Sensur av hovedoppgaver

Universitetet i Sørøst-Norge
Fakultet for teknologi og maritime fag

Prosjektnummer: **2020-05**
For studieåret: **2018/2020**
Emnekode: **SFHO3201-1 19H Bacheloroppgave**

Prosjektnavn:
Del gjenkjenning
Parts Vision Recognition

Utført i samarbeid med: Tronrud Engineering AS

Ekstern veileder: Øistein Røste

Sammendrag: Ergonomisk arbeidsbord med gjenkjenning av unike deler.

Stikkord:
- Løftemekanisme
- Gjenkjenning av deler
- Brukergrensesnitt

Tilgjengelig: JA

Prosjekt deltagere og karakter:

Navn	Karakter
Adithya Arun	
Kristian Klev Hansen	
Kirisan Manivannan	
Daniel Gjesteby	
Jens Paulsen	
Rahmat Mozafari	

Dato: 15. juni 2020

_____	_____	_____
Sigmund Gudvangen	Karoline Moholth	Hilde Berggren
Intern Veileder	Intern Sensor	Ekstern Sensor

Hovedrapport

Gruppe 5, rom i-116

25. Mai 2020

Gruppemedlemmer	Veiledere	Sensorer
Rahmat Mozafari	Ekstern veileder	Ekstern sensor
Adithya Arun	Øistein Røste	Hilde Berggren
Kristian Klev		
Daniel Gjesteby	Intern veileder	Intern sensor
Kirisan Manivannan	Sigmund Gudvangen	Karoline Moholth Mcclenaghan
Jens Paulsen		

Abstrakt

Vår arbeidsgiver, Tronrud Engineering, har et ønske om å delvis automatisere gjenkjenningsprosessen av deler de videresender til overflatebehandling hos eksterne leverandører. Denne prosessen utføres manuelt per dags dato. Målet til bachelorgruppen har vært å designe et produkt som kan automatisere denne prosessen. Hensikten med denne rapporten er å forklare utviklingen av vår løftemekanismen, samt utviklingen av programvaren som er brukt i produktet. Bildegjenkjenning, brukergrensesnittet og databaser er de store delene av programvaren. I tillegg fremstilles valg av komponenter og begrunnelse for design i rapporten.

Innholdsfortegnelse

Figurliste

Tabelliste

1 Introduksjon

1.1 Problemstilling

Tronrud Engineering produserer deler til automatiserte industriløsninger som de lager selv. Når delene skal overflatebehandles blir de sendt videre til et annet selskap. Per dags dato kommer delene i en pall hvor de er plassert over en 2D-tegning som består av informasjon om delen. Antall deler og størrelse kan variere, men vil ikke være større enn størrelsen på selve pallen. 2D-tegningene som tilhører delene blir lagret som en bunke i en hylle.

Når delene kommer tilbake fra behandlingen kan de være plassert på forskjellige steder på pallen, som betyr at en lagerarbeider på Tronrud må gjenkjenne delen ved å sammenligne med 2D-tegningen som de har lagret. Noen av delene kan se så og si identiske ut, hvor kun et hull utgjør forskjellen på delene. I tillegg er det tidskrevende å gå igjennom hele bunken med 2D-tegninger for å finne riktig artikkelnummer til de ulike delene. Problemet Tronrud ønsker bistand med å løse er derfor å finne en mer effektiv og mindre ressurskrevende måte å identifisere de ulike delene.

1.2 Oppgavebeskrivelse

Vår oppgave dreier seg om å gjøre prosessen enklere for arbeiderne ved hjelp av teknologi. 2D-tegningene er omfattende og inneholder alle delene som er på pallen. Ved hjelp av bildegjenkjenning er målet vårt å eliminere flest mulig alternativer av delene i 2D-tegningene, slik at det blir enklere å gjenkjenne de ulike delene. Dette vil gjøre hverdagen enklere for lagerarbeideren og prosessen med å finne de riktige 2D-tegningene. Når delen blir gjenkjent ved hjelp av vårt system skrives det ut en etikett som identifiserer delen.

En annen del av vår oppgaveløsning går ut på å utvikle en løftemekanisme som løfter delene opp til en ergonomisk posisjon for arbeideren. Arbeideren skal manuelt kjøre inn en europall med egenproduserte deler til systemet. Pallen vil da bli løftet opp til ønsket posisjon. Deretter kan arbeideren ta av ønsket del og plassere den i et gjenkjenningsområde. Her skal systemet lagre informasjon om delen i en database som Tronrud skal ha tilgang til. Etter dette skal arbeideren legge delen i en annen pall, som også er løftet opp til en ergonomisk posisjon for arbeideren. Når alle delene har blitt lagret i en database skal løftemekanismen senke pallen ned igjen på bakken og lagerarbeideren kan så sende delene videre til overflatebehandling.

Når delene kommer tilbake fra overflatebehandling skal prosessen gjentas med noen få tilpasninger. Pallen fraktes til systemet av arbeideren og delen løftes opp til ønsket posisjon. Delene løftes av pallen av arbeideren og plasseres i et gjenkjenningsområde. Systemet skal kunne kjenne igjen delen fra databasen. Deretter skriver systemet ut en etikett som identifiserer delen. Arbeideren klistrer etiketten på delene og legger de over på den andre pallen som er løftet opp til ønsket posisjon. Når alle delene har blitt gjenkjent og løftet over på den andre pallen, skal pallen løftes ned ved hjelp av arbeiderens betjening.

1.3 Gruppeoversikt

	**Adithya Arun** *Studieretning:* Maskiningeniør *Ansvarsområde:* SCRUM-Master og testansvarlig *Kontaktinformasjon:* *Tlf:* (+47) 950 02 690 *e-post:* adithya.arun19@gmail.com
	**Kristian Klev Hansen** *Studieretning:* Maskiningeniør *Ansvarsområde:* CAD-ansvarlig *Kontaktinformasjon:* *Tlf:* (+47) 957 33 464 *e-post:* kristianklev@gmail.com
	**Kirisan Manivannan** *Studieretning:* Dataingeniør *Ansvarsområde:* Programvareansvarlig *Kontaktinformasjon:* *Tlf:* (+47) 917 56 826 *e-post:* kirisanm@gmail.com
	**Daniel Gjesteby** *Studieretning:* Dataingeniør *Ansvarsområde:* Risikoansvarlig *Kontaktinformasjon:* *Tlf:* (+47) 468 09 350 *e-post:* daniel.f.gjesteby@gmail.com
	**Jens Paulsen** *Studieretning:* Dataingeniør *Ansvarsområde:* Dokumentansvarlig *Kontaktinformasjon:* *Tlf:* (+47) 970 33 314 *e-post:* jens.paulsen95@gmail.com
	**Rahmat Mozafari** *Studieretning:* Dataingeniør *Ansvarsområde:* Produkteier og prototypeansvarlig *Kontaktinformasjon:* *Tlf:* (+47) 472 92 878 *e-post:* rahmat@mozafari.no

1.4 Veiledere og sensorer

Ekstern veileder

Navn: Øistein Røste

Stilling: Head of Production Solutions

Den eksterne veilederen er bindeleddet mellom prosjektgruppen og bedriften. Den eksterne veilederen har ansvaret for å hjelpe prosjektgruppen med tekniske spørsmål. I tillegg skal den eksterne veilederen sørge for at prosjektgruppens nødvendige ressurser til å fullføre prosjektet er tilgjengelig.

Oppdragsgiver

Navn: Kristin Engebø

Stilling: General Manager Manufacturing

Oppdragsgiveren bidrar til å definere oppgaven som bedriften ønsker at prosjektgruppen skal fullføre. I tillegg skal oppdragsgiveren følge med på utviklingen av prosjektet og gi tilbakemelding på om prosjektgruppen jobber i riktig retning i forhold til bedriftens ønsker.

Konsulent

Navn: Tommy Karlsson

Stilling: Chief Digital Officer

Konsulenten bidrar til å hjelpe prosjektgruppen med tekniske utfordringer de møter underveis. Dette gjøres fra bedriftens perspektiv.

Ekstern sensor

Navn: Hilde Berggren

Stilling: General Manager

Ekstern sensor skal være med på karaktersetting av prosjektgruppen i henhold til bedriftens forventninger og standarder.

Intern veileder

Navn: Sigmund Gudvangen

Stilling: Førsteamanuensis

Den interne veilederen har som ansvar å veilede prosjektgruppen og komme med innspill ved hjelp av hans kompetanse. En intern veileder skal ikke styre prosjektgruppen, men heller gi råd og forslag til løsninger.

Intern sensor

Navn: Karoline Moholth Mcclenaghan

Stilling: Universitetslektor

Den eksterne sensoren skal være med på karaktersetting av prosjektgruppen i henhold til skolens læreplan og kompetansemål.

1.5 Ansvarsområder

1.5.1 SCRUM-Master (Adithya Arun)

SCRUM-modellen krever et gruppemedlem som tar ansvar for å sikre at gruppen får mest mulig bruk for modellen og kan levere best mulig resultater. Dette gjøres ved å sikre riktig implementering av SCRUM prosessen. Vanligvis er ikke SCRUM-master en del av utviklingsteamet, men i prosjektet vårt har Adithya fått dette ansvaret. SCRUM-masteren må bruke diskusjoner fra sprint-møtene til å optimalisere prosessen med hensyn til gruppens behov. Selv om tittelen har "master" i navnet, er ikke SCRUM-master en gruppeleder. SCRUM-master skal styre gruppen i riktig retning innenfor de definerte reglene av prosessen ved potensielle avvik.

1.5.2 Produkteier (Rahmat Mozafari)

Produkteieren er koblingen mellom kunden og utviklingsteamet. På samme måte som SCRUM-masteren, er ikke produkteieren en del av utviklingsteamet til vanlig, men i prosjektet vårt har Rahmat fått dette ansvaret. Produkteieren skal alltid sørge for at oppgavene i backloggen er prioritert etter kundens ønsker. I tillegg skal produkteieren sørge for at oppgavene i backloggen er sortert på en systematisk måte. Rollen sikrer at produktet utviklingsteamet leverer oppfyller kundenes krav.

