjør det også enklere å lokalisere bugs fordi kilden er enten klienten eller serveren (designkriteriet Maintainable). Det tilrettelegger også for sikkerhetsmekanismer for autorisering og autorisasjon (designkriteriet Secure).

4.4.2 Pub/Sub arkitektur

Pub/Sub arkitekturen er et attraktivt alternativ for å bygge skalerbare og distribuerte systemer (Chen et al., 2018). Det tillater for en hybrid «peer-to-peer» (P2P) kommunikasjon mellom noder i systemet. P2P-kommunikasjon betyr at det ikke er en node som har kontroll, slik som klient/server modellen, men kontrollen fordeles til flere noder (Peer-to-Peer Network Meaning & Definition, 2024). Dette realiseres gjennom pub/sub arkitekturen hvor hver node kan forespørre data ved å «subscribe» på «topics» og sende data ved å «publisere» data tilknyttet til «topics» ved hjelp av en broker som distribuerer dataen. I vårt prosjekt ble det valgt å bruke Mosquitto broker, som er en åpen kildekode løsning som kommuniserer over MQTT protokollen. MQTT protokollen er bedre egnet for kommunikasjon mellom enheter med lav beregnisngskapasitet. Det tillater for å selektere dataen man ønsker gjennom «topics», som gjør størrelsen på dataen mindre. Det tillater også for at man kan enkelt inkludere flere «topics» og noder som samsvarer med designkriteriet Scalabiltiy.

4.4.3 Hendelsesdrevet arkitektur (EDA):

Hendelsesdrevet arkitektur (Event Driven Architecture – EDA) fokuserer på hendelser og reagerer på disse og en slik arkitektur er ofte bygget videre på pub-sub arkitekturen (Dias et al., 2022). Dette systemet gjør nettopp dette, men kombinerer også klient/server arkitekturen hvor en forespørsel fra en klient er en hendelse som systemet reagerer på.

4.4.4 Sammendrag

For å forklare arkitekturen på en kortfattet måte så har systemet en hendelsesdrevet arkitektur hvor den sentrale serveren lytter etter forespørsler fra en klient som iverksetter P2P kommunikasjonen mellom noder hvor systemet igjen reagerer på hendelsene.

4.5 Valg av programvare, teknologiske komponenter, og kommunikasjonsprotokoller.

I dette delkapittelet vil vi utforske beslutningene bak valg av programvare, teknologiske komponenter og kommunikasjonsprotokollene som støtter funksjonaliteten og effektiviteten av vårt system. Det starter med å beskrive valg av programvare, hvor vi fremhever hvor viktig det er at systemet er skalerbart, ytelse og kompatibilitet. Deretter vil valg av de teknologiske komponentene legges frem og hvordan disse komponentene kan integreres til systemet. Til slutt kommer valgene av kommunikasjonsprotokollene som blir brukt med fokus på hvordan disse påvirker dataflyten og integrasjon i systemet.

4.5.1 Valg Programvare

Valg av programvare kan være avgjørende for ytelsen av systemet (tzsoftno, 2020), spesielt når man jobber med maskinvare med begrenset beregningskapasitet. Faktorer som avgjørende for beslutningen av programvare er skalerbarhet, ytelse, kompabilitet, utviklingsmiljø, tid, og økonomi.

Node-red

I følge Dias et al (2022), er Node-Red et av de mest brukte løsningene for prosjekter som bruker IoT-enheter. Dette utviklingsverktøyet er et visuelt verktøy hvor man tilknytter noder skrevet i JavaScript sammen for å lage informasjonsflyter. Dette gjør det tidseffektivt å arbeide med. I tillegg er det et stort utviklingsmiljø med mange tilgjengelige biblioteker med god dokumentasjon. Dette gjør at vi kan arbeide med flere forskjellige maskinvarer, APIér, og tredje partier. Med andre ord er det kompatibelt og skalerbart, se Tabell 7 designkriteriet Interoperable og Scalability. Det er også gratis i motsetning til andre alternativer med dyre lisenser som går over budsjettet, se Figur 3 prosjekttrekanten.

Programmering av Arduino med C++ og Arduino IDE
For å programmere Arduino bruker vi programmeringsspråket C++ gjennom Ardunio IDE som støtter programmering av Arduino sine mikrokontrollere. Språket baseres på språket C/C++ og har flere nyttige biblioteker (Dias et al., 2022).

PostgreSQL
Kommunikasjonen mellom den sentrale serveren og databasen består av både lese- og skriveoperasjoner, men på grunn av tidsseriedataen vil det være mer «skrive» operasjoner enn «lese» operasjoner. I følge Amazon Web Services (AWS) presterer PostgreSQL bedre enn MySQL ved skrive-operasjoner (PostgreSQL vs MySQL - Difference Between Relational Database Management Systems (RDBMS) - AWS, u.å.). I tillegg presterer den bedre enn "mariadb" ved både lese og skriveoperasjoner (MariaDB vs PostgreSQL - Difference Between Open-Source Relational Databases - AWS, u.å.). Dette samsvarer godt med design kriteriet «Techincal Cost». I tillegg gir det mulighet for å bruke Timescaledb som er bygget på toppen av PostgreSQL. Timescaledb er en tidsserie database som øker effektiviteten til spørringer, og det tilrettelegger for skalering (Timescale Products, u.å.). Dette er også i tråd med designkriteriet Scalability.

4.5.2 Valg av komponenter

For å sikre at systemet kjøres problemfritt og kvaliteten opprettholdes var valg av komponenter viktig. Videre vil det bli diskutert hvert valg av fastvare og hvorfor de ble valgt for å sikre funksjonalitet og effektivitet.

Arduino Uno Rev 4 m/WiFi
Mikrokontrollere står sentralt i dette prosjektet for å samle inn sanntidsdata fra produksjonsprosessene og ettersom avdelingen Smart Factory satser på å implementere IoT løsninger, havnet valget på en Arduino Uno Rev 4. Denne mikrokontrolleren er 32-bit som har en innebygd WiFi modul (UNO R4 WiFi | Arduino Documentation, u.å.). Dette var krav vi stilte til utstyr for å håndtere data. Denne mikrokontrolleren er bindeleddet mellom å hente data fra produksjonen og sende over MQTT kommunikasjon til broker som er kjørt på serveren.

Raspberry Pi 4 Model B 8GB
Valget av hvor den sentrale serveren, databasen og grensesnittet skulle bli kjørt på måtte ha egenskaper som var henhold til kravspesifikasjonene til programvaren samtidig som at den skulle ha plass inne i boksen som skulle monteres. Valget havnet på en Raspberry Pi 4 Model B 8GB. Dette er en single board computer som er stillegående, energieffektiv samtidig som den har de nødvendige tilkoblingene som trengs iht. kravspesifikasjonene som; USB-tilkoblinger, Ethernet, WiFi, bluetooth og micro-HDMI (Raspberry Pi Ltd, u.å.). Muligheten å kjøre Linux var også viktig ettersom det er åpen kildekode, som muliggjør skreddersydde konfigurasjoner som oppfyller de spesifikke prosjektbehovene(«Linux», 2024). Den støtter komplekse databehandlinger oppgaver og kan kjøre Node-Red og mosquitto-broker for å håndtere dataflyten mellom ulike komponenter i systemet.

Motor
Motoren som skal stå på hjulet har som formål til å starte produksjonen av slanger. Motoren som blir brukt er en Nema 23 23hs5628 57 moto 2.8a og ble valgt grunnet sin robusthet og presisjon som er nødvendig for å drifte hjulet i Mini smart Factory Lab. Denne stegmotoren tilbyr hastighetskontroll for presis manipulasjon av industrielt utstyr. Driveren som hører med, TB6600, gir effektiv strømstyring som sikrer jevn og stabil drift (Nema 23 23hs5628 57 Motor 2.8a Med Tb6600 Driver Nema17 23 For CNC ..., u.å.). Dette gjør det til en ideell motor for å drive hjulet i et automatisert produksjonsmiljø.

Rotary Encoder
Rotary encoder ble valgt i dette prosjektet på grunn av dens høye presisjon og pålideligheten den har i å måle rotasjonsbevegelser. Dette er en nøkkelkomponent som tar for seg å sende signaler til mikrokontrolleren med data om hvilke retning hjulet beveger seg i og overvåkning. Dette er mulig på grunn av at den er to-faset og er essensielt for presis prosesskontroll i automatiserte produksjonssystemer(New And Original Incremental Rotary Encoder Lpd3806-600bm-g5-24c Ab Two Phase 600ppr Diameter 38mm Shaft 6mm - Buy Lpd3806-600bm-g5-24c,Pulse Encoder,Cheap Price Encoder Product on Alibaba.com, u.å.). Komponentet er valgt fordi den faktisk kan brukes i industrielle kontekster.

4.5.3 Valg av kommunikasjonsprotokoller

Dette delkapittelet utforsker beslutningene bak valg av kommunikasjonsprotokoller som støtter systemets behov for effektiv datakommunikasjon, her vil MQTT for maskin-til-maskin kommunikasjon diskuteres i forhold til vår EDA-arkitektur og bruk av HTTP i kontekster der det er hensiktsmessig.

Message Queue Telemetry Transport (MQTT)
På grunn av systemets EDA arkitektur mellom den sentrale serveren og maskinene brukes MQTT protokollen. MQTT protokollen er en kommunikasjons protokoll med fokus på M2M kommunikasjon (Dias et al., 2022). Årsaken til at den er egnet for mindre enheter er fordi den teknologiske kostnaden er mindre. I tillegg tillater det for «Unifed Name Spacing» (UNS) gjennom Sparkplug-B som øker skalerbarheten («Hva er Unified Namespace (UNS)?», u.å.) (Implementing Unified Namespace (UNS) With MQTT Sparkplug, 2022) (Tabell 7, designkriteriet Scalability)

Hyper Text Transfer Protocol (HTTP)
HTTP er den mest vanlige kommunikasjonsprotokollen, men er ikke egnet for IoT-kommunikasjon på grunn av dens kompleksitet og høye teknologiske kostnad (Dias et al., 2022). Til tross for dette er frekvensen av forespørsler som bruker HTTP så lav at man kan tillate bruken. Denne kommunikasjonen er mellom serveren og klienten (Server-Klient-Arkitektur).

Etter å ha beskrevet designvalgene og beskrevet de tekniske arkitekturene i kapittel fire, flyttes fokuset mot gjennomgjøringen av selve prosjektet i kapittel fem. Dette kapittelet viser hvordan teoriene og konseptene blir omsatt i praksis gjennom en agil tilnærming. Hvordan vi drog nytte av dette med å opprettholde kvalitet, administrere tidsfrister og tilpasse oss de dynamiske forholdene.

5. Prosjektgjennomføring - Scrum i praksis

Basert på den agile karakteren til prosjektet og prosjektdeltakernes tidligere erfaringer, ble det besluttet å ta i bruk scrum rammeverket. Agile prosjekter kjennetegnes av flere iterasjoner sammenlignet med sine ikke-agile motparter (Ellis, 2016). Dette tillater gjentatte analyse-, design-, kode- og testfaser hvor tilpasninger kan gjøres, noe som muliggjør respons på endringer fremfor rigid planlegging (Manifesto for Agile Software Development, u.å.). Slik reduseres tiden som vanligvis brukes på den initiale fasen, ofte preget av en omfattende analyse- og designprosess, og gir rom for en bedre forståelse av systemet før viktige designvalg tas (Ellis, 2016). Dette kapittelet tar for seg rollefordelingen, møtestrukturer, prosjektstyringsverktøy, artefaktene fra scrum rammeverket, og til slutt hvordan kvalitetssikring og tidsstyring sammen var med på å gjøre prosjektet til en suksess.

5.1 Roller i Scrum teamet: Gjennomgang av rollene i teamet Smart Factory

Denne seksjonen tar for seg de ulike rollene i scrum teamet, som har sin hensikt å vise hvordan dette bidro til å drive prosjektet fremover. I begynnelsen av prosjektet ble det etablert tre ulike roller, som hadde vær sin funksjon under prosjektets levetid, dette var en produkteier, en scrum master og prosjektdeltakere. De involverte medlemmene i teamet var: Arnt Aske, Trygve Solheim, Kristoffer Sand, Torkel Ivarsøy og Asbjørn Hestnes. I introduksjonen presenteres også Parisa Karampour og Erik Mashmann, men disse hadde ikke like aktive roller i prosjektet, men var fortsatt viktig for prosjektet. Siden vi gjennomførte dette prosjektet i en nyetablert avdeling i Knowit Sør, samtidig som vi bare var to studenter, gikk noen av rollene inn i hverandre for å skape gode omgivelser for prosjektet. Videre beskrives ansvarsområdet og bidragene til de involverte.

5.1.1 Produkt eier

Kristoffer og Trygve ble definert som produkteiere, som hadde sitt ansvarsområde med å definere produktets visjon og mål. Disse var ansvarlig for å planlegge selve prosjektet, prioritere arbeidsoppgaver og funksjoner i produktets backlogg. De sikret også at teamet forsto kravene og målsetningene for hver sprint. Kristoffer og Trygve hadde en avgjørende rolle i å formidle kundebehov og tilbakemeldinger til teamet, samtidig sikre at sluttproduktet møtte forventningene til interessentene.

5.1.2 Scrum master

Arnt ble delegert som scrum master i prosjektet, som sto for å lede scrum møter som ukentlig stands, sprint planlegning, sprint review og sprint retrospektiv. Hans bidrag var ment for å fjerne hinder som kunne påvirke teamets produktivitet og fremgang. Han sikret også at de ulike scrum prosessene ble ivaretatt og fulgt gjennom prosjektet. Nøkkelrollen han ga til teamet var at han sikret et godt arbeidsmiljø og at prosjektet holdt seg til tidsrammene som var avgjørende for suksessen til prosjektet.

5.1.3 Prosjektdeltaker

Torkel og Asbjørn fungerte som prosjektdeltakere med hovedfokus på å utvikle og implementere de tekniske løsningene prosjektet krevde. Samtidig som å definere oppgaver i hver sprint, estimering og utførelse av oppgaver og sikre kvalitet og funksjonalitet i leveransene. Som studentene i prosjektet, hadde Torkel og Asbjørn en «hans-on» rolle i tekniske aspekter av prosjektet. Vi jobbet tett med både scrum master og produkteier for at prosjektet ble gjennomført på en smidig måte og at sluttproduktet møter funksjonelle og ikke-funksjonelle kravspesifikasjoner.

5.2 Møtestruktur

Iterasjonene ble organisert i tre ukers sprinter. Dette sikret gjennomføring av nødvendige scrum arrangementer som sprintplanlegging, ukentlige scrum møter, sprintgjennomgang og sprintretrospektiv, samtidig som det ga rom for systemutvikling.

Formålet med scrum arrangementene er treleddet. "Sprintplanlegging" dedikert til å definere arbeidsoppgavene for den kommende sprinten, og avholdes ved starten av hver sprint. "Ukentlige scrum møter" fungerer som en plattform for kommunikasjon innad i teamet, hvor gjennomført arbeid, fremtidige planer og fremskritt vurderes for å sikre tilfredsstillende fremgang mot målene. Denne møteformen bidrar til å identifisere eventuelle problemstillinger og tilpasninger som kan være nødvendige. "Sprintgjennomgang" evaluerer måloppnåelse i henhold til sprintplanen og krav fra produktets eier, mens "Sprintretrospektiv" fokuserer på å evaluere hvordan sprinten ble gjennomført med sikte på å identifisere områder for forbedring til neste sprint.

5.2.1 Standups

I et større scrum team er det normalt å ha daglige standups, og oppdatere hva hver enkelt har gjort og skal fortsette på for å holde flyt i arbeidshverdagen. I vårt prosjekt ble dette tilpasset ettersom det ble jobbet i par for det meste. Her fant vi en balanse mellom daglige og ukentlig møter for å sikre effektivitet og forhindre kommunikasjonssvikt. Fortsettelse tar for seg hvordan dette ble løst gjennom daglige standups, ukentlige standups og tilpasning og fleksibilitet

Daglige møter

Gitt vår lille gruppestørrelse, ble det avholdt korte og mer konkrete møter oss imellom, for å raskere diskutere dagens mål, utfordringer og fremdrift. Dette ble gjort ettersom resten av teamet ikke var nødvendig å involvere i flere sammenhenger. Disse møtene tillot oss å raskt indentifisere problemer, noe som var viktig for å opprettholde moment i arbeidsflyten.

Ukentlige møter
Det ble avholdt ukentlige møter sammen med teamet, hvor alle ble oppdatert på status, spørsmål og diskusjon av fremdrift og funksjoner. Her ble det tatt en dypere dykk i våre langsiktige mål og progresjon. Alt som ble gjort fortløpende, uke til uke, ble dokumentert i en PowerPoint og fremvist for hvert ukentlig møte, se Vedlegg G.

Tilpasning og fleksibilitet
Det ble bemerket tidlig i prosjektet at fleksibilitet og tilpasning i møtestrukturen var viktig. Om nødvendig ble daglige møter forlenget for å håndtere problemer som var komplekse eller korte ned på for å få raskere flyt i arbeidshverdagen.

5.2.2 Sprint planning
Sprint planleggingen var en kritisk del av scrum syklusen, hvor det ble etablert hva som skulle gjennomføres i de aktuelle sprintene. Før vi gikk inn en ny sprint ble det etablert en gjennomgang av backloggen og evaluering av arbeidsmengden basert på forrige sprint, alt etter de arbeidsoppgavene som hadde eller ikke hadde blitt gjort. De oppgavene som ikke ble fullført tide, ble diskutert om de var relevante eller ikke, for så å fjerne eller dra med videre inn i neste sprint. Som tidligere nevnt, hadde vi fra start en overordnet plan for hva hver sprint skulle inneholde, slik at størrelsen på prosjektet var håndterbart. Dette var de milepælene som ble satt på starten av prosjektet, se Vedlegg E. Etter at de ulike oppgavene hadde blitt kartlagt ble det tatt en gjennomgang på det ukentlige møte for å fastsette at det vi hadde planlagt var i deres favør også.

5.2.3 Retrospektiv
Formålet med retrospektiv er å gi teamet en avsatt tid til å reflektere over den siste sprinten som hadde blitt gjennomført og identifisere hva som fungerte bra, hva som kunne forbedres og hvordan tilrettelegge for neste sprint.

Disse møtene ble avholdt etter hver sprint var ferdig, og fungerte som en god måte på å reflektere på hvordan vi jobbet og hvordan vi kunne forbedre videre i prosjektet. Møtene ble organisert på en enkel måte, hvor det ble diskutert spørsmål som: Hva fungerte bra? Hva fungerte mindre bra? Hva fungerte ikke? Hva kan vi gjøre annerledes? Det ble skrevet et referat underveis hvert møte med de viktigste punktene som kunne videreføres til neste sprint se, Vedlegg F. Disse referatene ble først dokumentert i verktøyet Azure DevOps som vist i Vedlegg F, men fant ut at det krevde mye fra hvert teammedlemmene å gjøre det i verktøyet, så dermed ble disse møtene mer muntlige, som ga bedre tilbakemeldinger og dokumentert i et eget dokument.

5.3 Prosjektstyringsverktøy – Azure DevOps

I dette prosjektet ble verktøy Azure DevOps, som var vår primære prosjektstyringsplattform, brukt for å ha overblikk av prosjektet, samtidig som det var dette oppdragsgiver praktiserte og ønsket å bruke for å holde seg oppdatert på planlegning og fremdrift og for planlegning.

5.3.1 Verktøyoversikt

Azure DevOps er et DevOps verktøy som hjelper team å planlegge arbeid, samarbeide på kodeutvikling, og bygge og distribuere applikasjoner. Vi brukte en rekke av funksjonene av plattformen gjennom prosjektet, de funksjonene som ble mest praktisert var Boards, Repos og Wikipages.

Azure Boards ble brukt for å opprette brukerhistorier og oppgaver. Det ga en interaktiv, visuell oversikt over alle arbeidsoppgaver og hvor de befant seg. Gjennom sprintene ble dette aktivt brukt og oppgavene som ble opprettet ble flyttet til status aktiv mens en jobbet med en oppgave og deretter til lukket når oppgaven var ferdig se Vedlegg C.1.

Azure Repos, se Vedlegg C.2, var vår kodebase, som er en sentralisert versjonskontroll som bruker git repositorium som tillot teammedlemmene å arbeide parallelt uten konflikter.

Azure Wikipages ble brukt til å dokumentere viktig emner som er fundamentale for domene kunnskapen se Vedlegg C.3. Dette ble i hovedsak brukt for å gi en rapport videre til teamet på hvordan problemer ble løst og dokumentasjon for utviklerne til å se å lære av slik det ble en felles forståelse av domenet.

5.3.2 Integrering med Scrum

Azure DevOps tilbyr en naturlig integrering av scrum prosessen. Dette tillot oss å bruke funksjoner som støtter de agile prinsippene og praksisene som ble fulgt. Under er direkte eksempler på hvordan vi integrerte verktøyet med scrum metodikken.

Sprint planning: Azure boards var sentral for prosessen ved sprintplanleggingen. Dette ble brukt for å prioritere oppgaver alt etter hvor lang tid vi estimerte hver oppgave tok og tildele dem til kommende sprinter. Dette muliggjorde slik at teamet kunne se hva som var planlagt for hver sprint og se målene for hver sprint.

Standups: Verktøyet gjorde det visuelt enkelt hver uke det ble holdt en ukentlig standup sammen med teamet, slik de kunne se hva som hadde blitt gjort og hva som skulle ta for seg resten av uken. Samtidig som den ga et overblikk for hele teamet, hjalp dette oss som studenter ved å sjekke daglig den fremgangen og arbeidsoppgavene som skulle gjøres.

Review: Etter hver sprint, ble Azure DevOps tatt i bruk for å se om vi hadde nådd de målene vi hadde satt oss. Dette gjorde det enkelt å analysere ytelsen vår, identifisere forbedringsområder og dokumentere handlingsplaner for de kommende sprintene.

Implementeringen av Azure DevOps som en integrert del av vår scrum tilnærming, styrket vår evne til å levere programvare raskere og mer pålitelig. Ved brukt av verktøyet ble det skapt en kultur for kontinuerlig forbedring. Dette fungerte ikke bare som et verktøy for oppgavestyring, men også kjerne i prosjektledelse og var avgjørende for å støtte vår anvendelse av agile prinsipper.

5.4 Artefakter

Ved bruk av metodologien scrum, er artefakter et sentralt begrep, dette er verktøy som hjelper teamet å prioritere, organisere og spore fremgang av prosjektet. Disse fører til åpenhet og innsikt for de deltagende medlemmene og interessentene (Atlassian, u.å.-a). Videre i delkapitlene følger hvilke artefakter som har blitt tatt i bruk og hvordan disse ble implementert for å overholde orden i prosjektet.

5.4.1 Produkt backlogg

Produkt backloggen er en dynamisk liste over alt som trengs i prosjektet, hvorav funksjoner, funksjonaliteter, krav, forbedringer og feilrettinger er lagt til (Atlassian, u.å.-b). Listen er et viktig fundament for planlegging av leveranser i scrum metodikken slik at alle er oppdatert på prosjektets prioriteringer og retning.


USER STORY 1444*

1444 Lagre blokkdata/produksjonsdata knyttet til en produksjon.

Unassigned 0 comments Add tag

Description

Prioritet: Must have, Hendelse ID: 5

Brukerhistorie
Som en produksjonsleder ønsker jeg å lagre blokkdata som har gått med i en produksjonsprosess

Funksjon
Systemet skal registrere blokkdata som går med i produksjonsprosesser

Kriterier:
Lagring av blokkdata medgått i en produksjon

Test-scenario
Sjekke at blokkdata blir lagret i henhold til bestilling

Argumentasjon
Ved å lagre blokkdata som er knyttet til ulike produksjoner, kan en enkelt finne fram om kunden får problemer med slangen

Tilhørende Use-Case:

1. Etter en produksjonsprosess er ferdig lagres relevant data for bestillingen
2. Systemet lagrer dataen og validerer den i en sentral database, knyttet til den spesifikke bestillingen og produksjonen
3. Hente ut data knyttet til bestilling

Funksjon: UPDATE
Grad av kompleksitet: MEDIUM

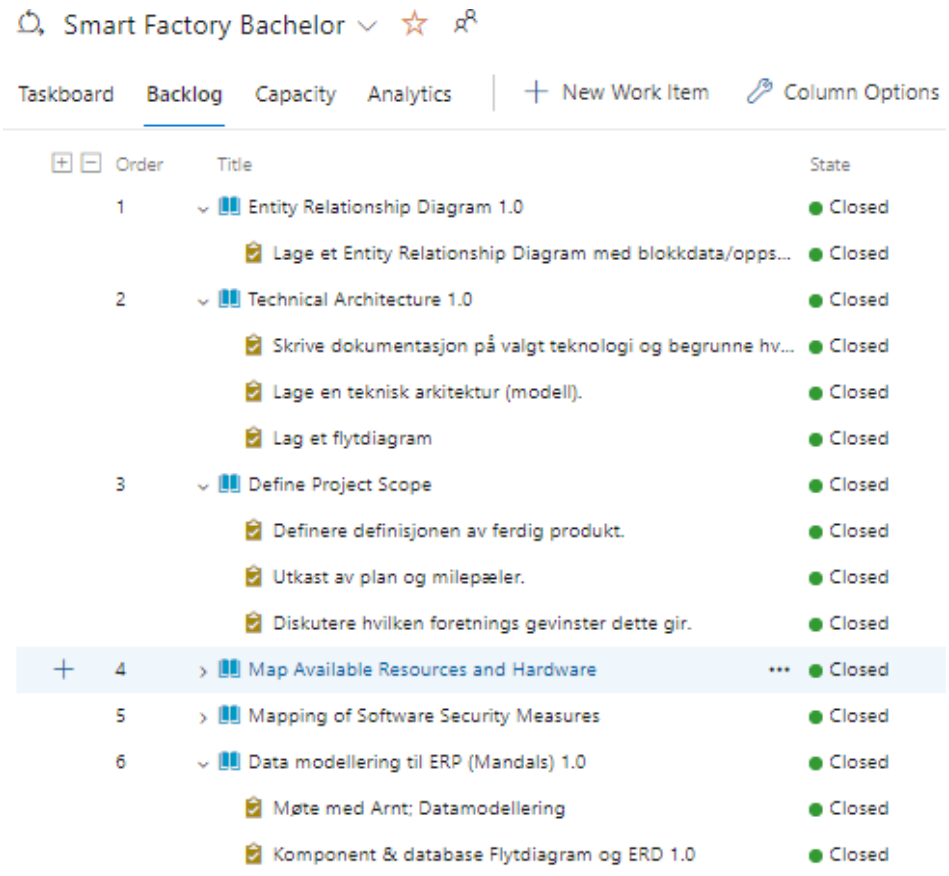
Figur 13 - Eksempel på brukerhistorie fra produkt backloggen

Figur 13 er et eksempel på en brukerhistorie fra produkt backloggen og viser hvordan vi benyttet prosjektstyringsverktøyet Azure DevOps i prosjektet. Dette eksempelet viser hvordan brukerhistorien kommuniserer mellom produkteier, utviklingsteamet og interessentene. Gjennom målene, funksjonene, kriteriene, test-scenarioene og tilhørende use-case, gir dette et

klart bilde av hva som skal utvikles, hvordan en oppnår ønsket resultat og hvordan funksjonaliteten skal testes for å sikre at systemet møter kravene. Dette var hvordan listen holdt struktur for å sikre kvalitet og orden innad i teamet. Produkt backloggen ga en helhetlig oversikt over alle funksjonelle og ikke-funksjonelle kravspesifikasjoner som skulle implementeres i produktet. På den måten kunne man evaluerer fremgangen av hele prosjektet under prosjektets levetid.

5.4.2 Sprint backlogg

Sprint backloggen var en samling av ulike elementer spesifikt valgt for den gjeldende sprinten. Dette inkluderte brukerhistorier og oppgaver som ble planlagt på sprint planning i starten av hver sprint. I starten av prosjektet hadde vi et overordnet overblikk over hvordan hele prosjektet skulle ta for seg, men siden vi valgte en agil tilnærming i prosjektet hadde vi fleksibilitet til å endre underveis. Sprint backloggen bidro til å skape en oversikt over oppgavene som måtte gjøres for å nå milepælen. Backloggen var grunnlaget for evalueringen av fremgangen under hver sprint.