1.5.3 Testansvarlig (Adithya Arun)

Testansvarlig har ansvaret for å lage en testplan av maskinvare og programvare. Dette gjøres ved å sjekke om produktet oppfyller oppførte krav. Testing inneholder fysisk testing av ferdige komponenter, matematisk beregning, kjøring av programvaren eller simuleringer. Testspesifikasjon må være strukturert på en forståelig måte slik at det kan føres tilbake til de respektive kravene. Testansvarlig har i tillegg ansvar for å sette opp og bestemme tid som kreves for å teste komponenter. Etter dette må den testansvarlige sette opp en passende frist for den ferdige komponenten.

1.5.4 Risikoansvarlig (Daniel Gjesteby)

Risikoanalyse er en kontinuerlig prosess gjennom hele prosjektet som evaluerer risiko konstant. Dette gjelder både tekniske risikoer og prosjektrisikoer. Risikoansvarlig må sikre at denne prosessen foregår jevnt gjennom hele prosjektet. Dette er viktig siden designet av produktet er en dynamisk prosess og kan forandres veldig ofte. Da er det risikoansvarlig sitt ansvar å oppdage nye risikoer som følger med forandring i design eller prosjektet. I tillegg har personen ansvar for å prioritere risikoene, vurdere alvorlighetsgrad og lage en plan for hvordan utviklingsteamet kan forhindre risikoen gjennom design. Hvis risikoen er uunngåelig må risikoansvarlig fikse problemet på en annen måte. Siden vi er et flerfaglig utviklingsteam har alle ansvar for å evaluerer risiko, men risikoansvarlig har hovedansvaret for å sikre at risikoanalyse blir gjort til riktig tid.

1.5.5 Dokumentansvarlig (Jens Paulsen)

Dokumentansvarlig har hovedansvaret for å bestemme mappestruktur og utseende av dokumentene. Dokumentansvarlig har også ansvaret for å sørge for at alle skriver deres del av dokumentasjon og lage det ferdige dokumentet. Bestemmelse av programvare til dokumentskriving, tekststørrelser og struktur, kildehenvisning og rettskrivning er ansvarsområdet til dokumentansvarlig.

1.5.6 CAD-ansvarlig (Kristian Klev Hansen)

CAD-ansvarlig har ansvaret for at alle delene som modelleres følger samme oppbyggingsmetode. Metoden skal basere seg på å være forståelig for alle som åpner CAD-filene. CAD-ansvarlig skal også sørge for at gruppen følger standardene som næringslivet følger og at delene skal være enkle å lese fra 2D-tegningene. I tillegg til dette skal CAD-ansvarlig sørge for at delene er mulige å produsere.

1.5.7 Prototypeansvarlig (Rahmat Mozafari)

Prototypeansvarlig har ansvaret for at design av komponentene som skal brukes i prototypen er ferdige til riktig tid. I tillegg må prototypeansvarlig ha oversikt over oppbygging av prototypen. Selv om alle gruppemedlemmene har ansvar til å bidra til prototypen, har prototypeansvarlig hovedansvaret. Prototypen må være ferdig til riktig tid for at at mekanismen kan testes før vi lager sluttproduktet.

1.5.8 Programvareansvarlig (Kirisan Manivannan)

Posisjonen programvareansvarlig kan bli sett på som prosjektleder for programvaredelen av et system. Ansvarsområde strekker seg fra både det tekniske til det psykiske med tanke på det å holde motivasjonen oppe i en gruppe når programmeringen foregår. I ansvarsområdet er det først og fremst viktig å passe på at alle jobber som et lag. Det er viktig å motivere de på laget, slik at engasjementet ikke avtar. Som programvareansvarlig blir man også mellomleddet mellom data og maskin, slik at krav, design og programmet samsvarer med hverandre. Det er viktig å passe på at alle bruker samme kodespråk og struktur slik at kravene blir møtt. Koordinering av datarelaterte oppgaver og følge prosessen er også en del av ansvarsområdet til en programvareansvarlig. Når et gruppemedlem ikke er klar over hvordan en oppgave skal løses, så er det programvareansvarlig sin jobb å gjøre det mer forståelig. Den som er programvareansvarlig må ha full oversikt over hva programvaren gjør til enhver tid. Det er viktig å ha tett kontakt med kunden slik at kravene blir satt riktig og møtt.

2 Prosjektstyring

2.1 Prosess

Prosjektgruppen begynte prosjektet med å velge en prosessmodell. Ettersom det eksisterer et stort utvalg av modeller med sine respektive fordeler og ulemper, var vår hensikt å velge den modellen som er best egnet til vårt prosjekt. Den opprinnelige idéen var å velge en modell som hovedmodell og deretter lage en hybrid ved å implementere funksjoner som vi synes kan forbedre hovedmodellen. Etter å ha evaluert prosessmodeller som CAFCR, V-modell, Unified Process, Spiral Model, KANBAN og SCRUM kom vi frem til at SCRUM passer best til våre behov. Sprint 1 ble brukt som en testperiode for SCRUM.

2.2 SCRUM

SCRUM er en iterativ "agile" modell som gjør det mulig å identifisere risiko/feil og fikse det med relativt lave kostnader. SCRUM gjør det mulig for gruppemedlemmene å alltid vite hva deres nåværende oppgave er og hvordan oppgaven bidrar til prosjektet som helhet. Modellen tillater definering av både kortsiktige og langsiktige mål, som var veldig tiltalende for oss.

SCRUM inneholder bruk av sprinter (iterasjoner). En sprint kan vare fra en til to uker i vårt prosjekt. Vi valgte å ha sprinter som varte i syv dager (mandag - søndag) i den tidlige fasen av prosjektet. Sprinter med varighet i 2 uker begynte vi med da vi var i gang med det tekniske arbeidet. Denne strukturen ble valgt fordi den ville fungere bra med de ukentlige møtene med Tronrud Engineering og den interne veilederen. Fremgangen på produktet ble vist til Tronrud Engineering hver mandag etter en sprint som ble ferdig på søndag. På denne måten hadde vi mulighet til å justere produktet med hensyn til kundenes ønsker og implementere justeringene i neste sprint-planlegging. Deretter ble justeringene diskutert med intern veilederen på onsdager. Ved å dele arbeidsmengden i mindre segmenter fikk vi mulighet til å se fremgangen vår og vurdere om arbeidsmengden måtte økes for å nå det endelige målet.

2.3 Evolusjonær modell

Den evolusjonære modellen innebærer at det foregår utvikling av produktet fra de tidlige stadiene av prosjektet. Prosessen gav oss mulighet til å få tilbakemelding om hele systemet fra kundene, før detaljert teknisk

arbeid ble igangsatt. Dette ville reduserer tap av tid og penger dersom endringer måtte gjøres. Flere detaljer ble lagt til i produktet under hver sprint, for å komme nærmere et ferdig produkt. På denne måten fikk gruppemedlemmene en god forståelse av systemet og det ble enklere å ha fokus på grensesnitt mellom forskjellige deler av systemet som helhet.

2.4 SCRUM-vilkår

SCRUM består av flere bestemte gjøremål som må bli fulgt i dette prosjektet. Disse gjøremålene er presenteres nedenfor.

2.4.1 Sprint planlegging

Etter mandagsmøtene med Tronrud Engineering, hadde gruppen planleggingsmøter for å planlegger den neste sprinten. Arbeidsoppgaver ble da fordelt mellom gruppemedlemmene og hensikten til neste sprint ble definert. Etter at hovedmålet ble bestemt skulle det ikke forandres. Vi vektla å velge oppgaver som ville være oppnåelige i den fastsatte tidsrammen. Oppgavene ble hentet ut fra produkt backloggen og plassert i sprint backlog.

2.4.2 Sprint gjennomgang

Ved oppstart av en ny sprint ble forrige sprint gjennomgått i et gruppemøte. Hensikten med møtene var å oppsummere fullførte oppgaver og om målet ble oppnådd. Potensielle forbedringer og generell bruk av SCRUM-prosessen ble diskutert. Disse forbedringene og eventuelle justeringer ble tatt videre til planleggingsmøtet.

2.4.3 StandUp møter

Arbeidsdagene begynte med et standup-møte. Alle gruppemedlemmene oppsummerte da hva som var blitt gjort og hva som skulle gjøres fremover. Her tok gruppemedlemmene opp eventuelle utfordringer som hadde oppstått, slik at andre medlemmer kunne bistå i oppgaveløsningen. Ved bruk av standup-gjennomganger fikk hele gruppen oversikt over fremgangen innenfor alle fagområdene. Etter gruppemedlemme begynte å jobbe hjemfra på grunn av korona viruset, ble disse møtene mer uoffisiell.