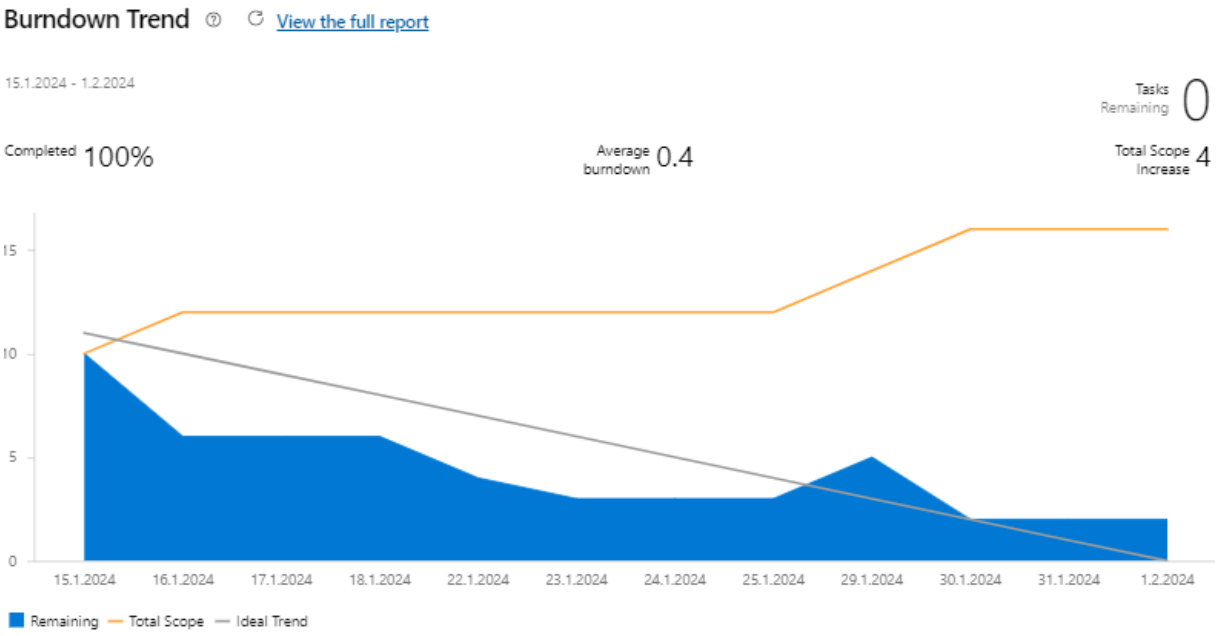


Figur 14 - Bilde av Sprint Backloggen som tilhører sprint 1

Figur 14 viser brukerhistoriene og tilhørende oppgaver for sprint 1, som er sprint backloggen for de gjeldene sprinten.

5.4.2 Burndown Chart

For å holde enkelt holde styr på hvordan sprintene utviklet seg ble det benyttet burndown chart i Azure DevOps. Dette var integrert i verktøyet som gjorde det enkelt å ta i bruk ved at når det ble lagt til nye oppgaver ble burndown charten oppdatert med estimatetene som var satt for oppgavene. Dette førte til at vi fikk en pekepinn på hvor vi var og hvor vi skulle ligge gjennom sprinten for å sikre at vi var på riktig vei.



Figur 15 - Burndownchart som viser fremgangen av en sprint

Figur 15 viser et utklipp fra Burndown charten i Azure DevOps som viser teamet et klart mål om arbeidsmengden som skulle leveres innen sprintens slutt. Verktøyet tillot for flere visualiseringer av burndown charten. Figur 15 viser antall oppgaver i sprinten, men det var også mulig å se på antall timer. Etter en oppgave ble fullført og flyttet til closed i DevOps, kunne man observere det blå området, som representerer gjenværende arbeidet, minsket dag for dag. Den grå linjen viser den ideelle fremgangen for å opprettholde passende flyt gjennom sprinten. Den oransje linjen viser totalt omfang av arbeid, som økte gjennom sprinten, noe som skjer når det legges til oppgaver i selve sprinten. I utgangspunktet skal ikke en sprint øke med antall oppgaver, men på grunn av prosjektets agile natur var dette nødvendig. De oppgavene som gjensto ved sprintens slutt, ble videreført til neste.

5.5 Kvalitetssikring

Hvert prosjekt søker alltid etter kvalitet, og det er ofte det som skiller et vellykket prosjekt i motsetning av et mislykket ett (Jakobsen, 2019). I vårt prosjekt ble kvalitetssikring innpasset fra start og ble en hjørnestein i prosjektstyringen.

Formålet med kvalitetssikring var det flere grunner til. Det tok for seg å sikre at alle leveranser hadde høy kvalitet, at brukernes behov var i samsvar med de kravene som ble forhåndsdefinert. Denne tilnærmingen til kvalitet la grunnlaget for tillitten hos oppdragsgiver og opprettholdt prosjektet sin integritet gjennom hele utviklingsprosessen. Særlig i dette feltet vi ga oss ut i, var detaljer av stor betydning og kvalitetssikring fungerte som vårt kompass som sikret at vi holdt oss i riktig retning.

Overordnet så ble det iverksatt en rekke strategiske kvalitetssikring aktiviteter. Disse var blant annet regelmessige tester for å identifisere og løse problemer effektivt, opprettholde høy kodekvalitet, dokumentering underveis, og kontinuerlig integrasjon av ny funksjonalitet for å validere endringene. Det ble etablert en god arbeidsflyt på starten av prosjektet, som gjordet det enkelt å overvåke kvaliteten på ulike stadier av utviklingen av systemet. Ved å gjøre dette på en slik måte kunne vi håndtere eventuelle avvik før de ble større problemer. Videre blir det tatt for hvordan suksess faktorene og suksess kriteriene ble oppfylt ved prosjektets slutt. Dette kan ses i Vedlegg B, evalueringsmøte.

5.5.1 Kvalitetssikrings prosesser

For å sikre kvalitet av prosjektet ble det implementert systematiske aktiviteter for å oppfylle produktets krav til kvalitet. Dette inkluderte bruk av Agile metoder, spesielt aktivt bruk av scrum rammeverket, som enkelt tilrettela for jevnlig evaluering og forbedring gjennom sprintene. Det var også et klart fokus å bryte ned prosjektet i deler som var håndterbare som enkelt kunne evalueres gjennom sprint reviews og retrospektive.

Ytterlige kvalitetssikringstiltak inkluderte:

- Sprint reviwes for å identifisere problemer og forberede tiltak relatert til produktet.
- Sprint retrospektiver for å identifisere problemer med prosjektgjennomførelsen og forberede tiltak.
- Evaluering av kodekvalitet gjennom hele utviklingsprosessen.
- Fokus på dokumentasjon og kommentarer i koden for å fremme felles forståelse og støtte overgangsprosesser.
- Kontinuerlig integrasjon av ny funksjonalitet for å sikre endringer, CI/CD
- Definering av hva som ble sett på som ferdig, for å sikre at leveranser møte de forhåndsdefinerte kravene og for å være mer effektiv.
- Produkteier tester funksjonalitet ofte for å fange opp feil eller mangler tidlig.

Suksessfaktorene som ble satt i starten av prosjektet som god kommunikasjon, tett oppfølgning med produkteier, trygt teammiljø, god teknisk forståelse og engasjement fra toppledelse, ble jevnlig evaluert, dette var det et større fokus på under sprint retrospektivene. Disse tiltakene som ble fastsatt bidro til at prosjektet til slutt oppfylte suksesskriteriene som å implementere alle «MUST HAVE» brukerhistorier, ingen av brukerhistoriene feilet i testscenariene, fullføre prosjektet innen fristen og ikke overstige budsjettet.

5.5.2 Risikoanalyse

En risikoanalyse er en analyse av risiko knyttet til aktiviteter. Det er også en måte for å få forståelse over hvilken hendelser som kan skje, hvor hendelsen skjer, hvorfor det kan skje, og konsekvensen av hendelsen (Aven, 2023).

I dette prosjektet er det flere aktiviteter som kunne defineres som en risiko (påvirke prosjektet eller produktet negativt), men gjerne relatert til forskjellige aspekter av prosjekter. Vi kategoriserte aktivitetene i fire grupper (kode, sikkerhet, prosjektstyring, og teknologi). Kategorien kode fremhevde aktiviteter relatert til måten man jobbet med kode. Kategorien teknologi fremhevde aktiviteter relatert til maskinvare. Kategorien sikkerhet fremhevde aktiviteter relatert til cybersikkerhet. Kategorien prosjektstyring fremhevde aktiviteter relatert til samarbeid.

For å få en oversikt over alle aktiviteten og dens risiko brukte vi en riskomatrise. Vi brukte en enkel risikomatrise med to dimensjoner, sannsynlighet og konsekvens. Det beskriver hvor stor sannsynlighet det er for at aktiviteter skjer og hvor stor konsekvens det medfører. Dette skaper tre forskjellige soner, se tabell 10, hvor ønskelig sone er grønn.

Sannsynlighet (X) Konsekvens (Y)	1	2	3
1	Lav (1)	Lav (2)	Medium (3)
2	Lav (2)	Medium (4)	Høy (6)
3	Medium (3)	Høy (6)	Høy (9)

Tabell 10 - 3X3 risikomatrise

Etter vi hadde identifisert alle aktivitetene evaluerte man aktiviteten ved å gi den en sannsynlighetsverdi og konsekvensverdi hvor man multipliserte verdiene. Summen av multiplikasjonen plasserer aktiviteten i riskomatrise. Etter initial verdi skrev vi ned tiltak for å enten redusere konsekvens eller sannsynlighet hvor målet var å redusere sluttverdien slik at aktiviteten ikke kunne plasseres i rød sone. Dette bidro til å få en oversikt over risikoen og hvilke tiltak man kunne bruke for å imøtekomme aktiviteten. Se tabell 11 og 12 for utdrag av risikoanalysen.

Kategori	Risiko	Sannsynlighet	Konsekvens	S*K
Sikkerhet	**IP Spoofing:** En angriper forfalsker med vilje en annen enhets identitet ved å endre pakkehodet med en forfalsket IP-adresse. I industrielle anlegg kan det være risikabelt hvis Ethernet/IP industriprotokollen brukes til kommunikasjon, som bruker IP-adresser for å identifisere enheter	3	3	9

Tabell 11 - Eksempel fra risikoanalyse del 1

Tiltak	Ny Sannsynlighet	Ny Konsekvens	NS * SK
VPN som forhindrer Spoofing og tjeneste nekt angrep. MQTT kommunikasjon som bruke digitale signaturer for data integritet (x.509 - sertifikat)	1	2	2

Tabell 12 - Eksempel fra risikoanalyse del 2

5.6 Tidsstyring

Prosjektplanlegging er ofte relatert til planlegging av aktiviteter for å oppnå et ferdig produkt. Tidsplanlegging handler om å definere datoer for når disse aktivitetene skal starte og ferdigstilles (Jan Terje Karlsen, 2021). Hoved hensikten til tidsplanlegging er å sikre at leveransen blir levert som avtalt. Andre viktig hensikter med tidsplanlegging er at man får et grunnlag for å evaluere fremgangen. Det gjør ressurser målbare som gir et bedre grunnlag for planlegging dersom det skjer endringer. Mulighet for å motivere prosjekt deltakere (Jan Terje Karlsen, 2021).

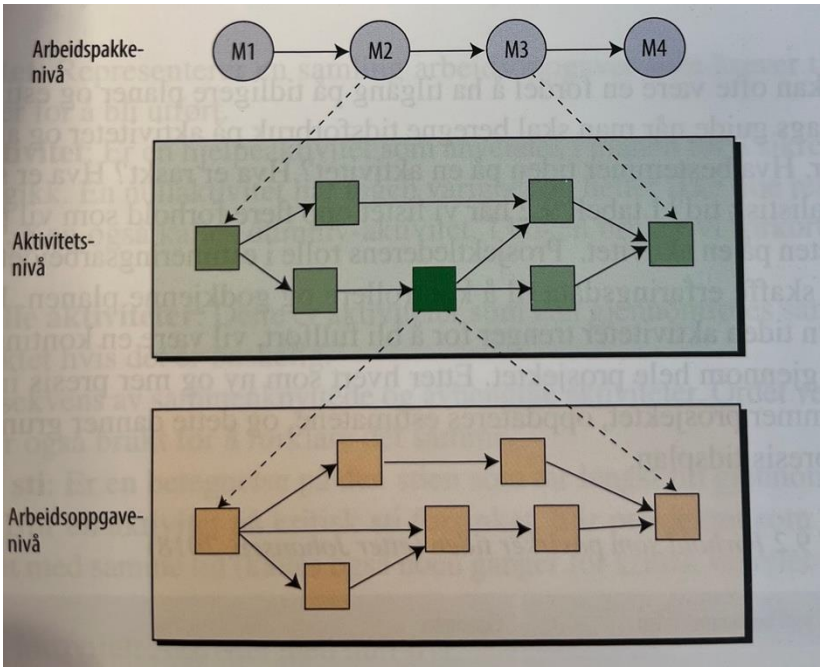
Det finnes flere planleggingsteknikker, men ett hvert prosjekt må tilpasses sine omgivelser. På bakgrunn av prosjektets agile natur var milepælplanlegging egnet. Det er en tidsplanlegging på et mer overordnet nivå. Milepæler blir brukt som en kontrollstasjon for man kan evaluere om prosjektet er i riktig kurs. Det innebærer ofte et overordnet mål man som prosjektet bør har oppnådd, i tillegg innebærer det ofte noen betingelser (Vibeke Edvardsen, 2011).

Prosjektets milepæl plan, se Vedlegg E, er tilknyttet til prosjektet sprinter slik at man kunne evaluere om man har nådd milepælen ved en sprint review. På grunn av prosjektets omgivelser, hadde man også mulighet til å omdefinere milepæler ved sprint planning, slik at man kunne møte endringer fra produkteier. I tillegg valgte man en overordnet tilnærming fordi man ønsket å redusere tiden på planlegging, med tanke på at tid er en knapp ressurs i dette prosjektet, men fortsatt ivareta fordelene med en systematisk fremgangsmetode, se Figur 3 prosjekttrekanten.

5.6.1 Milepæler, brukerhistorier og tid

Milepælene var et overordnet mål som man hadde tre arbeidsuker på å gjennomføre. Prosjektet har to prosjektdeltakere som har 66% tilgjengelig timer per uke hvor en uke er 37,5 timer. Dette tilsvarer 37,5 * 0.66 * 3 = 74,25 timer per deltaker. 74,25 * 2 = 148,5 timer. En sprint kan dermed ikke overstige 148,5 timer. Brukerhistoriene ble indikert med størrelsene small, medium, large og xtra large (S, M, L, XL), hvor S = 10 timer, M = 15, L = 20, og XL = 25. Defineringen av størrelse ble diskutert imellom oss hvor grad av kompleksitet var en viktig faktor for avgjørelsen, se Tabell 4 for eksempel av dette i en brukerhistorier. På denne måten kunne vi plassere brukerhistorier inn i sprintene og ha oversikt over tilgjengelig kapasitet.

Figur 16 som er laget av Jan Terje Karlsen gir en oversikt over hvordan man kan nedbryte en aktivitetsplan. På arbeidspakkenivå har man milepæler som består av flere aktiviteter. For prosjektet plasseres brukerhistorier på aktivitetsnivå. Summen av antall timer av nodene på aktivitetsnivå kan ikke overstige kapasitetsnivået til noden på arbeidspakkenivå. Se figur 14 for eksempel på nedbryting av brukerhistorie (fra aktivitetsnivå til arbeidsoppgavenivå). Her blir brukerhistoriene delt opp i arbeidsoppgaver i lavere nivåer, som er med håndterbare.



Figur 16 - Modell av en aktivitetsplan (Jan Terje Karlsen, 2021).

For å evaluere fremgangen innad i en sprint brukte vi Azure DevOps sin analytics fremvisning. Verktøyet fremviser en «brundownchart», se figur 15. Den fremhever den ideale trenden for å nå målet, og hva som gjenstår. På den måten kunne vi evaluere fremgangen for hver sprint under det ukentlige stand møte, og iverksette tiltak ved behov.

Etter en gjennomgang av prosjektgjennomføringen i dette kapittelet, hvor vi gikk gjennom vår agile praksis og kvalitetssikringsmetoder, reflekteres prosjektets resultater i det neste kapittelet. Her vil det bli tatt et tilbakeblikk på hvordan målene til prosjektet ble møtt og hva vi kan trekke fra denne erfaringer.

6. Refleksjon

I denne refleksjonsdelen av bacheloroppgaven vil vi se på og evaluere de metodene og tilnærmingene som har vært mest sentrale i prosjektet. Her vil vi diskutere hvordan den agile arbeidsmetodikken har hatt innvirkning på prosjektets fremgang og kvalitet, og hvordan dette har bidratt til å håndtere usikkerheter som oppstod fra teknologi som vi ikke var kjent til skiftende kravspesifikasjoner. Vi vil reflektere over de beslutningene som drev prosjektet fremover og de punktene vi tar med oss videre, noe som ga innsikt i utfordringene vi møtte og suksessene som ble oppnådd.

6.1 Evaluering av metoder og teknikker

Det viktigste for dette prosjektet har vært valget av en agil arbeidsmetodikk. Som nevnt tidligere er en agil arbeidsmetodikk egnet for prosjekter med en grad av usikkerhet. Usikkerhet kommer fra manglende informasjon (Rolstadås, 2023c), og for dette prosjektet stammet usikkerheten fra ukjent teknologi og skiftene kravspesifikasjoner. En agil tilnærming senket skuldrene for å foreta valg fordi det var mulighet til å endre på valgene senere i prosjektets livsløp. Dette har bidratt til raskere fremgang i prosjektet. Uten denne arbeidsmetodikken er vi usikre på at vi hadde oppnådd samme resultat fordi usikkerheten hadde hindret oss i å ta valg som hadde hindret fremgangen.

På den andre siden øker det sannsynligheten for valg som ikke er best egnet. Derfor har fokus på kvalitetssikring av produktet gjennom kontinuerlige tester og leveringer vært avgjørende for kvaliteten. Dette har tillatt for å oppdage feil eller oppgraderinger tidlig slik at man kan foreta tiltak. I tillegg tilrettela det for bedre kommunikasjon mellom utviklere og produkteiere som hindret at man leverte store leveranser som ikke var slik produkteier ønsket. Dette ble oppnådd siden den agile tilnærmingen bruker sprinter.

Noen valg er mer faste enn andre hvor det er vanskeligere å endre valget senere i prosjektets livsløp. For dette prosjektet er eksempler på dette valg av maskinvare, programvare, og arkitektur. Normalt sett er agil arbeidsmetodikk iterativt hvor man gjennomfører analyse-, design-, utvikling- og testfasen for hver iterasjon. Hver iterasjon trenger nødvendigvis ikke være lik. Dette prosjektet sin initiale iterasjon bestod bare av analyse og design. Dette var på bakgrunn at det var her vi var avhengig av å ta viktig valg som komt til å påvirke produktet og prosjektet under hele livsløpet. Dermed brukte man mye mer tid på å senke usikkerheten for valgene i motsetning til valg eller endringer tatt seinere i livsløpet. For å forsikre at man hadde tatt hensyn til alle risikoene var en risikoanalyse veldig nyttig. Det ga en oversikt over flere risikoer og hvor mye det komt til å påvirke prosjektet og tiltak for å redusere effekten. Dette var med på å danne et solid grunnlag som man kunne jobbe videre på. Det bidro også med å senke skuldrene for å ta valg fordi man hadde et grundig forarbeid. Vi sammenligner det med å utruste seg best mulig før man velger å gå til kamp.

6.2 Kvalitetssikring

Som tidligere nevnt er kvalitetssikring systematiske aktiviteter som skal sikre at leveranser oppfyller kravene av kvalitet (Halbo, 2023b). Dermed er det viktig at man definere hva kvalitet er. Kvalitet kan være vanskelig å definere fordi det er avhengig av hvor produktet eller tjenesten brukes. For eksempel kan en utvikler definere kvalitet som koden er optimalisert for ytelse og fri for bugs, mens for en produkteier, kan kvalitet defineres som at funksjonalitet møter brukerens behov og forretningsmål. Derfor er det viktig å definere kvalitet slik at det imøtekommer omgivelsene sine og ha systematiske aktiviteter som tilrettelegger for at leveransen blir definert som kvalitet av alle deltakere.

Definisjonen av kvalitet for scrum master var at prosjektet hadde en fremgang og at omgivelsene var tilrettelagt for andre prosjektdeltakere. Ved å aktivt bruke scrum rammeverket gjennom scrum artefaktene og scrum aktiviteten kunne scrum master evaluere prosjektets kvalitet. Slik at man kan tilpasse bruken av scrum rammeverket for bedre resultat. Eksempel på endringer var gjennomførelsen av retrospektiv hvor vi brukte Azure DevOps retrospektive verktøyet ved første sprint, se Vedlegg F. Verktøyet og de predefinerte spørsmålene bidro mer som en hindring til åpen dialog, dermed kuttet vi ut verktøyet og de predefinerte spørsmålene og fokuserte mer på åpen dialog og suksess faktorene.

Definisjonen av kvalitet for produkteier var når brukerhistorier sine funksjonelle og ikke-funksjonelle krav var blitt implementert. Definisjonen av kvalitet for produkteier og prosjektdeltaker er liknende, men ofte tolker produkteier og utvikler de funksjonelle og ikke-funksjonell kravene forskjellig og i denne avgjørelsen er det produkteier som har siste ord. Derfor er det viktig med god dialog mellom produkteier og utvikler. I tillegg at produkteier tester kontinuerlig. Dette ble gjennomført gjennom ukentlig møter hvor prosjektdeltakere gikk gjennom hva som hadde blitt gjort hvor produkteier kunne teste og evaluere brukerhistoriene. Dersom produkteier ikke var enig, fremhevde produkteier manglene. Brukerhistorien endret så tilstand fra «closed» til «aktiv» og man jobbet videre. Ved sprint review gikk man over alle brukerhistoriene en gang til. Kravspesifikasjoner fra produkteier kan endre seg iløpet av prosjektet, dermed var det viktig med flere evalueringer.

Definisjonen av kvalitet for en prosjektdeltaker var også når brukerhistorien sine funksjonelle og ikke-funksjonelle krav var blitt implementert. Definisjonen av ferdig var derimot forskjellig. prosjektdeltakere kan løse problemstillinger på flere forskjellige måter, men for å sikre effektivt samarbeid og tilrettelegge for videre utvikling har vi definert noen krav. For å sikre de funksjonelle og ikke-funksjonelle kravene skulle ingen test-scenario feile.

For å sikre effektivt samarbeid fulgte man god «git-hygiene» som inkluderte kode-reviews og beskrivende meldinger i «git-comits». Dette bidro til at man skjønte hensikten med koden under kode-reviewsene. Det var også fokus på forklarende kommentarer av koden. Koden måtte også bli godkjent før det ble implementert i hoved-branchen. Kommentarene bidro til at man enkelt kan sette seg inn i koden og vil være viktigere jo større kodebasen blir. Reviewsene skapte en plattform for å sikre at koden var av høy kvalitet og at kommentarene

var godt skrevet. Dette bidro til at kvalitet bevarte seg. Samtidig som det var fokus på kommentarer i koden, forklarte det bare hensikten med små kodefragmenter. Dermed var det fokus på dokumentasjon som dekket større deler, som for eksempel databasen eller hvordan man setter opp systemet, se Vedlegg D. Det var viktig slik at man kunne se på dokumentasjonen dersom man var usikker, og for at nye utviklere kunne enkelt sette seg inn i hele systemet.

For at å sikre at man møtte kvalitet iht. omgivelsene sine har det vært viktig å etablere en felles forståelse og sette riktig styringsfaktorer for prosjektet. Hvordan tar man riktig beslutning som sikrer best kvalitet for prosjektet? For å ta beslutninger har prosjekttrekanten, se Figur 3, vært en god ressurs som har bidratt til å ta viktige valg som hvordan vi skal prioritere brukerhistorier eller hvilke maskinvare og programvare som skal brukes. I tillegg har prioritering av design kriterier, se Tabell 9, også bidratt mye. Dette la til grunn for hvordan vi helt fra starten tok de riktige valgene.

6.3 Samarbeid

For å sikre kontinuerlig oppdateringer av teamets arbeid mellom teammedlemmene, ble det valgt å ha ukentlige møter. De ukentlige møtene rustet for dypere diskusjoner om fremgangen i prosjektets og de overordnede målene, dette viste seg å være avgjørende for å ha momentum i prosjektet. Under disse møtene benyttet vi Azure DevOps som visuell støtte, som hjalp med å holde oversikt over oppgaver og fremdrift uke til uke, uten dette verktøyet ville møtene vært uoversiktlig og krevende med å holde styr på progresjon og oppgaver.

Prosjektet krevde mye tilpasning i hvordan vi planla og utførte oppgaver underveis. Evnen for å tilpasse oppgaver etter behov under prosjektet har vist seg å være uvurderlig, spesielt når vi kom over forhold hvor vi trengte å omprioritere oppgaver for å møte kravene, dette var viktig ettersom vi har stått litt fritt til å velge hvordan systemet skulle være. Samtidig skulle det være innenfor de rammene de hadde tenkt seg, dette ble jevnlig diskutert og tilpasset, og ga en fin fleksibilitet på arbeidet vi gjorde. Daglige stand-ups var også tiltenkt å ha, ble mindre prioritet, da det ble sett på som unødvendig og mer tidskrevende enn det ga tilbake ettersom vi jobbet tett sammen gjennom hele prosjektet og oppdaterte hverandre jevnlig med hva som hadde blitt gjort og ikke. Gjennom prosjektet har det vært fokus på at en har ansvar under frihet, ettersom det ble etablert tillit fra starten av.

Som tidligere nevnt ble det definert suksessfaktorer som skulle til for å skape trygge og gode rammer for teamet. Sett tilbake på gjennomføringen av prosjektet vil vi si at disse ble overholdt om ikke overgått fra forventingen. Kommunikasjonen og tilliten som ble etablert og skapt, oppmuntret til en kultur hvor alle kunne uttrykke sine meninger og ideer fritt. Utenom å styrke teamets forhold, førte dette til mer innovative løsninger og beslutningstaking.

6.4 Læring og personlig vekst

Prosjektet har gitt oss en reise med både faglig og personlig utvikling. Verden vi ble møtt av besto av IT og OT og hvordan disse kunne integreres i en industrielle kontekster var nytt, men tidligere teori og praksiser forberedte oss for å overkomme de største utfordringene. Gjennom praktisk og teoretisk anvendelse av agile metodikker, som scrum og DevOps, ble det enklere å ta til seg ny tekniske kompetanse og forståelse av disse to verdene. Alt fra programmering av mikrokontrollere til å innhente informasjon som skulle brukes til datamodelleringen.

Vi opplevde en betydelig utvikling ettersom vi jobbet i et profesjonelt team, som viste engasjement og faglig kompetanse under prosjektets livssyklus. Noe av det viktigste vi kommer til å ta med videre er fra planlegging til implementering, hvordan vi vurderer og anvender kvalitet systematisk for å forbedre prosjektresultatene. Arbeidet sammen med veiledere og ansatte hos Knowit har også lært oss verdien av tverrfaglig samarbeid. Dette har understreket hvor viktig det er å dele kunnskap og erfaringer, noe som sentralt for suksessen av prosjektet.

7. Konklusjon

Dette prosjektet har vist verdiene av konvergensen mellom IT og OT, i en industriell kontekst. Og hva som kan skapes og fremmer mulighetsrommet og de utfordringene som følger med. Dette er et relevant tema og kommer til å bli mer fokus på i fremtiden av «industri 4.0». Sanntidsdata fra OT i kombinasjon med IT tilrettelegger for økt effektivisering, eksempelvis gjennom analyse av data eller automatisering av arbeidsoppgaver. Ved at man har tilgjengelig data å analysere kan datamaskiner oppdage mønstre som mennesker ikke klarer å oppfatte. Det skaper også muligheten for å bruke KI for å analysere og oppdage avvik.

Automatisering av arbeidsoppgaver frigjør ressurser slik at det kan bli brukt hvor det er mer behov. Eksempelvis er dette prosjektet hvor systemet fjerner muligheten for menneskelig feil av produksjonsstart og produksjonsstopp. Systemet skaper mulighet for å bruke de menneskelige ressursene sine mer konstandseffektivt, samtidig som det hindrer overforbruk av materielle ressurser som også påvirker miljøet negativt.

Dataen som blir tilgjengelig gjennom konvergensen, og systemets tilpasningsdyktighet gjør at man enklere kan imøtekomme omgivelsene sine. Fremtiden bærer preg av høyere fokus på bærekraft hvor det blir strengere krav om bærekrafts rapportering (*Bærekraftsrapportering for store bedrifter | europalov*, 2021). Konvergensen tilrettelegger for at man har dataen tilgjengelig til å møte kravene.

For videre utvikling av systemet vil optimalisering av databasen og fokus på sikkerhet være viktig. Optimalisering er avhengig av konteksten databasen er i. Databasen bør optimaliseres ved å implementere regler for massesletting og masseoppdateringer. For å optimalisere søk anbefales bruk av indekser. Siden databasen er en flerbrukerdatabase er det viktig å implementere begrensninger (låse mekanismer). Et eksempel på dette er dersom to klienter sjekker tilgjengelig ressurser samtidig. Tilgjengelige ressurser bør være oppdatert før neste transaksjon har skjedd.

For implementering av et slikt system er det ikke hensiktsmessig å ha en kun en lokal database på grunn av mengden data. Ettersom dette prosjektet har til hensikt å være en PoC, er en lokal database mest hensiktsmessig, men vi anbefaler å bruke timescaledb, som er en DBaaS (DataBase as a Service) for å løse dette problemet som er diskutert tidligere, se delkapittel 4.5.1. Dette tillater for sikkerhetskopiering av dataen slik at mengden data lokalt ikke overstiger minnekapasiteten til edge-enheter.