2.4.4 User-stories

User-stories er brukt i SCRUM til å definere hva systemet må kunne på en minst mulig teknisk måte. Strukturen vi har fulgt er:

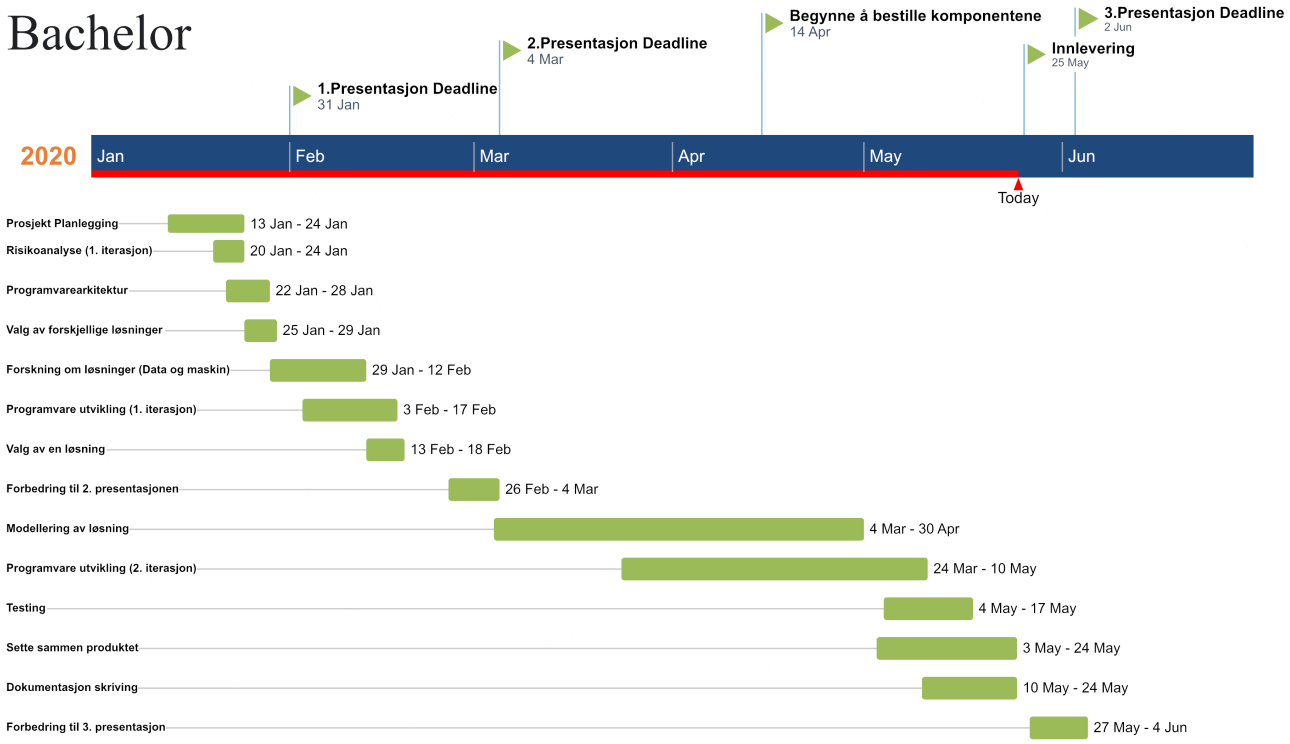
- Jeg er .....
- Og jeg vil at systemet skal .....
- Fordi .....

Ved bruk av user-stories har vi fått oversikt over hva "stakeholders" og "shareholders" ønsker fra systemet. Deretter ble user-stories utviklet til tekniske krav.

2.4.5 Product backlog

Produkt-backloggen inneholder alle arbeidsoppgavene bachelorgruppen må utføre for å oppfylle de oppsatte kravene. Imøtekommelse av de de definerte kravene ble vektlagt for å utvikle et ferdig produkt som tilfredsstiller kundens forventninger. Oppgavene ble sortert og prioritert av produkteieren med hensyn til kundenes ønsker.

2.5 Prosjektplan



Figur 1: Gantt Diagram av prosjektplan

En prosjektplan er nødvendig for å evaluere prosjektets progresjon. Figur 1 viser det første utkastet til prosjektplanen vår. Selv om dette var en tidlig iterasjon som ville bli endre underveis, fikk vi visualisert hva som måtte gjøres. I tillegg gir prosjektplanen en oversikt over når de ulike oppgavene måtte gjøres for å kunne levere et ferdig produkt til riktig tid. Gjennom prosjektplanen fikk bedriften og intern veileder muligheten til å følge med på progresjonen til prosjektgruppen. Ved å sette viktige tidsfrister gjennom hele prosjektperioden, ble oppgavene til hver sprint enklere å følge opp med tanke på å få fullført sluttproduktet.

Det å bygge et ferdig produkt ble veldig komplisert grunnet konsekvenser knyttet til Covid-19, som forhindret samlinger av grupper. Tronrud Engineering har ikke hatt lov til å gi oss tilgang hos de før smittefaren av koronaviruset er borte. Løsningen vi kom på etter diskusjon med Tronrud Engineering var å levere en bestillingsliste og monteringsmanual. Målet er å levere et ferdig definert produkt, slik at bedriften kan produsere produktet med så lite etterarbeid som mulig.

2.6 Målsetting

Det vi prøver å oppnå med denne oppgaven å utvikle et system slik at arbeidsprosessen til lagerarbeiderne blir forenklet. Måten vi ønsket å realisere dette på var ved å sammenslå en løftemekanisme med et gjenkjenningsprogram, med mål om å gjøre prosessen mer ergonomisk hensiktsmessig for lagerarbeidere. Tanken vår var å eliminere risiko for ryggskader ved at systemet legger til rette for at arbeideren ikke må bøye ryggen uhensiktsmessig. I tillegg har vår målsetting vært at produktet skal forenkle prosessen med gjenkjennelse av deler som likner på hverandre, da dette krever mye fokus og tidsbruk for arbeideren. Disse målsettingene er lagt til grunn for å utvikle et fungerende produkt for Tronrud engineering.

2.7 Begrensninger

Tronrud engineering har ønsket at gruppen utvikler et ferdigstilt produkt, så tidsbruk har vært vår største begrensing for å imøtekomme deres ønske. Datastudentene har møtt på utfordringer i forhold til håndtering av ukjente verktøy oppgaven har krevd, samt å innhente kunnskap om utvikling av et program som går ut på bildebehandling. Dette er kunnskap som går utenom lærebøkene. Dette har ført til at mye tid har blitt brukt på å lære å håndtere de ulike verkøyene og mindre tid har blitt brukt på å utføre arbeidet som vi må gjøre med disse verktøyene. Under gjennomføringen av bachelorprosjektet har det foregått en global pandemi, som har ført til at skolen ble stengt etter krav fra myndighetene. Gruppemedlemmene har derfor måttet holde seg adskilt hjemme og jobbet separat med bachelorprosjektet. Dette har ført til utfordringer i form av at vi ikke har hatt tilgang til utstyr som kunne gjort oppgaveløsningen mer effektiv, samt at det har vært vanskelig å jobbe helhetlig som en gruppe.

2.8 Økonomi

Vi har en god ide om hvor mange og hva slags komponenter vi trenger til produktet. Få leverandører viser informasjon om prisene til delene vi har tenkt til å kjøpe inn. Vi har laget en komponentliste som inneholder pris på de komponentene vi har funnet pris på (Vedlegg - Kapittel 6). Tronrud Engineering har mulighet til å hjelpe oss med å finne prisene på de resterende delene. Vi har kjøpt inn et kamera som mest sannsynlig vil fungere for 700kr.

2.9 User-stories

User-stories til produktet har blitt laget utifra synspunktet til oppdragsgiveren, lagerarbeideren, bedriften, og ingeniøren ("stakeholders" og "shareholders"). Dette er for å gi produktet og prosjektet et bredt aspekt av hvordan systemet vil fungere for de forskjellige aktørene. Som user-stories er definert i 2.4.4 så er dette brukt i SCRUM. User-stories til datadelen er delt opp i mindre deler (epic stories): UI (Brukergrensesnitt), databasen, gjenkjenningen og programvaren (software) for å kunne skille mellom de og få en bedre oversikt innenfor programvaren.

2.10 Verktøy

2.10.1 Kommunikasjonsverktøy

Microsoft Teams

Microsoft teams blir brukt til å kommunisere og dele filer med oppdragsgiver. Teams var ønsket plattform for oppdragsgiver så vi har forsøkt å laste opp filer som vi ønsker å vise fram til oppdragsgiver før møtene.

Facebook messenger

Vi har brukt Facebook messenger for å kommunisere internt med prosjektgruppen. Når vi har ønsket å dele informasjon hurtig har vi brukt Facebook messenger ettersom det blir sjekket oftere enn andre kommunikasjons-plattformer.

Discord

Slack ble brukt til intern kommunikasjon, før vi gikk over til Discord. Da skolen ble steng grunnet Covid-19 fant vi ut at det ikke var mulig med felles "voice channel" i slack med gratis-bruker. Derfor gikk vi over til å bruke Discord for å kommunisere internt. Informasjon vi ønsket at skulle ligge mer synlig for gruppen enn på Facebook messenger ble sendt på Discord, siden informasjon på Facebook messenger fort kan gå tapt i mengden av meldinger. Discord er også brukt til å ha interne stemmechat-møter med prosjektgruppen, og til å kommunisere med individuelle gruppemedlemmer over stemmechat.

E-post

E-post ble brukt til mer formell kommunikasjon over internett. E-post ble mest brukt til å avtale møter og kontakte prosjektgruppens veiledere, både ekstern og intern.

2.10.2 Arbeidsverktøy

Google Drive

Dokumenter som vi selv har opprettet er samlet og lagret i en felles mappe i Google Drive, som alle kan se og endre. I denne fellesmappen har vi mapper til hvert enkelt gruppemedlem og til andre hensikter for prosjektet.

Trello

Trello er et verktøy som er veldig kompatibelt med SCRUM prosessen. Dette er fordi Trello lar oss sette opp en tavle med flere seksjoner hvor vi kan legge til oppgaver. Oppgavene ligger synlig på Trello-tavlen og tildeles til hver person. Dette gjør det lett å organisere og holde oversikt over gruppemedlemmenes oppgaver. Ettersom vi har brukt SCRUM som prosjektmodell har det vært naturlig for oss å bruke Trello.

Overleaf

Vi har brukt latex for å skrive dokumentene våres. Overleaf er da et nyttig arbeidsverktøy for gruppen siden det lar alle medlemmene skrive i det samme dokumentet i latex format over internett.

Online.Officetimeline

Dette er en nettside som lar brukeren lage et Gantt diagram som gjør det mulig for gruppen å lage en prosjektplan som gir en oversikt over framtidige oppgaver og hvordan gruppen ligger an tidsmessig.

Lucidchart

Lucidchart er et nettbasert samskrivningsverktøy som gjør at flere personer kan jobbe og få tilgang til ulike typer modeller. Vi har brukt Lucidchart til å modellere UML-diagrammer for programvarearkitekturen.