For sikkerhet er det tilrettelagt for x509 sertifikater, men som tidligere nevnt er Arduino Uno Rev 4 en flaskehals for dette. For at Pub/Sub kommunikasjonen skal bruke sertifikatene er man avhengig av at alle noder bruker det. Dermed bør man undersøke nærmere på løsninger for å bruke det enten på en Arduino Uno Rev 4 m/WiFi eller vurdere å bytte til en mikrokontroller som er kompatibel med MQTT kommunikasjon med sertifikat.

Alt i alt har prosjektet vært spennende, utfordrende både teknisk og organisatorisk, veldig lærerikt, og ikke minst gøy. For å evaluere om prosjektet er et vellykket eller mislykket prosjekt referer vi til suksesskriteriene, se delkapittel 2.3.1. Suksesskriteriene er noe vi har jobbet aktivt mot og vært et mål under prosjektets levetid. Sluttproduktet har implementert alle brukerhistorier kategorisert som «MUST HAVE» hvor ingen test scenarioer feiler. Produktet ble levert innenfor planlagt tidsramme, og oversteg ikke budsjettet. Kunde kvalifiserer dette som et PoC og utrykker at de er fornøyde. De kvalifiser kodekvaliteten som høy og bekrefter at de kan sette seg inn koden og videreutvikle.

Et vellykket prosjekt kan resultere i et godt produkt og en fornøyd kunde, men at det ikke skaper merverdi for kunden. Eksempelvis, en bestilling på en ny hjemmeside hvor kunden er fornøyd, men det resulterer ikke i økt antall besøkende på hjemmesiden deres. Derfor ønsket vi å skape merverdi for Knowit. Etter siste evaluering møte med Knowit fikk vi bekreftelse på nettopp dette. Prosjektet har skapt grunnlaget for tilsvarende prosjekter i fremtiden og skapt interesse både internt og eksternt for Knowit. Evalueringsmøte og uttalelsene av oppdragsgiver ligger som vedlegg, se Vedlegg B - rapport av evalueringsmøte, og Vedlegg I - Uttalelse av Oppdragsgiver. Med dette i betraktning anser vi (prosjektdeltakere) og Knowit (oppdragsgiver) prosjektet som vellykket.

8. Litteraturliste

12 Principles Behind the Agile Manifesto | Agile Alliance. (2015, november 4). https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/

Andersen, E. S. (2016, mars 2). *Åtte bud for å lykkes med prosjekter*. BI Business Review. https://www.bi.no/forskning/business-review/articles/2016/03/atte-bud-for-a-lykkes-med-prosjekter/

Atlassian. (u.å.-a). *Learn about Agile Scrum Artifacts*. Atlassian. Hentet 7. mai 2024, fra https://www.atlassian.com/agile/scrum/artifacts

Atlassian. (u.å.-b). *Product Backlog Explained [+ Examples]*. Atlassian. Hentet 7. mai 2024, fra https://www.atlassian.com/agile/scrum/backlogs

Atlassian. (u.å.-c). *What is Agile?* Atlassian. Hentet 20. februar 2024, fra https://www.atlassian.com/agile

Aven, T. (2023). Risikoanalyse. I *Store norske leksikon*. https://snl.no/risikoanalyse

Bjørn Kristoffersen. (2021). *Databasesystemer* (5. utgave). Universitetsforlaget AS.

Bærekraftsrapportering for store bedrifter | europalov. (2021, april 22). https://europalov.no/rettsakt/baerekraftsrapportering-for-store-bedrifter/id-29041

Chen, C., Tock, Y., & Girdzijauskas, S. (2018). *BeaConvey: Co-Design of Overlay and Routing for Topic-basedPublish/Subscribe on Small-World Networks*. https://doi.org/10.1145/3210284.3210287

Computer Network | Client and Server Model—Javatpoint. (u.å.). Www.Javatpoint.Com. Hentet 15. mars 2024, fra https://www.javatpoint.com/computer-network-client-and-server-model

David J. Barnes & Michael Kölling. (2016). *Objects First with Java—A Practical Introduction Using BlueJ* (6. utg.). Pearson Education Limited 2013.

Dennis, A., Wixom, B., & Tegarden, D. (2015). *Systems Analysis and Design: An Object-Oriented Approach with UML*. Wiley. https://books.google.no/books?id=rbLrBgAAQBAJ

Dias, J., Restivo, A., & Ferreira, H. (2022). *Designing and constructing internet-of-Things systems: An overview of the ecosystem*. https://doi.org/10.1016/j.iot.2022.100529

Ehie, I. C., & Chilton, M. A. (2020). *Understanding the influence of IT/OT Convergence on the adoption of Internet of Things (IoT) in manufacturing organizations: An empirical investigation*. *115*(103166). https://doi.org/10.1016/j.compind.2019.103166

Ellis, G. (2016). Chapter 8—Agile Project Management: Scrum, eXtreme Programming, and Scrumban. I *Project Management in Product Development*. http://dx.doi.org/10.1016/B978-0-12-802322-8.00008-5

Embracing Uncertainty: Rethinking Project Timelines in Agile Environments | Scrum.org. (u.å.). Hentet 1. mai 2024, fra https://www.proscrumtraining.com/embracing-uncertainty-rethinking-project-timelines-in-agile-environments/

Forskningsrådet. (2022, mai). *Fylkesvise kunnskapsgrunnlag*. https://www.forskningsradet.no/tall-analyse/forskning-innovasjon/fylkesvise-kunnskapsgrunnlag/

Frederik Borgersen. (2018). *Håndtering av risiko i digitaliseringsprosjekter—En studie av fallgruver og suksessfaktorer*. Universitetet i Stavanger. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://uis.brage.unit.no/uis-xmlui/bitstream/handle/11250/2565820/Borgersen_Frederik.pdf?sequence=1&isAllowed=y

Halbo, L. (2023a). Kvalitetssikring. I *Store norske leksikon*. https://snl.no/kvalitetssikring

Halbo, L. (2023b). Kvalitetssikring. I *Store norske leksikon*. https://snl.no/kvalitetssikring

How Is OT Different From IT? OT vs. IT. (u.å.). Cisco. Hentet 20. februar 2024, fra https://www.cisco.com/c/en/us/solutions/internet-of-things/what-is-ot-vs-it.html

Hva er Unified Namespace (UNS)? (u.å.). *Novotek Norge*. Hentet 1. mai 2024, fra https://www.novotek.no/insight/hva-er-unified-namespace-uns/

Implementing Unified Namespace (UNS) With MQTT Sparkplug. (2022, desember 20). https://www.hivemq.com/blog/implementing-unified-namespace-uns-mqtt-sparkplug/

Jakobsen, E. M. (2019, januar 9). Hva er et vellykket prosjekt? • ADIRE. *ADIRE*. https://www.adire.no/hva-er-et-vellykket-prosjekt/

Jan Terje Karlsen. (2021). *Prosjektledelse—Fra initiering til gevinstrealisering* (5. utg.). Universitetsforlaget AS.

Kashyap, M., Sharma, V., & Gupta, N. (2018). *Taking MQTT and NodeMcu to IOT: Communication in Internet of Things*. *132*. https://doi.org/10.1016/j.procs.2018.05.126

Kristoffersen, B. (2021). *Databasesystemer* (5. utg.). Universitetsforlaget.

Kumar, P., Raouf, I., & Soo Kim, H. (2023). *Review on prognostics and health management in smart factory: From conventional to deep learning perspectives*. *107126*. https://doi.org/10.1016/j.engappai.2023.107126

Lage, forstå og presentere statistikk—Diagrammer—Medie- og informasjonskunnskap 1—NDLA - NDLA. (u.å.). ndla.no. Hentet 3. mai 2024, fra https://ndla.no/nb/subject:1:058bdbdb-aa5a-4a29-88fb-45e664999417/topic:1:537598a2-4857-40e0-b0bc-9a937e954374/topic:1:5ea3737b-e963-4563-ad07-791da6e21064/resource:1:178906/4108

Linux. (2024). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Linux&oldid=1208801826

Manifesto for Agile Software Development. (u.å.). Hentet 14. mars 2024, fra https://agilemanifesto.org/

MariaDB vs PostgreSQL - Difference Between Open-Source Relational Databases—AWS. (u.å.). Amazon Web Services, Inc. Hentet 21. februar 2024, fra https://aws.amazon.com/compare/the-difference-between-mariadb-and-postgresql/

Mathiassen, L., Munk-Madsen, A., Nielsen, P. A., & Stage, J. (2018). *Object Oriented Analysis & Design*.

Nema 23 23hs5628 57 Motor 2.8a Med Tb6600 Driver Nema17 23 For CNC ... (u.å.). Fruugo. Hentet 22. februar 2024, fra https://www.fruugonorge.com/nema-23-23hs5628-57-motor-28a-med-tb6600-driver-nema17-23-for-cnc-og-3d-skriver-del-del-deler-635m/p-155782458-329985672?language=no&ac=ProductCasterAPI&asc=pmax&gad_source=1&gclid=CjwKCAiA75itBhA6EiwAkho9excsDYvpM1fGaqWWO_iP_G1rTup8DdCWiL2-s_Rjnr1E62_gU3bUVBoCAu8QAvD_BwE

New And Original Incremental Rotary Encoder Lpd3806-600bm-g5-24c Ab Two Phase 600ppr Diameter 38mm Shaft 6mm—Buy Lpd3806-600bm-g5-24c,Pulse Encoder,Cheap Price Encoder Product on Alibaba.com. (u.å.). Hentet 22. februar 2024, fra https://www.alibaba.com/product-detail/new-and-original-incremental-rotary-encoder_60729422083.html

Node-RED. (u.å.). Hentet 20. februar 2024, fra https://nodered.org/

Object-Oriented Analysis and Design | OOAD. (2020, april 7). *GeeksforGeeks*. https://www.geeksforgeeks.org/object-oriented-analysis-and-design/

Om Knowit | Digitaliseringskonsulenter. (u.å.). Hentet 22. februar 2024, fra https://www.knowit.no/om-knowit/

Peer-to-Peer Network Meaning & Definition. (2024, februar 8). Brave. https://brave.com/glossary/peer-to-peer/

PostgreSQL vs MySQL - Difference Between Relational Database Management Systems (RDBMS)—AWS. (u.å.). Amazon Web Services, Inc. Hentet 21. februar 2024, fra https://aws.amazon.com/compare/the-difference-between-mysql-vs-postgresql/

Prosjekttrekanten—Støtte for Microsoft. (u.å.). Hentet 19. januar 2024, fra https://support.microsoft.com/nb-no/office/prosjekttrekanten-8c892e06-d761-4d40-8e1f-17b33fdcf810

Raspberry Pi Ltd. (u.å.). *Buy a Raspberry Pi 4 Model B*. Raspberry Pi. Hentet 22. februar 2024, fra https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

Rolstadås, A. (2020). Prosjektstyring. I *Store norske leksikon*. https://snl.no/prosjektstyring

Rolstadås, A. (2023a). Suksessfaktor – prosjektledelse. I *Store norske leksikon*. https://snl.no/suksessfaktor_-_prosjektledelse

Rolstadås, A. (2023b). Suksesskriterium. I *Store norske leksikon*. https://snl.no/suksesskriterium

Rolstadås, A. (2023c). Usikkerhet – prosjektledelse. I *Store norske leksikon*. https://snl.no/usikkerhet_-_prosjektledelse

Timescale Products. (u.å.). Hentet 29. april 2024, fra https://www.timescale.com/products

tzsoftno. (2020, oktober 5). *Tegn på at du bør investere i tilpasset programvare – TZ Soft*. https://tzsoft.no/2020/10/05/tegn-pa-at-du-bor-investere-i-tilpasset-programvare/

UNO R4 WiFi | Arduino Documentation. (u.å.). Hentet 22. februar 2024, fra https://docs.arduino.cc/hardware/uno-r4-wifi/

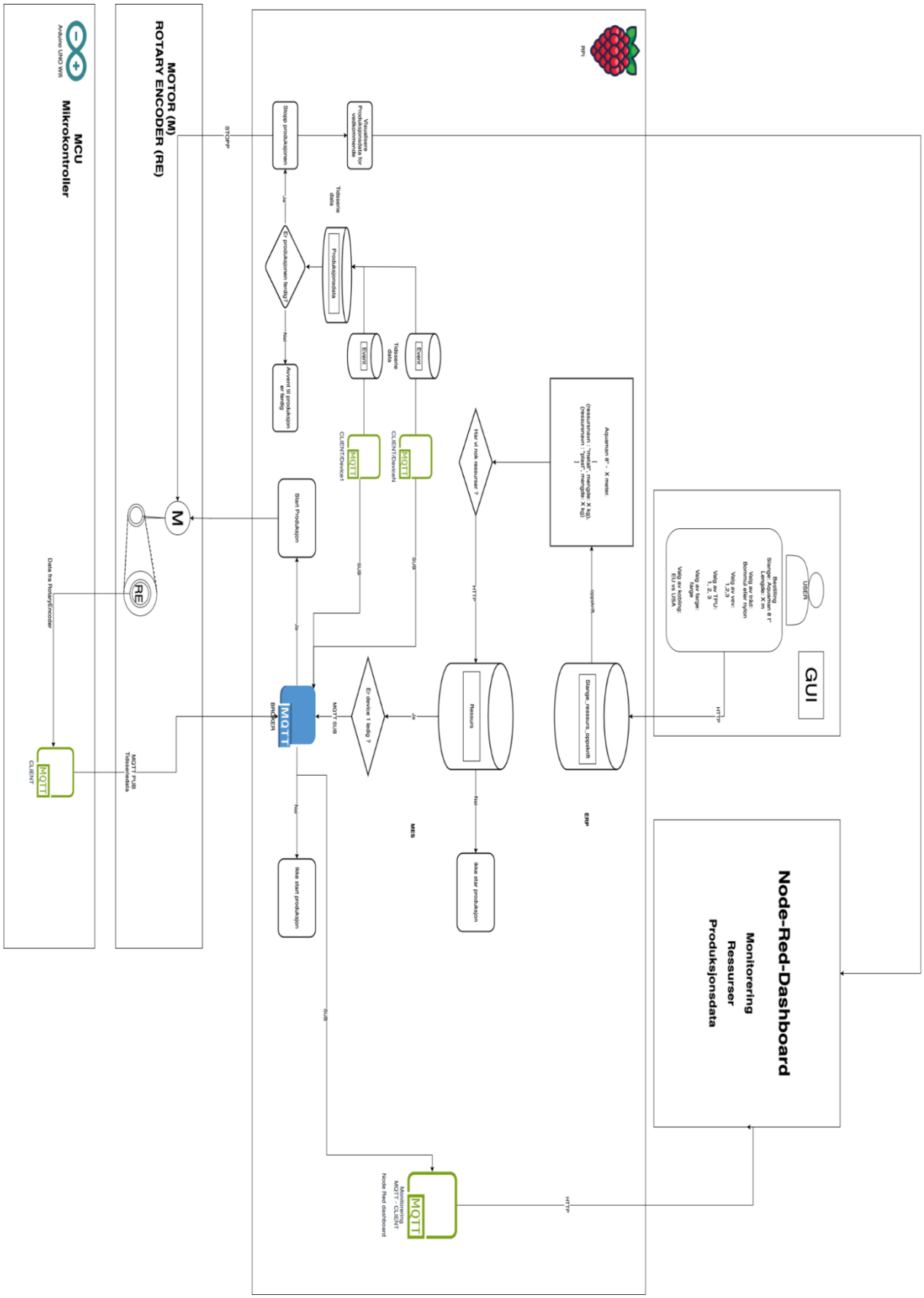
Vibeke Edvardsen. (2011). *Prosjektledelse* [Masteroppgave]. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://uia.brage.unit.no/uia-xmlui/bitstream/handle/11250/135624/BE-501-1%202011%20vsr%20Masteroppgave%20Vibeke%20Edvadsen.pdf?sequence=1