Draw.io

Draw.io er et nettbasert tegningsverktøy som er brukervennlig og gir oss muligheten til å dele tegningene på en enkel måte. Bruk av draw.io har samme hensikt som lucidchart, men er helt gratis, da lucidchart kun har gitt oss begrenset tilgang i gratis-versjonen.

Solidworks

Solidworks er brukt både til modellering av komponentene, og til FEM-analyse for å evaluere styrken til produktet. Maskinstudentene har opplæring i dette programmet.

OpenCV

OpenCV er et klassebibliotek basert på bildebehandlig.

Qt

Qt er et programvarebibliotek som er brukt for å utvikle brukergrensesnittet.

Qt creator

Qt creator er en programvare som både lar oss utvikle brukergrensesnittet gjennom Qt creator sitt eget grafiske brukergrensesnitt, samtidig som det fungerer som et utviklingsmiljø som lar oss utvikle vår egen programvare til prosjektet.

2.11 Kravspesifikasjon

Et krav er en funksjon som systemet må ha, eller en begrensning på systemet. Kravspesifikasjonen er et system eller en liste hvor gruppemedlemmene og kunden kan bli enige om kravene som er satt på produktet og prosjektet. Kravene skal godkjennes av kunden. Etter kravspesifikasjonen er definert er det ansvaret til prosjektgruppen å lage et produkt som oppfyller disse kravene. Enkle oppgaver som blir delt ut i sprinter, blir derivert ut ifra disse kravene. Kravene er prioritet basert på viktigheten til produktets funksjonalitet, og ønsker fra kunden. Kravene er laget fra user-stories og har vanligvis tallverdier som kan testes når produktet er klart. Kravspesifikasjonen er tilgjengelig i 3. avsnitt i vedlegget.

2.12 Testplan

En testplan er nødvendig for å sikre at programvaren, komponentene og hele systemet som blir produsert fungerer slik at de oppfyller kravene. Testplanen skal beskrive når, hvordan og hvorfor testene blir utført. Testprosessen skal foregå samtidig som utviklingsprosessen, for å identifisere potensielle feil raskest mulig, uten at de går utover andre deler av prosjektet. Etter testing får delen av systemet som ble testet "godkjent" eller "ikke godkjent", avhengig av om den oppfylte kravene eller ikke. Testing kan utføres ved bruk av forskjellige metoder. Delene trenger ikke å være fysisk produsert for å kunne testes.

Følgende metoder har blitt brukt til testing av bachelorgruppen:

- Fysisk testing av deler som har blitt produsert/montert.
- FEM analyse av modellene I SOLIDWORKS.
- Simulering av programvare.
- Visualisering av software gjennom diagrammer.
- Håndregninger.

2.13 Risikoanalyse

Under utviklingen av systemet og gjennomplanlegging av systemet har vi identifisert problemer som kan oppstå både med systemet og rundt systemet. Ved å vite hvilke riskoer som kan oppstå har vi prioritert de viktigste problemene først og funnet risikoer som må bli undersøkt ved en eventuelt videreutvikling av systemet. Ved å gi en god oversikt over både fysiske risikoer og digitale risikoer innen for programvaren kan det hjelpe personer som skal jobbe videre med systemet.

Risikoer innen for maskin er problemer som kan være strukturelle angående styrken på delene eller sikkerheten rundt systemet. Disse risikoene er generelt lettere å kvantifisere for å teste om risikoen er verdt å ta. Digitale risikoer innen for programvare har ofte med design å gjøre og om valgene som har blitt tatt er de riktige og om ytelsen til systemtet kommer til å bli tungt påvirket av designvalgene som er tatt.

Vi har sett at noen av risikoene som har blitt tidligere identifisert har hatt større påvirkning på gruppearbeidet enn det som var forventet. Problemene angående COVID-19 hadde større påvirkninge på gruppearbeidet enn det vi kunne forutse så tidelig i prosjektet. Vi hadde planlagt hvordan vi skulle jobbe når vi måtte jobbe hjemme fra, men det problemer som oppstå var vanskeligere å løse når vi ikke kunne jobbe fysisk sammen.

3 Teknisk - Maskin

3.1 Vurderingsparametere

Prosessen startet med å undersøke de forskjellige mekanismene som vi kunne bruke for å løse problemstillingen. Ønskede parametere ble brukt som utgangspunkt i undersøkelsen av løftemekanismer og pugh matrix ble brukt for å visualisere fordeler og ulemper ved hver av de forskjellige mekanismene. Parametrene som ble satt var:

Vurderingsparametere
Løftelengde
Løftekapasitet
Kapitalkostnad
Vedlikeholdskostnad
Kompleksitet av vedlikehold
Korrosjonsbestandighet
Energiforbruk
Lengden på en livvsyklus
Sikkerhet for arbeideren
Testing
Materialtretthet
Støynivå

Tabell 1: Vurderingsparametere

Målet med vurderingsprosessen var å finne en mekanisme som var best egnet til å kunne oppfylle kravane fra Tronrud Engineering. Etter prosessen hadde begynt, fikk vi informasjon at Tronrud Engineering vil helst ha elektriske motorer i produktet. Hydraulikk og pneumatikk ble da uaktuelt. Etter modellering av de andre konseptene, kom vi frem til at kjeder og kjedehjul hadde mest balanse mellom pris, kompleksitet og ytelse. Derfor ble kjeder og kjedehjul konseptet som ble utviklet frem til april. Grunnen til forandring av konseptet i april er forklart senere i rapporten.

Følgende konseptene ble vurdert:

- Hydraulikk
- Pneumatikk
- Stativ og tannhjul
- Ledeskrue
- Glidelåskjeder
- Kjede og kjedehjul

Referer til avsnitt (4-Tidligere iterasjoner) i vedlegget for mer detaljert informasjon om disse konseptene.

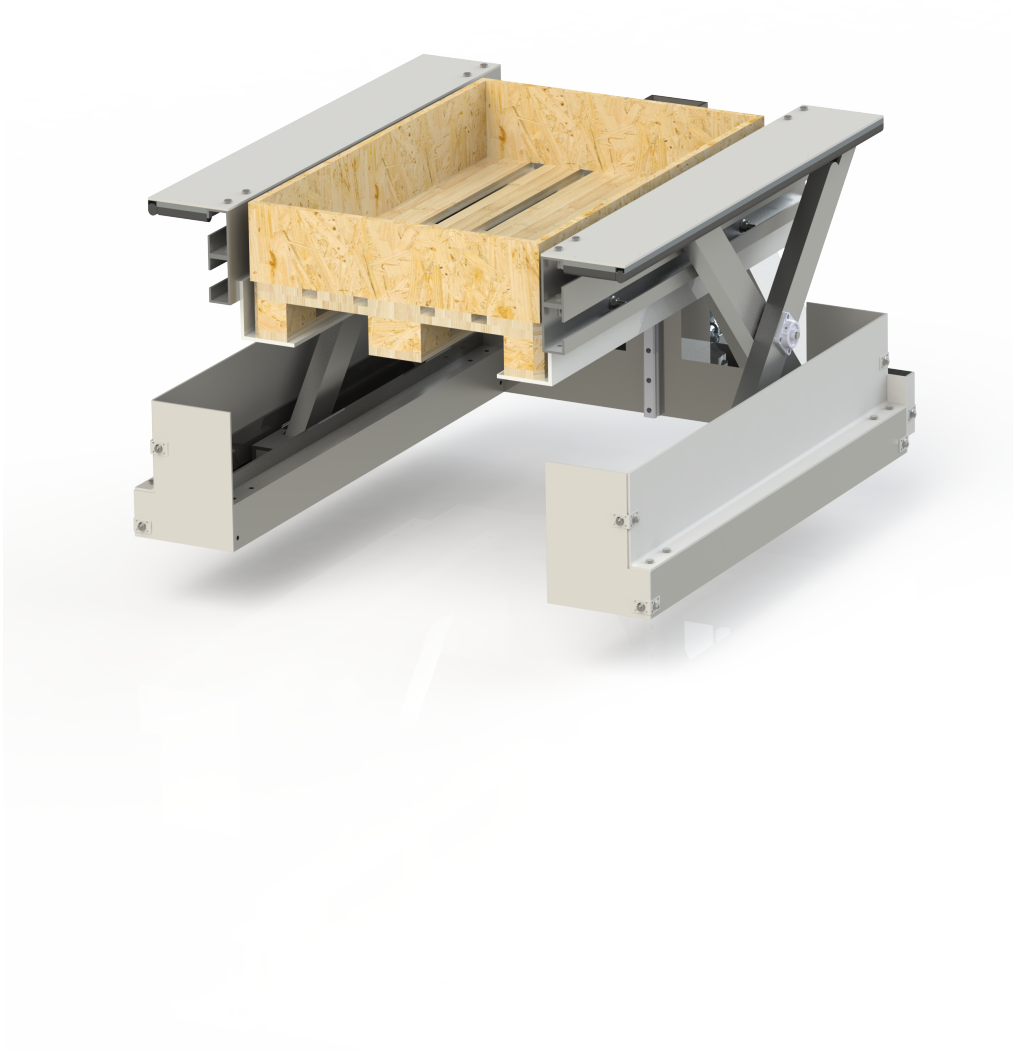
3.2 Pugh-matrise av løftemekanismer

Vi har laget en Pugh-matrise til å visualerisere fordelene og ulempene med de forskjellige alternativene vi har vurdert. Vi har ikke brukt tallene fra pugh-matrisen til å velge et konsept, men som et verktøy til å kunne sammenligne og visualerisere konseptene.