What is a Microcontroller and How Does it Work? (u.å.). IoT Agenda. Hentet 20. februar 2024, fra https://www.techtarget.com/iotagenda/definition/microcontroller

What Is IT/OT Convergence? (u.å.). Palo Alto Networks. Hentet 22. februar 2024, fra https://www.paloaltonetworks.com/cyberpedia/what-is-it-ot-convergence

9. Vedlegg

Vedlegg A – Teknisk arkitektur og informasjonsflyt



Vedlegg B – Rapport fra evalueringsmøte

Suksess Faktorer:

- God og klar kommunikasjon.
    - Spørsmål: Under prosjektets levetid, mener dere at vi har kommunisert godt, klart og tydelig.
    - Svar: Klart og tydelig, selvstendighet og sparring mellom avdelingen. Kunne ikke blitt bedre.
- Tett oppfølging med produkt eiere.
    - Spørsmål: Har vi oppdatert produkt eier med fremdrift og gjort justeringer etter tilbakemelding?
    - Svar: Dette ble gjort på en veldig fin og avrundende måte, hvor justeringene ble etterkommet og møtt.
- Et trygt og godt teamforhold.
    - Spørsmål: Har dere vært usikre på om vi skal nå målene dere har satt for oss?
    - Svar: Har ikke vært usikre på at vi kommer til å nå de målene vi har satt oss
    - Spørsmål: Har dere vært komfortable med å gi oss ansvar.
    - Svar: De har sett at vi har vært selvgående, hatt en indre motor og gitt oss ansvar utover det som har vært tiltenkt
- God forståelse over det tekniske.
    - Spørsmål: Har dere inntrykk over at vi har god forståelse over det tekniske og kan forsvare design valg og tekniske valg under prosjektets levetid.
    - Svar: Forsvart de tekniske valg på en bra måte, klar å tilpasse oss rask på en fin måte. Tatt selvstendige valg som de mener har vært fornuftige.

- Engasjement fra toppledelsen.

  Vi har et inntrykk over at samtlige deltagere har vært engasjerte.
    - Spørsmål: Syntes dere at prosjektet har vært spennende?
    - Svar: Uten tvil, vi har hatt engasjement som har gjort at prosjektet har nådd tidsfrister og jobbet agilt som har bidratt til en fin kompetanse i avdelingen
    - Spørsmål: Har prosjektet skapt engasjement fra andre interne interessenter i knowIt?
    - Svar: Kjørt kompetanse i avdelingen, vist frem til andre interessenter som da er involvert i som kunde.

Suksess Kriterier:

- 100% av brukerhistorier kategorisert som «MUST HAVE» er implementert.
    - **Oppfylt**/ikke oppfylt

- Ingen testscenario relatert til brukerhistorier kategorisert som «MUST HAVE» feiler.
    - **Oppfylt**/ikke oppfylt

- Prosjektet er ferdigstilt innen fristen 26.04.2024.
    - **Oppfylt**/ikke oppfylt

- Prosjektet overstiger ikke budsjettet.
    - **Oppfylt**/ikke oppfylt
    - **Kommentar:** De sier at det har vært en god investering i prosjektet og oss som studenter, der de har fått igjen mer enn de har forventet og brukt av deres ressurser

- Produktet skal kunne kvalifiseres som et PoC.
    - Spørsmål: Kan dette prosjektet kvalifiseres som et Proof of Concept?
    - Svar: **Ja**
    - Spørsmål: Skaper prosjektet vårt verdi for dere?
    - Svar: **Ja**

- Kodekvalitet kan kvalifiseres som høy.
    - Spørsmål: Hvordan vil dere kvalifisere kode-språket.
    - Svar: De mener vi har overholdt en god struktur i koden, men savner litt dokumentasjon, men det blir gjennomført forløpende
    - Spørsmål: Kunne dere satt dere inn i koden kjapt?
    - Svar: Vil ha litt mer wiki, en oversikt i git, ellers veldig bra

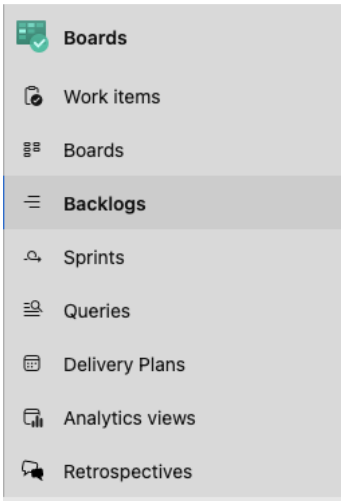
- Produkteier er fornøyd med produktet.
    - Spørsmål: Hvordan vil dere evaluerer produktet fra en skala fra 1-5 hvor; (1) = ikke bra nok, (2) = under forventet, (3) = forventet, (4) = Bra , (5) = Veldig Bra
    - Trygve: 5
    - Kristoffer: 5
    - Arnt: 5
    - Gjennomsnitt: 5

    - Spørsmål: Føler dere at dere har nok informasjon til å ta over prosjektet?
    - Diskusjon: Når det har blitt gjort en dokumentasjon, mener de på at overføringen vil bli bra, ettersom vi har hatt leveranser tidligere hvor de har sett at vi overkommer forventningene

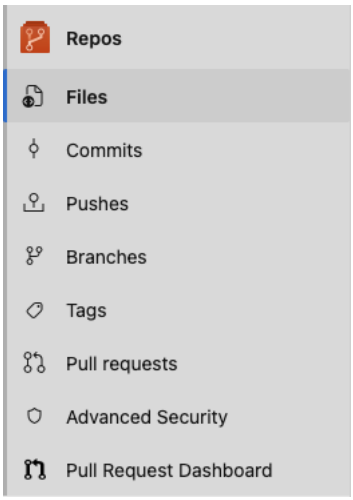
- Avsluttende Spørsmål:
    - Har dere noe dere ønsker å si helt til slutt?
    - Svar: Alt av funksjonelt er 100%, navn og datamodellering kunne vært bedre, men bra nok i prosjektet. Kunne ikke vært mer fornøyd med hvordan dette prosjektet ble løst

Vedlegg C – Azure DevOps

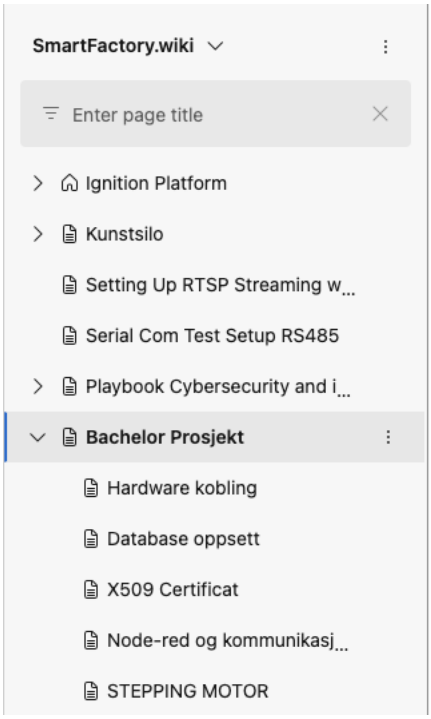
C.1 - Boards



C.2 – Repo



C.3 – Wiki


SmartFactory.wiki
Enter page title
Ignition Platform
Kunstsilo
Setting Up RTSP Streaming w...
Serial Com Test Setup RS485
Playbook Cybersecurity and i...
Bachelor Prosjekt
Hardware kobling
Database oppsett
X509 Certificat
Node-red og kommunikasj...
STEPPING MOTOR

Vedlegg D – Mini Smart Factory Lab – Manual

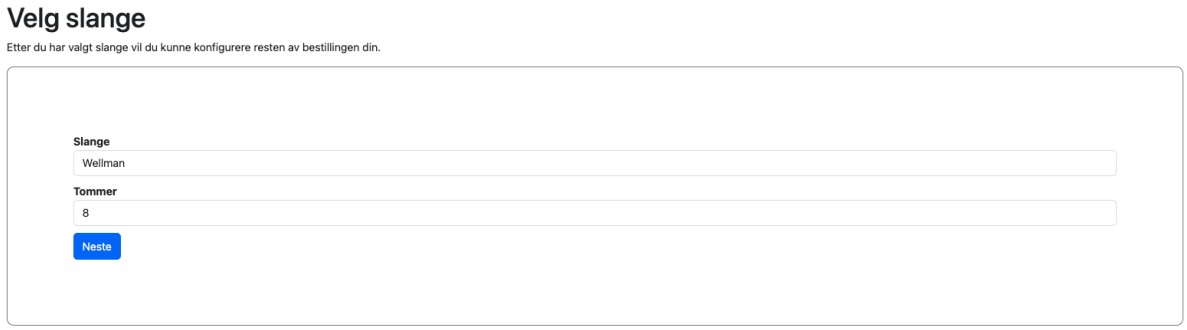
Mini Smart Factory Lab

Manual

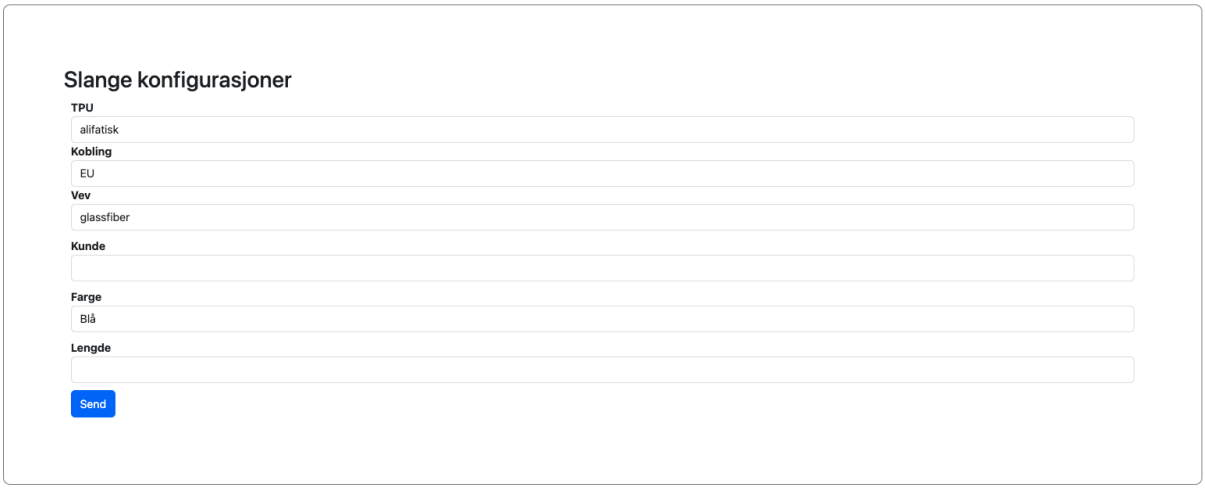
Mini Smart Factory Lab har i hensikt å vise en hel automatisert produksjonsprosess fra en konfigurator til ferdig produkt.

Konfigurator:

Denne konfiguratoren sørger for at man formaterer en bestilling riktig og forhindrer at man sender inn ugyldig data. Eksempelvis å skape en konfigurasjon som ikke er mulig, sende inn negativ lengde, eller velge tommestørrelse på slangen som ikke finnes. Den sørger også for at bestillingen blir sendt til riktig endepunkt, men dette krever at man har VPN tilgang.



Figur 1- Konfigurator Vindu 1



Figur 2 - Konfigurator - Vindu 2

Krav:

1. Har docker installert og kjørende på pc-en.
2. Har tilgang til VPN.

Slik gjør du:

1. sørg for at docker-dameon kjører.
2. Skriv inn kommandoen docker-compose up -d
3. Besøk http://localhost:6688 i nettleser

Dersom nettleseren din ser ut som figur 1 er man klar til å sende bestillinger, men man har behov for å sette opp boksen.

Nettverk:

Produksjonslaget kommuniserer via et lokalt nettverk. Dette blir simulert gjennom en hotspot som kjøres gjennom en pc. Det er viktig at både Arduino Uno Wifi og RPIéne er koblet til samme nettverk som bruker 2.4 GHz.

Ardunio Uno Wifi

Arduino Uno Wifi vil forsøke å koble seg til nettverket som ligger hard kodet i maskinen. Dersom man endrer nettverk, må man laste opp endringene og reboote maskinen.