Parametere	Pneumatikk	Hydraulikk	Stativ og pinjong	Leadskrew	Glidelåskjeder	Kjeder og tannhjul
Korrosjon	4	3	2	4	4	2
Løftelengde	3	3	4	4	5	4
Løftekapasitet	3	5	4	4	5	4
Kapitalskostnad	4	2	3	3	1	4
Vedlikeholdskostnad	3	4	2	2	3	2
Vedlikehold	4	3	3	4	3	2
Livssyklus	3	4	3	3	4	3
Energiforbruk	4	3	2	4	4	3
Utmatting	3	3	2	3	3	2
Kompleksitet av produksjon	4	2	3	4	2	5
Sikkerhet	4	4	3	3	4	3
Testing	5	4	3	4	2	3
Lydnivå	3	4	1	2	5	3
Plass	3	3	3	3	4	3
SUM	50	47	38	47	49	43

Tabell 2: Pugh-matrise av løftemekanismer

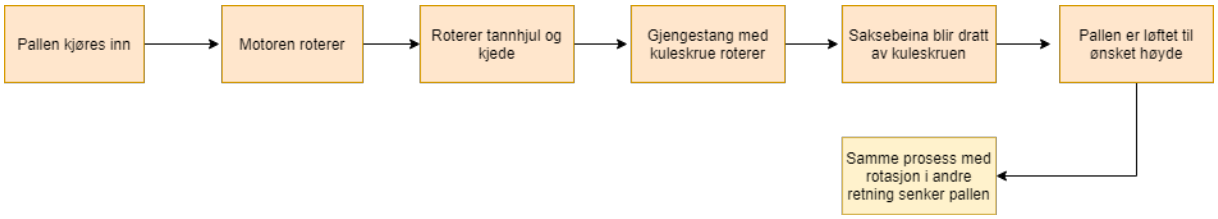
3.3 Valgt konsept



Figur 2: Ferdig produkt

3.3.1 Introduksjon av valgt konsept

Hensikten med den mekaniske delen av produktet vårt er å løfte en pall med maksimum last av 500kg. For at systemet skal inneha en sikkerhetsmargin har vi valgt å designe et system som tåler det dobbelte. Dette avsnittet skal forklare hovedfunksjonene og løsningene brukt i produktet slik at alle krav fra kunden blir møtt. Utdypende informasjon av komponenter presenteres senere i underkapitlene.



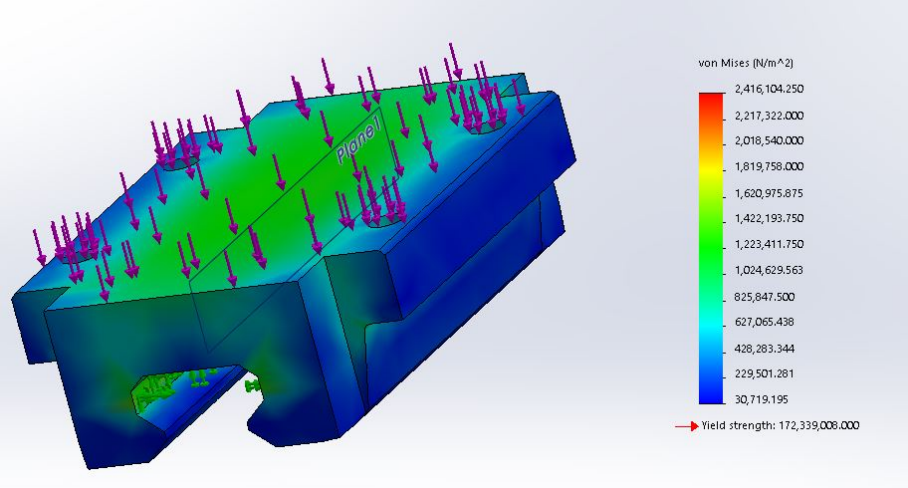
Figur 3: Enkel forklaring av den mekaniske delen av produktet

Som fremstilt i figur 3, så vil løftemekanismen begynne når motoren roterer et tannhjul som deretter driver et kjede. Kjedet er koblet til et tannhjul på hver side av pallen, som er festet til gjengestenger. Disse gjengestengene vil gjøre om rotasjon til en horisontal lineær bevegelse ved hjelp av kuleskruer. Kuleskruene er festet til bunnen av saksebeina. Dette fører til at den nederste delen av saksebeina forflytter seg horisontalt og den øverste delen er montert fast i kulelagre. Pallen blir dermed løftet opp. Ved senking av lasten vil motoren rotere gjengestengene i motsatt retning. Vi har utviklet to løftesystemer. Et system som løfter en pall med deler, og et annet som skal løfte en tom pall. Lagerarbeideren skal derfor få lett tilgang på delene i skanneprosessen og legge delene som er ferdig skannet i den tomme pallen.

Nøyaktighet av styrkeberegning

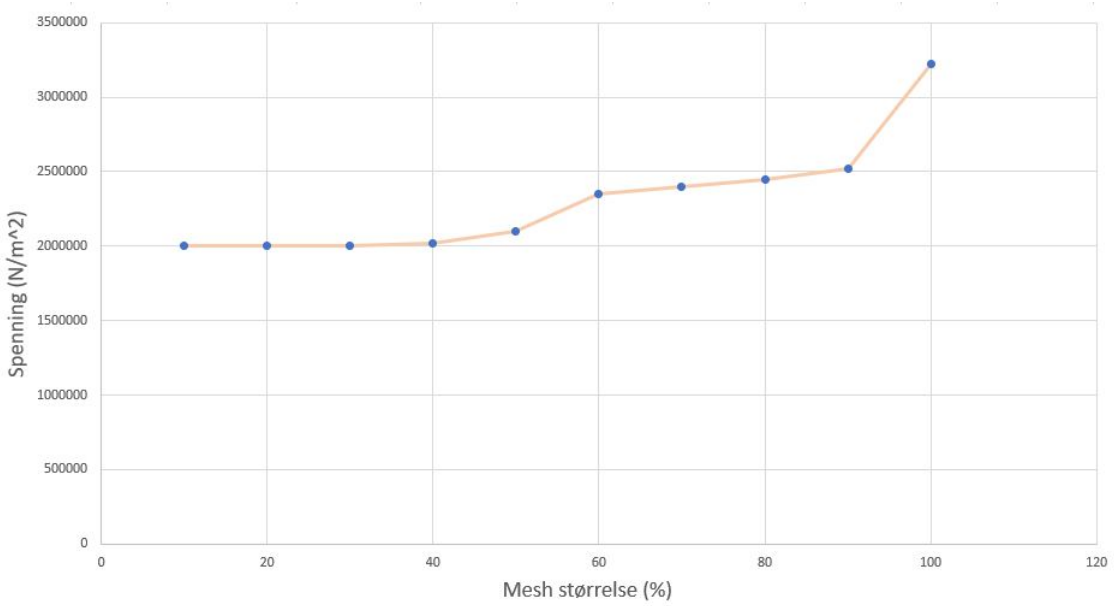
Vi har valgt å gjennomføre en styrkeberegning på hver enkelt komponent og i diverse sammenstillinger. Som en fremstilling av produktet viser vi styrkeberegningene i en sammenstilling for å gi leseren av dette dokumentet en bedre oversikt over hva vi gjør styrkeberegninger på. Solidworks Simulation er brukt til styrkeberegninger av deler i dette prosjektet. Til tross for at Solidworks er et veldig godt verktøy, kan nøyaktigheten av resultatene variere. Følgende steg har blitt tatt til å øke nøyaktighetsgraden av resultatene og samtidig redusere tiden som blir brukt på styrkeberegninger.

Før simuleringen av en del begynner vil programmet dele opp den ukjente delen i mange små pyramider. Programmet beregner spenningene i alle disse pyramidene og legger de sammen til et endelig resultat. Metoden kalles "Finite Element Method" (FEM). Størrelsen på disse pyramidene kan vi variere selv. I Solidworks heter dette "Mesh Size". Ved å variere størrelsen av pyramidene kan vi forandre nøyaktighetsgraden av resultatene og redusere tid som blir brukt av Solidworks på analysering. Optimalt vil innstillingen til de forskjellige delene variere og derfor har vi kjørt følgende metode for å redusere tidsbruk uten å få dårligere nøyaktighetsgrad.



Figur 4: Styrkeberegning av vogn i lineære skinner føring

Vi har brukt vognen fra den lineære føringen som et eksempel (Figur 4). Ved å gjennomføre flere analyser på samme del med varierende mesh-størrelse kan vi se på forskjellene i resultater. Etter at mesh-størrelsene har blitt små nok, kommer resultatene til å stabilisere seg. Det vil si at nøyaktighetsgraden av resultatene ikke øker om vi fortsetter å redusere mesh-størrelsen. Som vist i figur 5 vil makspenningen stabilisere seg på rundt 50 prosent mesh-størrelse. Dette betyr at vi får nøyaktige svar ved denne mesh-størrelsen. Ved å redusere størrelsen enda mer hadde det bare ført til mer tidsbruk.



Figur 5: Styrkeberegning av vogn i lineære skinner føring

3.3.2 Komponenter og hyllevarer

Delnr	Navn på del	Antall	Materiale	Vekt
1	Aluminiumsvinkel	1	6082-T6	8,12 kg/m
2	Aluminiumsplate	1	6082-T6	77,10 kg
3	Hulprofil (Stål)	1	S355J2H	15 kg/m
4	Hulprofil (Stål)	1	S355NH	16,8 kg/m
5	Hulprofil (Stål)	1	S355NH	37,9 kg/m
6	Akselstål	1	S355J2	10 kg/m
7	Flenslager	2	Flere	0,6 kg
8	Fotlager	2	Flere	0,6 kg
9	Hjul	2	Flere	1,2 kg
10	Bronseforinger	6	Rødmetall	-
11	Låseringer	20	Rustfritt stål	-
12	Kjede	1	Rustfritt stål	2,7 kg/m
13	Vinkelbeslag	12	Galvanisert stål	-
14	Egenprodusert del (ved kuleskruene)	2	Aluminium	-
15	Kjedehjul	4	Rustfritt stål	0,7 kg
16	Motor	1	Flere	39 kg
17	Lagerbukker til kuleskrue	4	-	2 kg
18	Mutter til kuleskrue	2	-	-
19	Mutterhuset til kuleskrue	2	-	4,7kg
20	Gjengestang	1	-	-
21	Lineær skinne	1	-	6 kg/m
22	Vogn til skinne	1	-	2 kg
23	Klemlister	1	Flere	0,75 kg/m
24	Kulelager til kuleskruene	2	-	0,9 kg
25	Egenprodusert del (motorfeste)	1	Aluminium	-

Tabell 3: Komponentliste

Delnr	**Type**	**Antall**
1	M10 - 20mm - Bolt	32
2	M16 - 55mm - Bolt	4
3	M8 - 30mm - Bolt	4
4	M12 - 45mm - Bolt	20
5	M12 - 80mm - Bolt	4
6	M10 - Mutter	52
7	M12 - Mutter	24
8	M16 - Mutter	4
9	M4 - 10mm - Bolt	18
10	M10 - 40mm - Bolt	8
11	M8 - 35mm - Bolt	10
12	M12 - 100mm - Bolt	8
13	M10 - 40mm - Bolt	16

Tabell 4: Tabell av bolter og muttere

I tabell 3 og 4 har vi fullstendige tabeller med alle komponentene som inngår i produktet vårt. Felter som står tomme betyr at vi ikke har funnet data på det spesifikke området. En mer omfattende komponentliste (Vedlegg - Kapittel 6) har vi laget i Excel og kan sees i vedlegget. Hver tabell har som utgangspunkt i en enkelt løftemekanisme og hvis det skal bestilles to løftemekanismer må det dobbelte bestilles. Under "Antall" menes det antall som må bestilles og ikke nødvendigvis antall som inngår i produktet. De fleste delene som inngår i produktet må maskineres forskjellig.

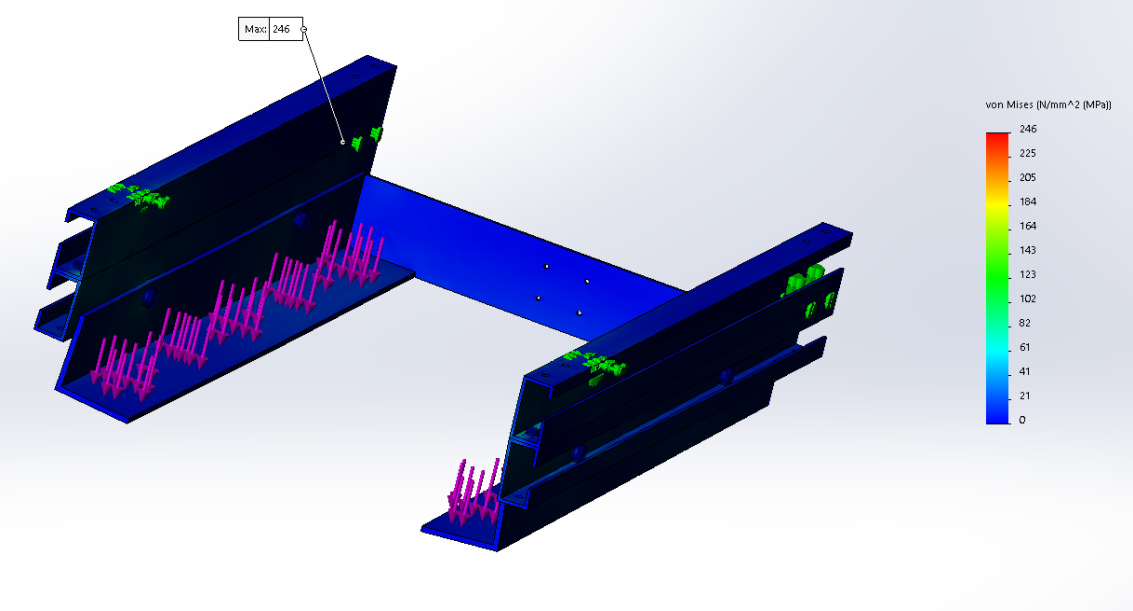
3.3.3 Sammenstilling av pallholder

Denne sammenstillingen er satt sammen av en vinkelprofil og plate av aluminium, samt en hulprofil laget av stål. Vinkelprofilene som er brukt i produktet, er en slags plattform hvor pallen skal sitte. Dette er på grunn av et krav fra bedriften som sier at trallen (med pall) ikke skal kjøres opp på en rampe eller plattform. Ved bruk av vinkelprofiler kan vi kjøre pallen rett inn i systemet, uten å kjøre trallen opp på en plattform. Stålprofilene behøver maskinering først og fremst for å lage spor der hvor saksebeina skal bevege seg. Her kunne vi brukt U-profiler istedenfor hulprofiler, men vi ser på hulprofiler som en bedre løsning i denne sammenhengen. Dette er fordi vi vil ha mulighet til å maskinere presise spor uten at hjulet som skal inn i stålprofilen faller ut når hele sammenstillingen er ferdig. I tillegg til dette er vi avhengige av å å maskinere ut boltehull til både hulprofilen og vinkelprofilen for å bolte de sammen. Hensikten med aluminiumsplaten som holder vinklene sammen er å unngå for mye bøyning i hele strukturen, samtidig som at denne kan brukes til å festes til en lineær føring som vil føre til at hele strukturen blir mer motstandsdyktig mot moment inn fra sidene på hele strukturen. Dette blir skrevet om i (3.3.6). Denne aluminumsplaten må maskineres ut ifra en større aluminiumsplate som finnes i komponentlisten (3) og deretter sveises på aluminiumsvinklene.

I denne sammenstillingen er det viktig å ta en styrkeberegning på alt som en sammenstilling og ikke bare som enkeltkomponent. Dette er først og fremst for å finne ut av hvordan bøyningene i strukturen vil fungere når komponentene er i relasjon til hverandre. Disse bøyningene er en viktig faktor for at delsystemet ikke skal sette seg fast i andre deler av systemet. For å simulere vekten til pallen med deler er vi nødt til å legge sammen alle kreftene som vinkelprofilen vil bli påvirket av. I og med at Astrup ikke oppgir vekten til aluminiumsplaten, men oppgir vekten til aluminiumsvinkelen, går vi ut ifra at de veier det samme (med samme dimensjoner) i og med at det er det samme materialet (6082-T6). Dette vil ikke bli helt nøyaktig i og med at aluminiumsplaten er tynnere enn vinkelen, men vi anser det som bedre at vi overdimensjonerer litt i beregningen. Tronrud oppgir at pallen med pallekarm veier omtrent 25kg i tørre omgivelser.

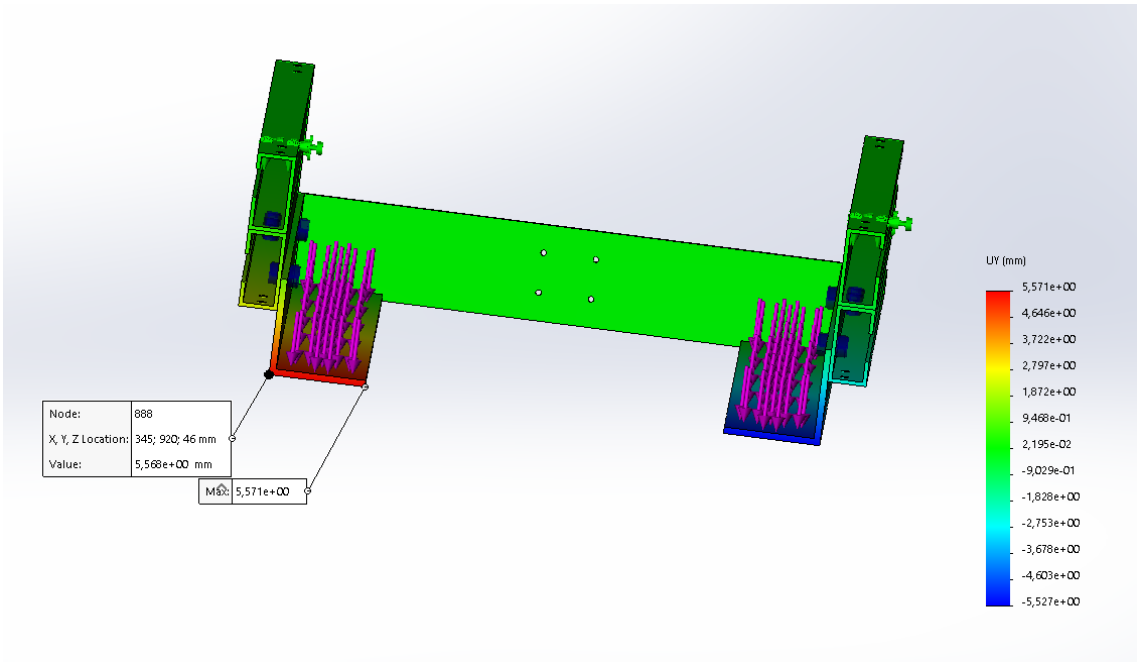
$$F = (m_{deler} + m_{pall} + m_{vinkelprofil}+) * g = (1000kg + 25kg + 12kg) * 9,81\frac{m}{s^2} = 10173N \qquad (1)$$

Denne krafta blir påført på samme måte som vi tenker oss at pallen skal stå på vinkelprofilen og blir vist som de lilla pilene på figur 6



Figur 6: Resultat av styrkeberegning på palleholderen

I denne simuleringen blir hullene hvor flenslageret er koblet til satt som "fixed" og det samme med området hvor overflaten til hjulene treffer hulprofilen. Overordnet kontakt i analysen er "Bonded". Kontaktoverflatene i sammenstillingene vil fiksjonelt være "limt" sammen. "Contact Set" mellom overflatene til hulprofilene og vinkelen blir satt til "No penetration". Palleholderen skal ifølge Solidworks klare å løfte pallen med deler, men vi ser at det blir en del forskyvning. Dette valgte vi å se nærmere på for å unngå at systemet setter seg fast ved en belastning på 1. tonn.