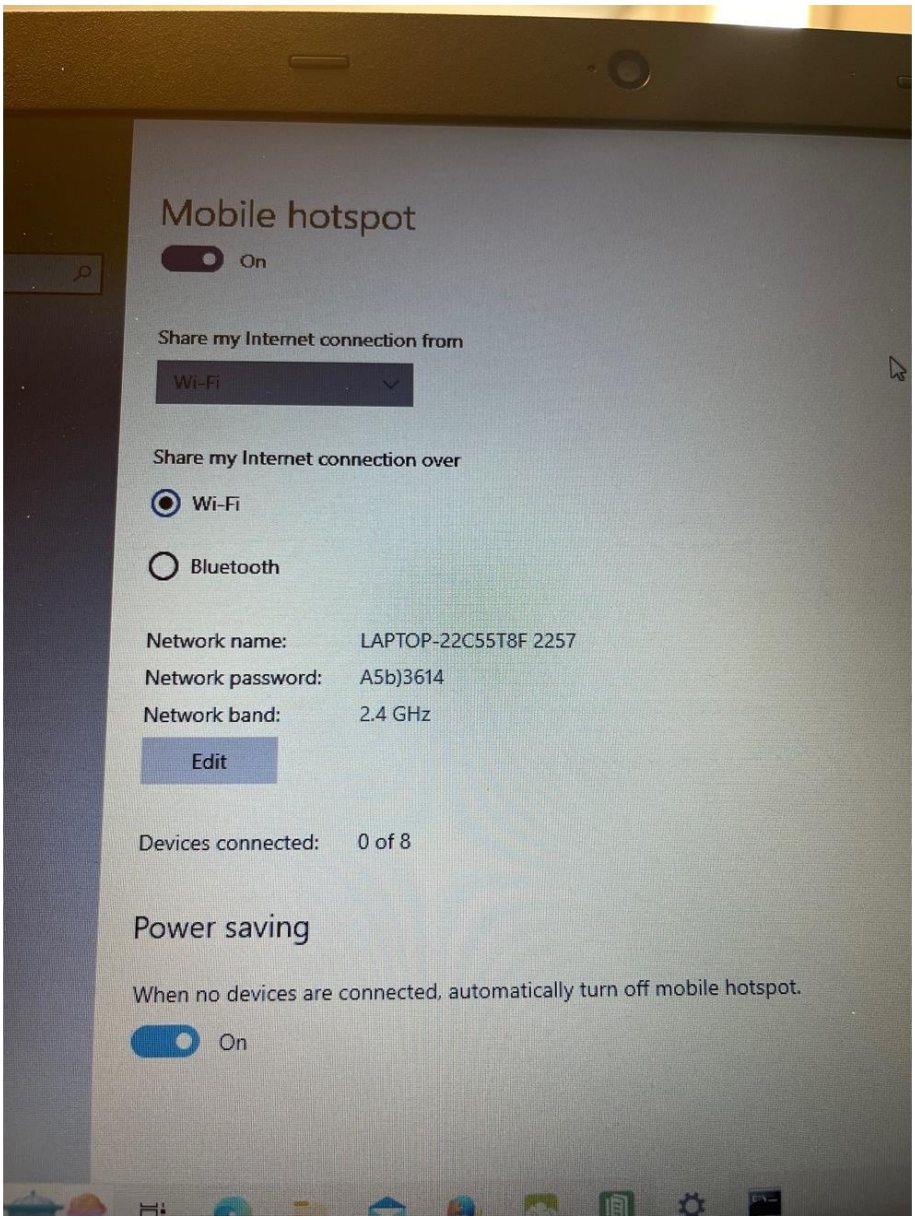
```
// Nettverk
#define SECRET_SSID "LAPTOP-22C55T8F 2257"
#define SECRET_PASS "A5b)3614"
```

Figur 3 - Arduino Uno Wifi – Nettverksinstillinger

Når man først foretar endringer i ardunio_secrets.h filen bør man sørge for at IP adressen som ardunioen bruker for å koble seg til mosquitto brokeren stemmer. Den skal peke på RPIén sin offentlige IP-adresse. For å finne riktig IP-adresse kan man bevege seg inn i terminalen på RPIén og skrive «hostname -I». Det vil sannsynligvis komme opp to IP-adresser. Du skal ikke bruke IP-adressen som begynner på «10.0..».

```
// IP - adresse til RPI
#define SECRET_IP "192.168.137.51" //CHANGE ME
```

Figur 4 - IP-adresse som brukes for å koble til mosquitto broker



Figur 5 - Bilde av nettverk hotspot

Sørg for at dataen i arduino_secrets.h samsvarer med hotspot konfigurasjonene, sammenlign figur 3 og 5.
Når man har koblet til nettverkets hotspot kan man se hvilke enheter er koblet til under «device connected», se figur 5.

Raspberry Pi Sentral Server

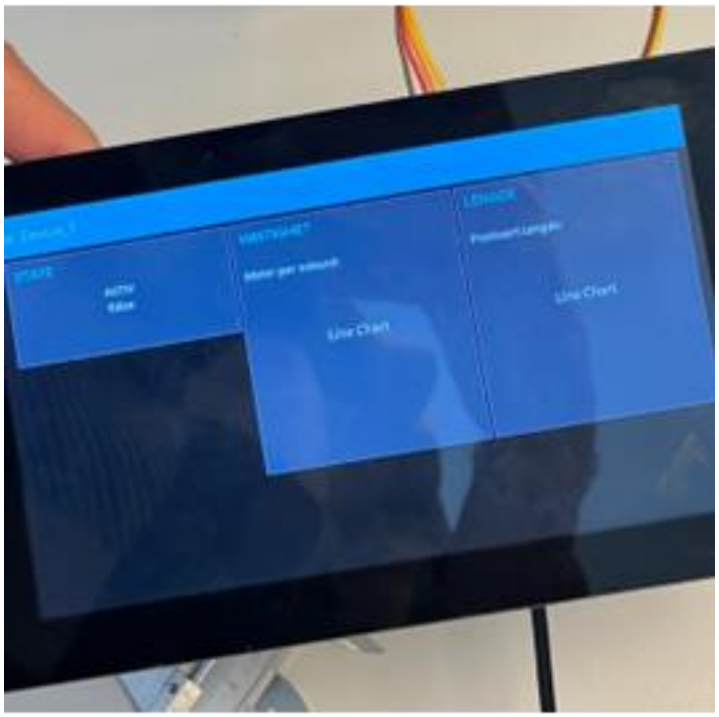
Sørg for at Raspberry Pi er koblet til riktig nettverk. Samme nettverk som Arduino Uno Wifi. Både mosquitto broker og node-red hostes på RPIén. Den er konfigurert til å starte tjenestene ved oppstart.

Visualisering av sanntidsdata og produksjonsdata

Vi bruker en annen Raspberry Pi for å visualisere dataen. For å få tilgang til visualiseringen behøver man bare å besøke en url, men man er avhengig av at man er tilkoblet samme nettverk som de andre komponentene, se avsnitt Nettverk og figur 3.

URL: [IP-adressen til RPI Sentral server]:1880/ui

- [IP-adressen til RPI Sentral server] må være lik som IP-adressen som Arduino Uno Wifi bruker, se figur 4.



Figur 6 - Visualisering av sanntidsdata og produksjonsdata

Dersom du har fulgt stegene og kravene møtes alt bør man være klar til å se sende en bestilling.

Anbefalt fremgangsmåte

1. Sjekk beholdning gjennom padden.

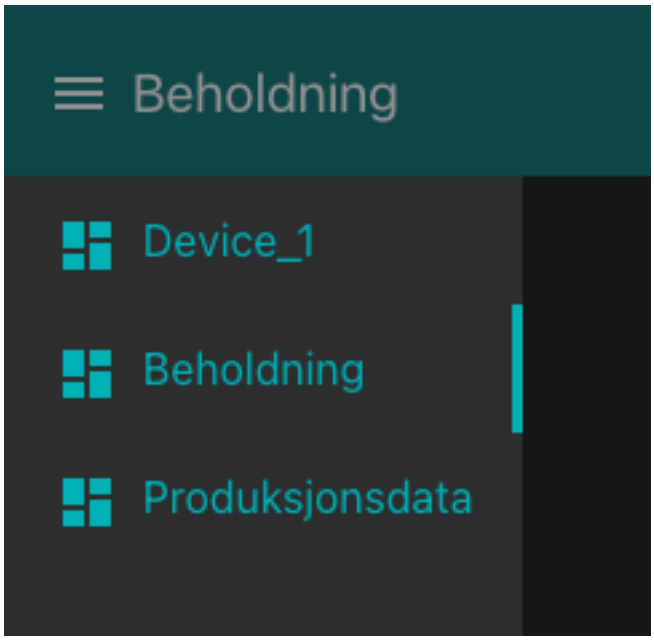
Når du foretar en bestilling i konfiratoren vil man velge hvilken TPU, Kobling, og Tråd/Vev. Etter produksjonen er ferdig skal beholdningen av valgt type reduseres.

Ressurser

Ressurs Id	Ressurs Navn	Ressurs Type	Mengde	Benevnelse	Pris per benevnelse
10	USA8	Kobling	338.00	antall	3250.00
6	polyesterbasert	TPU	7994.00	kg	3000.00
11	USA10	Kobling	322.00	antall	3500.00
12	USA12	Kobling	400.00	antall	5000.00
4	flammehemmende	TPU	7882.13	kg	4000.00
7	EU8	Kobling	74.00	antall	2000.00
9	EU12	Kobling	350.00	antall	4500.00
1	glassfiber	Tråd	9751.88	meter	350.00
5	alifatisk	TPU	7722.43	kg	2500.00
2	aramidfiber	Tråd	9908.21	meter	200.00
8	EU10	Kobling	298.00	antall	3000.00
3	nylon	Tråd	9942.74	meter	100.00

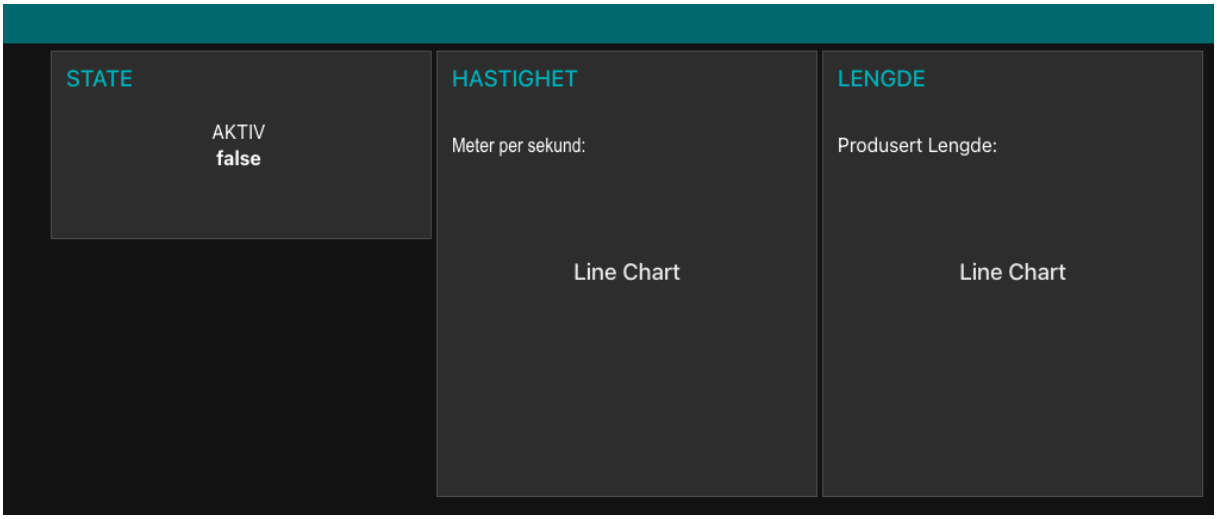
Figur 7 - Bilde av hvilken ressurser man har tilgjengelig

2. Beveg seg til Device_1 gjennom padden.



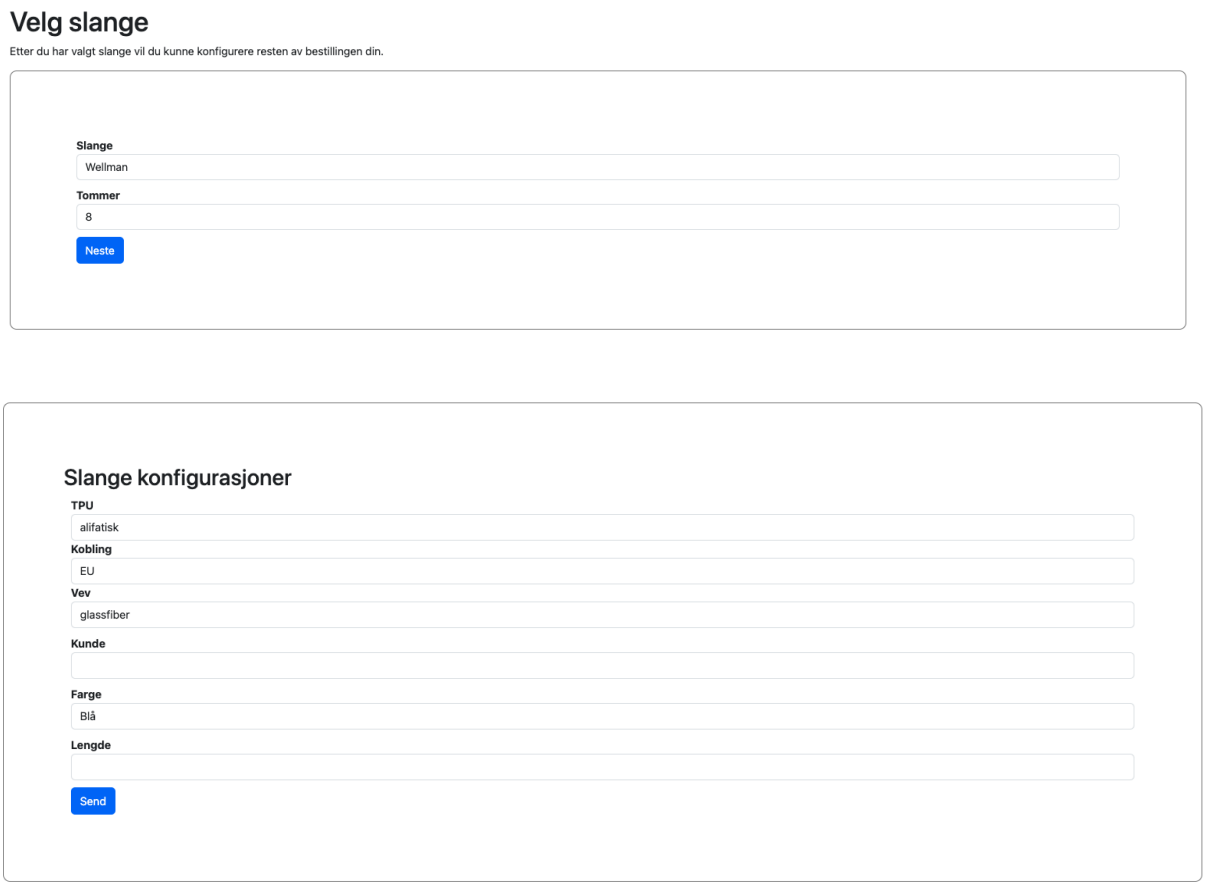
Figur 8 - Navigasjon meny i venstre hjørne