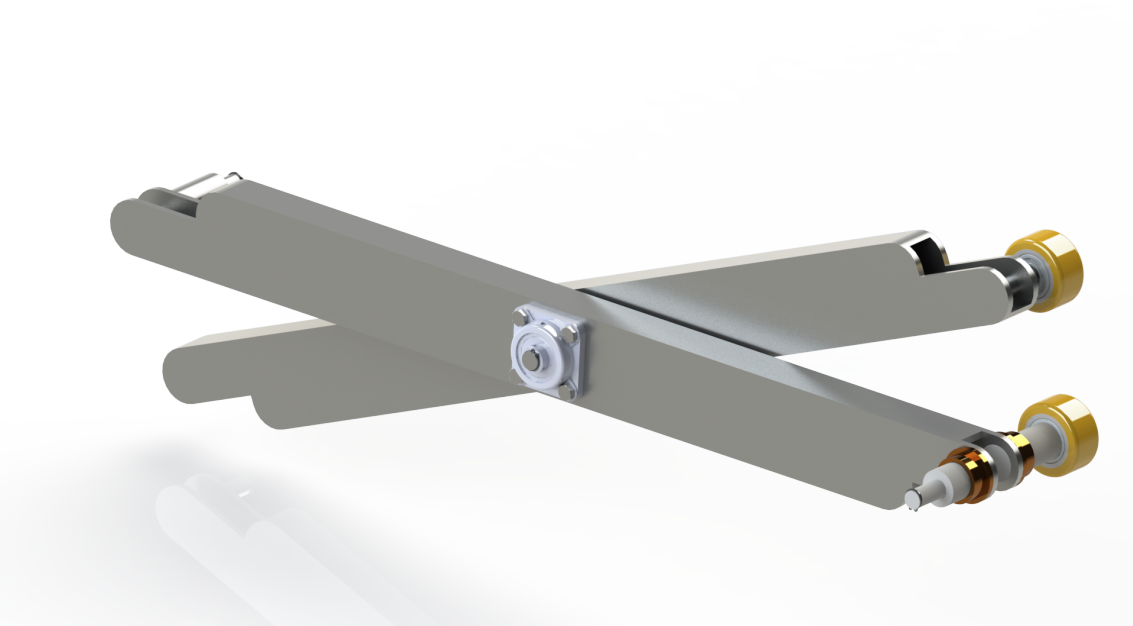


Figur 7: Forskyvning palleholderen i x-aksen under maksimal belastning

Av denne simuleringen ser vi at vi må ta høyde for at pallholderen vil bevege seg litt i x-aksen under maksimal belastning. Dette har vi tatt høyde for i videre utvikling av konseptet.

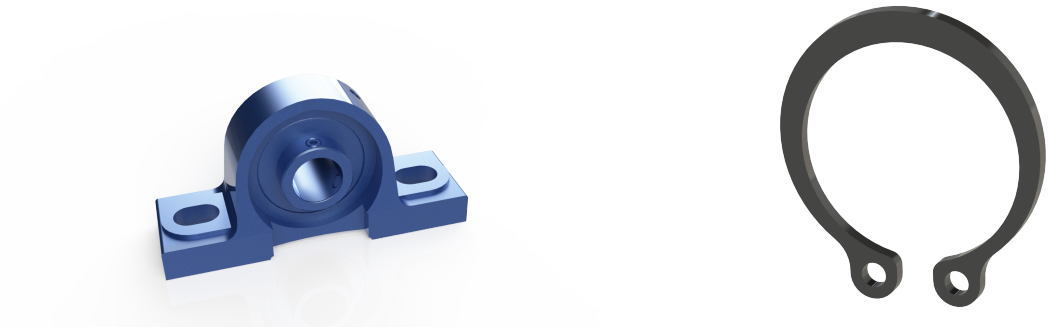
Begrunnelsen for at vi bruker 2 hulprofiler under hverandre henger sammen med kritisk vinkel som vi skriver om i avsnitt (3.3.4). Dette gir forklaring på hvorfor vi har valgt de profilene av hulprofiler vi har gjort i sammenstillingen av pallholderen. I tillegg til den kritiske vinkelen, gir hulprofilene oss høyden vi ønsker i forhold til at systemet ikke skal stå i veien for arbeideren. Hvis vi bolter sammen 2 stykker av samme profil under hverandre gir det oss en høyere startvinkel i tillegg til fordelen med at vi kan feste vinkelprofilen i hulprofilen hvor det ikke roterer eller beveger seg noen deler. Dette kunne gitt problemer for bevegelsen av hjulene i hulprofilen. I tillegg til dette var de sterke nok til å løfte pallen ifølge Solidworks-analysen, uten at de legger til for mye vekt på systemet.

3.3.4 Sammenstilling av saksebein



Figur 8: Modell av saksebein

Denne sammenstillingen består av en ny hulprofil av stål. Denne hulprofilen er ikke den samme som ble brukt som spor i pallholderen. Vi ønsker at profilene skal være så smale som mulig for at vi skal holde oss innenfor de ergonomiske standardene som har blitt satt av arbetsmiljøverket i Sverige [29]. De forrige profilene var litt for store med tanke på at vi ønsker at arbeideren skal være så nærme pallen som mulig. Vi ønsket en helt vertikal bevegelse i saksebeina som gjør at hjulene vil bevege seg i horisontal retning. Det gjør ikke den andre siden av saksebeina (venstre side på figur 8). Disse akslingene er koblet fast til et fotlager som er skrudd fast i den øverste hulprofilen i pallholderen, som vil sørge for at akslingene kan få rotere fritt, men ikke ha noen forflytning i posisjon. Dette fotlageret har navnet UCP-204. Alle akslinger med muligheter for at akslingene skal skli ut av posisjon har låseringer på for å sørge for at dette ikke skal skje.

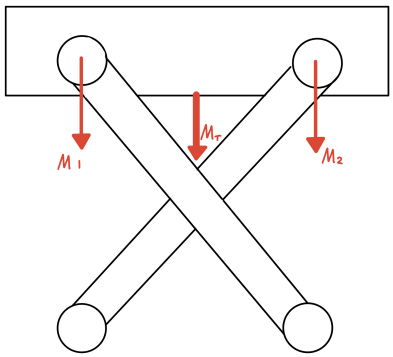


(a) Fotlager som er tegnet i Solidworks

(b) Låseringer som er tegnet i Solidworks

Figur 9: Komponenter som er tegnet i Solidworks

Fotlagrene og hjulene i sammenstillingen må tåle belastningene som pallen påfører systemet. Fotlageret i denne sammenstillingen er produsert for å tåle store radielle krefter. Dette kulelageret kan tåle en belastning på 12,8kN i dynamisk radiell kraft og 6,65kN i aksielle krefter, ifølge de tilhørende databladene [30]. Den radielle kraften vil være det som klarer å løfte pallen opp fra bakken. 12,8kN tilsvarer 1305kg på jordoverflaten, så noen vil nok argumentere for at dette er en overdimensjonert del i systemet vårt. Problemet kommer når akslingen som passer i disse kulelagrene kommer inn i beregningene. I en UCF-204 passer det bare med en 20mm aksling, som vil være svakheten til sammenstillingen. Motstandsdyktighet mot aksielle krefter er også viktige i denne sammenhengen. Dette er for å sørge for at sammenstillingen av pallholderen ikke faller av saksebeina. Hjulene skal ifølge sitt datablad ha en belastningskapasitet på 280kg [31].



Figur 10: Enkel beskrivelse av hvordan kreftene til pallen påvirker saksebeina

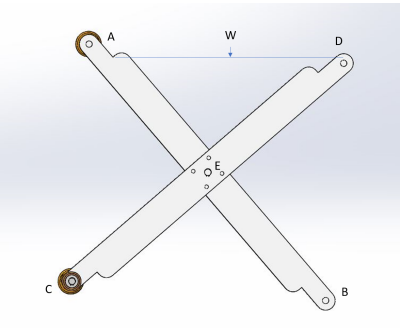
Hvis vi tenker oss at at vi har et saksebein på hver side av pallen og at pallen med deler veier 1000kg blir M_T i tegningen lik 500kg og fordeler seg jevnt utover i M_1 og M_2. Dette tilsvarer 250kg på hver side, som gjør at både hjulet og fotlageret tåler kreftene som blir påført under maksimal last. Den samme monteringen vil skje nederst på saksebeina, som gjør at kreftene vil oppføre seg helt likt nede. Denne maksimale lasten er

heller ikke realistisk med tanke på at Tronrud har gitt oss beskjed om at hver pall veier i snitt rundt 300kg. Disse utregningene gjør at vi kan se bort ifra hjulene og fotlagrene i Solidworks-analysen.

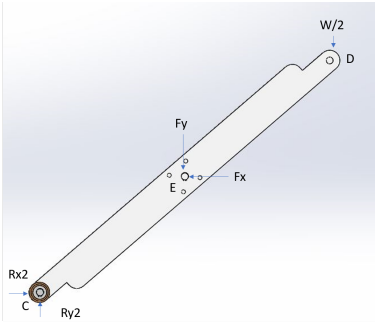
Saksebeina inneholder også tre bronsjeforinger per saksebeinpar. To av de er synlige i figur 8. Disse Foringene er der for å føre kraften fra kuleskruene til saksebeina. Dette blir gjort ved at to egenproduserte deler av aluminium vil bli festet til kuleskruene med en form som gjør at bronseforingene kan monteres imellom denne delen og akslingene nederst ved hjelp av "shrink-fitting". En bronseforing er også på innsiden av saksebeina i rotasjonspunktet til saksebeina (se exploded view - 2D-tegning). Dette er for å sørge for at det innerste og ytterste saksebeinet holder avstand fra hverandre. Hovedfunksjonen til foringa er først og fremst å gjøre friksjonskoeffisienten mellom saksebeina lavere. Dette gjør at de roterer lettere rundt rotasjonspunktet. Disse bronseforingene får støtte av enda et kulelager som også er synlig i figur 8. Dette er en UCF-204. Dette kulelageret holder hele sammenstillinga til saksebeina sammen ved at en aksling som er sveiset til innsiden av det ene saksebeinet går igjennom hele sammenstillinga og er festet til dette kulelageret. Det vil si at hvis saksebeina blir dratt fra hverandre vil majoriteten av disse kreftene bli tatt opp som aksiell last i kulelageret. For å unngå at denne bevegelsen skjer har vi satt på en lineær føring som det står skrevet om i avsnitt 3.3.6.