3. Devicee_1 – Visualiserer Sanntidsdata for Device_1 (Ardunio Uno Wifi)



Figur 9 - Vindu som visualiserer sanntidsdata

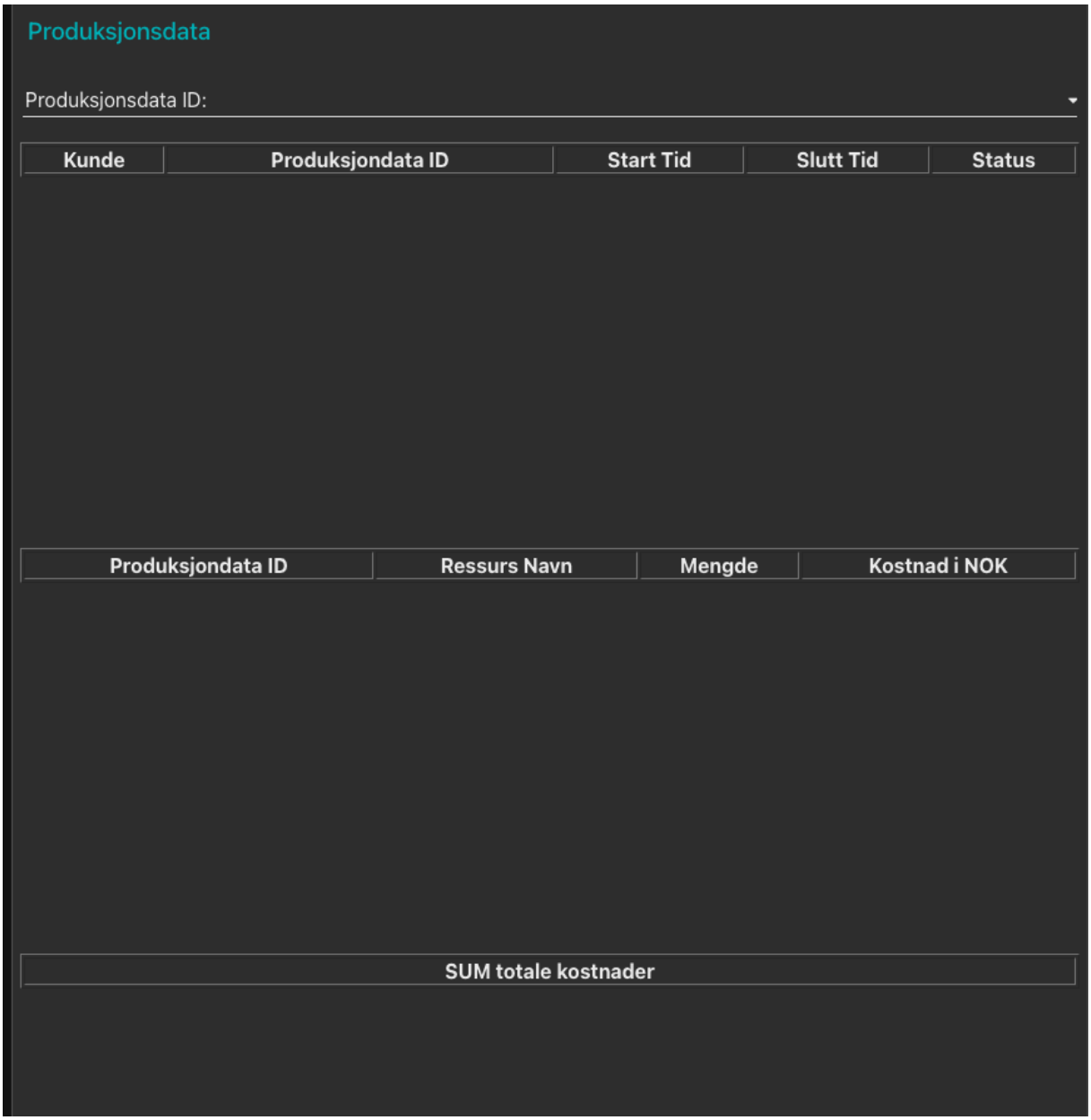
4. Fyll ut parameterene i bestillingen og trykk send.



Figur 10 - Brukergrensesnitt for å sende inn bestilling

5. Se på sanntidsdataen på padden.
6. Etter produksjonen er ferdig kan man bevege seg til Beholdning å se at ressursene er iht. bestillingen er blitt redusert.

7. Beveg deg til Produksjonsdata gjennom padden og velg produksjonsdata ID.



Figur 11 - Vindu for at man kan visualisere produksjonsdata

Produksjonsdata viser hvem som er kunde, unik id, start- og slutt- tid, og status. Den viser også hvilken ressurs som er medgått i produksjonen, kostnadene, og den totale summen av kostandene.

Vedlegg E - Milepæler

Milepæler
Sprint 1 - Konseptualisering og Planlegging: Definer prosjektets omfang, mål og planlegg de tekniske aspektene. Inkluder valg av komponenter som Arduino, encoder, samt programvarevalg som Node-Red eller Ignition.

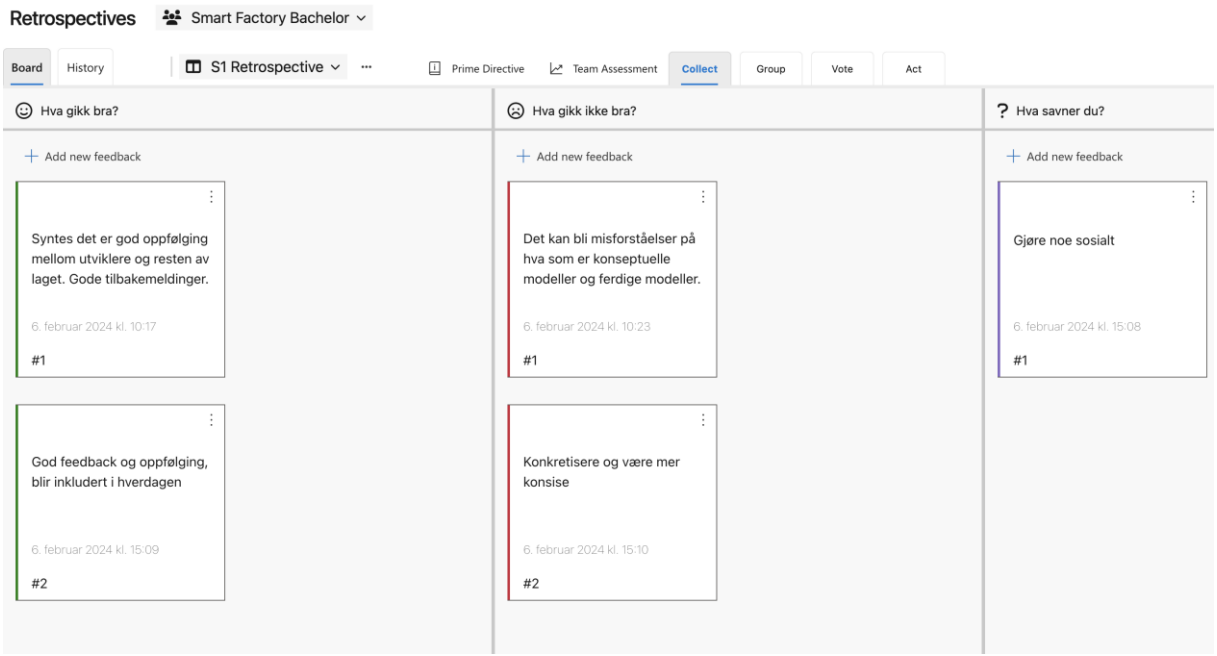
Sprint 2 - Utvikling av Prototype: Begynn byggingen av prototypen. Prioriter arbeidet med å koble Arduino til encoder og teste grunnleggende funksjonalitet. Sikre at alle komponenter kommuniserer effektivt. Få ting opp å gå. Skape grunnlaget for inkremental utbedring.

Sprint 3 – Brukergrensesnitt og VPN: Implementere løsningen av Erik inn i vårt prosjekt for å sikre kommunikasjon. Implementere de inkrementelle funksjonene.

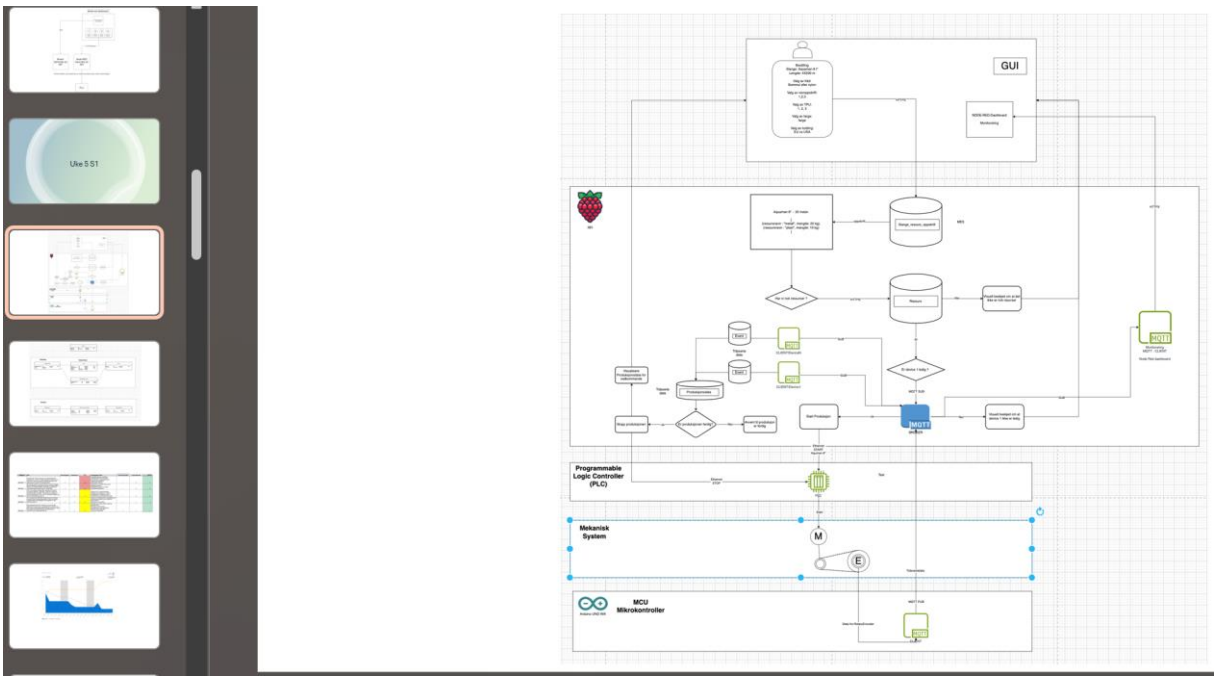
Sprint 4 – Planlegge utforming av den fysiske boksen og montere komponentene

Sprint 5 - Finjustering og Demonstrasjon: Gjennomfør finjustering av systemet, fiks eventuelle feil, og forberede demonstrasjonen av Smart Factory mini Lab box. Fokuser på hvordan løsningen kan integreres med ERP-systemer og presenter den endelige løsningen for interessenter.

Vedlegg F – Eksempel på retrospektivt notat fra Sprint 1



Vedlegg G – ukentlige møter



Vedlegg H – Roller i prosjektet

Asbjørn og Torkel har begge spilt viktige roller i gjennomføringen av dette prosjektet i samarbeid med Knowit, hvor kompetansen til hver av oss har bidratt til en velbalansert arbeidsfordeling. Asbjørn har hatt fokus på den fysiske delen av prosjektet, spesielt i kabling og implementering av fastvare, som krever presisjon og teknisk forståelse for å sikre påliteligheten av systemfunksjonaliteten. På den andre siden har Torkel hatt større fokus på datamodelleringen, som har vært kritisk for å utvikle robuste datastrukturer og dataflyter som understøtter systemets ytelse og effektivitet. Sammen har vi sørget for at både hardware- og software-aspektene av prosjektet harmonerer godt, som har vært essensielt for å møte de tekniske og funksjonelle kravene i prosjektet.

Vedlegg I – Uttalelse fra oppdragsgiver

knowit

Formell uttalelse fra Knowit angående bachelorprosjektet til Torkel Herskedal Ivarsøy og Asbjørn Håland Hetnes

Vi i Knowit er stolte av å fremheve den enestående innsatsen til Torkel Herskedal Ivarsøy og Asbjørn Håland Hetnes i gjennomføringen av deres bachelorprosjekt. Prosjektet er utført på en strukturert og grundig måte, som på en utmerket måte gjenspeiler Knowit sine verdier og standarder. Dette bachelorprosjektet har spilt en vesentlig rolle i etableringen av vår nye avdeling "Smart Factory". Denne avdelingen bygger på OT til IT-konvergensen, hvor IT integreres stadig tettere med det fysiske laget, og data strømlinjeformes både vertikalt og horisontalt i organisasjoner.

For at norsk prosessindustri skal kunne håndtere det digitale skiftet, må teknologier som IoT, AI og robotikk anvendes på en bærekraftig og effektiv måte. Resultatet av dette bachelorprosjektet vil være en nøkkelfaktor for å demonstrere at data kan flyte vertikalt, både fra nye og eldre maskiner. Ved å utvikle et brukergrensesnitt for uthenting av produksjonsdata fra eksisterende maskiner, legges grunnlaget for datadrevne beslutninger, tilgjengeliggjort gjennom solid datamodellering og bruk av IoT-enheter. Dette utgjør det første steget for en hjørnesteinsbedrift til å oppnå tidlige gevinster av produksjonsdata.

Samarbeidet med bachelorstudentene i Smart Factory-prosjektet har vært svært vellykket. Teamet har tilpasset prosjektet effektivt til våre spesifikasjoner og levert resultater som overgår våre forventninger. Den Mini Smart Factory-laben de utviklet integrerer IoT-enheter og nettverkssikkerhetsprotokoller på en imponerende måte, og gir oss verdifull innsikt gjennom sanntids- og blokkdata. Konfiguratoren og bestillingssystemet de implementerte har tydelig demonstrert verdien av et slikt IoT-prosjekt for våre kunder.

Kommunikasjonen har vært svært effektiv, med lavterskeldialoger, felles utviklingsprosesser og ukentlige møter. Teamet har vist fremragende evner til å arbeide individuelt samtidig som de følger opp prosjekteiere ved behov i en hektisk hverdag. Torkel Herskedal Ivarsøy har ledet arbeidet med datamodellering og håndtering av data på IT-siden, mens Asbjørn Håland Hetnes har ledet utviklingen av programvare på OT-siden, og sammen sikret en vellykket konvergens i et godt utviklet modulært system.

Vi ser frem til å bygge videre på denne suksessen og til å fortsette samarbeidet med talentfulle studenter i fremtidige prosjekter.

Vedlegg J – Automasjonspyramiden med VPN