Beregning av nødvendig kraft for å løfte pallen

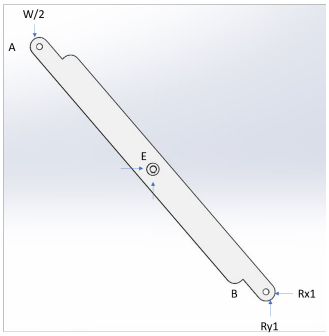
For å dimensjonere produktet må vi vite den horisontale kraften som er nødvendig for å løfte pallen når den er maksimalt belastet. Denne prosessen begynte med å lage et fritt-legeme-diagram som kan sees i figur 11a. Et fritt-legeme-diagram gir oss en oversikt over alle kreftene som påvirker saksebeina. Deretter blir diagrammet delt opp i to hvor hvert av de representerer et saksebein (Figur 11b og 11c).



(a) Fritt-legeme-diagram av begge saksebeina



(b) Fritt-legeme-diagram av det ene saksebeinet



(c) Fritt-legeme-diagram av det andre saksebeinet

Figur 11

Sum av krefter og moment rundt punkt B på venstre saksebein

$$M_b = \frac{W}{2} \times 2L \times cos(\theta) - F_y \times L \times cos(\theta) - F_x \times L \times sin(\theta) \tag{2}$$

$$\sum(F_x) = F_x - R_x1 \tag{3}$$

$$\sum(F_y) = F_y + R_y1 - \frac{W}{2} \tag{4}$$

Sum av krefter og moment rundt punkt C på høyre saksebein

$$M_a(C) = \frac{W}{2} \times 2L \times cos(\Theta) - F_y \times L \times cos(\Theta) + F_x \times L \times sin(\Theta) \tag{5}$$

$$\sum(F_x) = R_x2 - F_x \tag{6}$$

$$\sum(F_y) = R_y2 - F_y - \frac{W}{2} \tag{7}$$

Nå har vi 6 likninger og 6 ukjente, ved å løse de får vi:

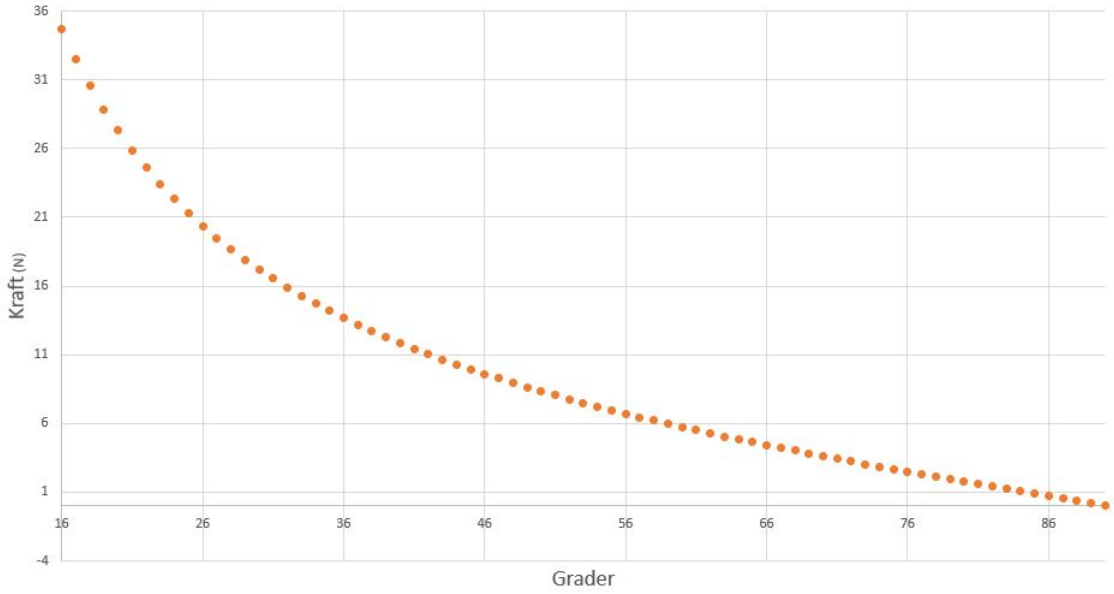
$$(F_x) = R_x1 = R_x2 = \frac{W}{tan(\Theta)} \tag{8}$$

Vinkelen mellom saksebeina og bakken, når pallen er i startposisjon, endte opp med å bli 16,32° når vi brukte de hulprofilene vi har valgt. Ved å bruke likning 8, kan vi finne ut hvor mye kraft kuleskruene må yte for å løfte pallen. Tabell 3 forteller oss hvor stor W blir per saksebein:

$$W = (m_{deler}+m_{pall}+m_{vinkelprofil}+m_{hulprofil})*g = (500kg+25kg+12kg+42kg)*9,81\frac{m}{s^2} = 5680N \tag{9}$$

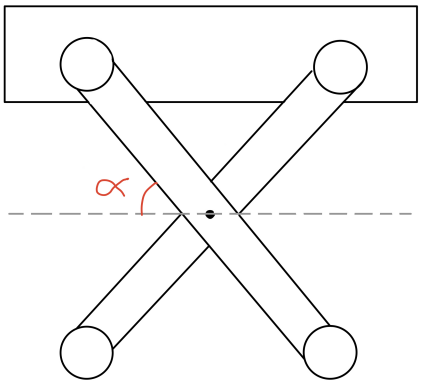
$$(F_x) = R_x1 = R_x2 = \frac{W}{tan(\Theta)} = \frac{5680N}{16,32°} = 19400N \tag{10}$$

Ved bruk av samme likning er det mulig å vite hvordan kraft minker ved forskjellige vinkler (Figur 12) Figuren viser at maksimum kraft er nødvendig for å løfte pallen når den er på bunnen og minker ved høyere vinkler. Derfor brukte vi 20kN som kraftverdien til å definere kuleskrue-systemet. Ut ifra maksimum verdien (20kN) blir nødvendig moment for å rotere kuleskrue beregnet.



Figur 12: Foranding av horisontal kraft nødvending for å løfte pallen ved varierende vinkel av saksebein

Disse utregningene forteller oss at vinkelen mellom saksebeina er kritisk for at kreftene som belaster systemet blir så lave som mulig. Samtidig som at vi ikke ønsker at startposisjonen blir for høy fordi at vi ønsker ikke at systemet står i veien for arbeideren. Dette var hele grunnen til at vi forkastet konseptet med kjeder og tannhjul. Derfor tenkte vi oss at et system hvor det høyeste punktet i løftemekanismen er på samme høyde som pallekarmen hadde vært det optimale.



Figur 13: Kritisk vinkel på saksebein

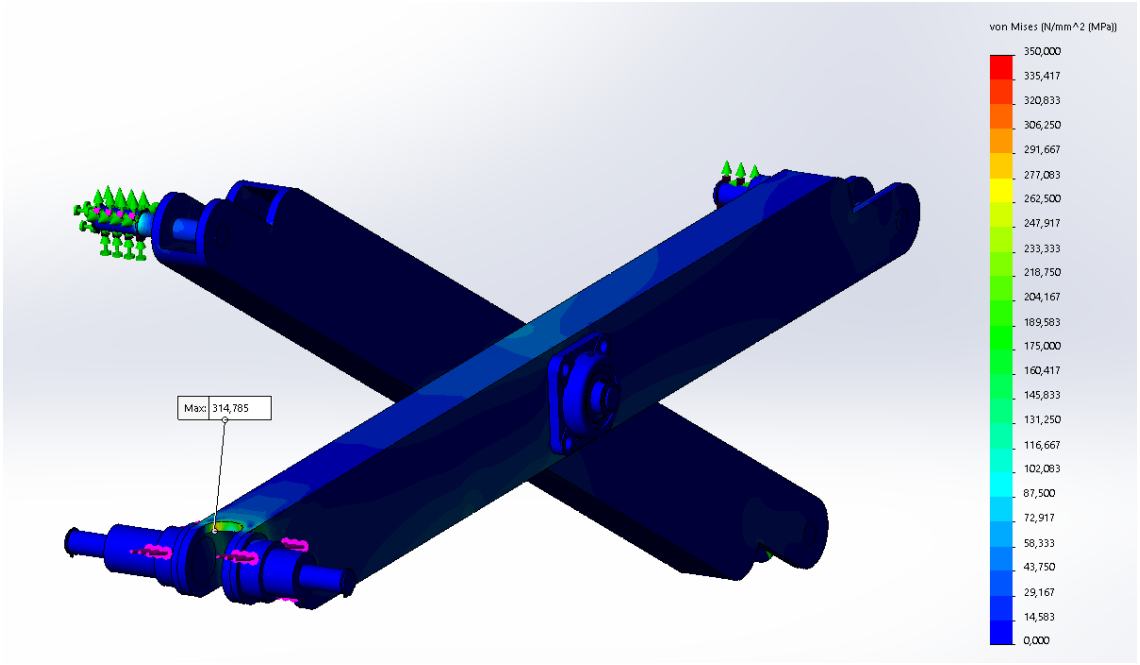
Denne kritiske vinkelen er også grunnen til at vi har valgt å maskinere som vi har gjort på hulprofilene til saksebeina. Maskineringa av hull til montering av hjul og fotlagere sammen med saksebein gjør at avstanden imellom det øverste og nederste saksebeinet blir litt lengre i starten av løftebevegelsen. Dette gjør at vinkelen blir noen grader større i forhold til om vi hadde plassert hjulene sentralt på hulprofilen.

Vinkelen imellom saksebeina og bakken, når pallen er i startposisjon, endte opp med å bli 16,32°. Dette er et resultat av hulprofilene som vi syntes passet inn med resten av systemet. Dette er en relativt lav vinkel i forhold til hva andre i industrien produserer men dette er noe vi ønsker for å gi best mulig tilgang til pallen for lagerarbeideren. Dette betyr at det vil bli mer belastning på saksebeina, motor og kuleskruene, som vi endte opp med at skulle drive saksebeina.

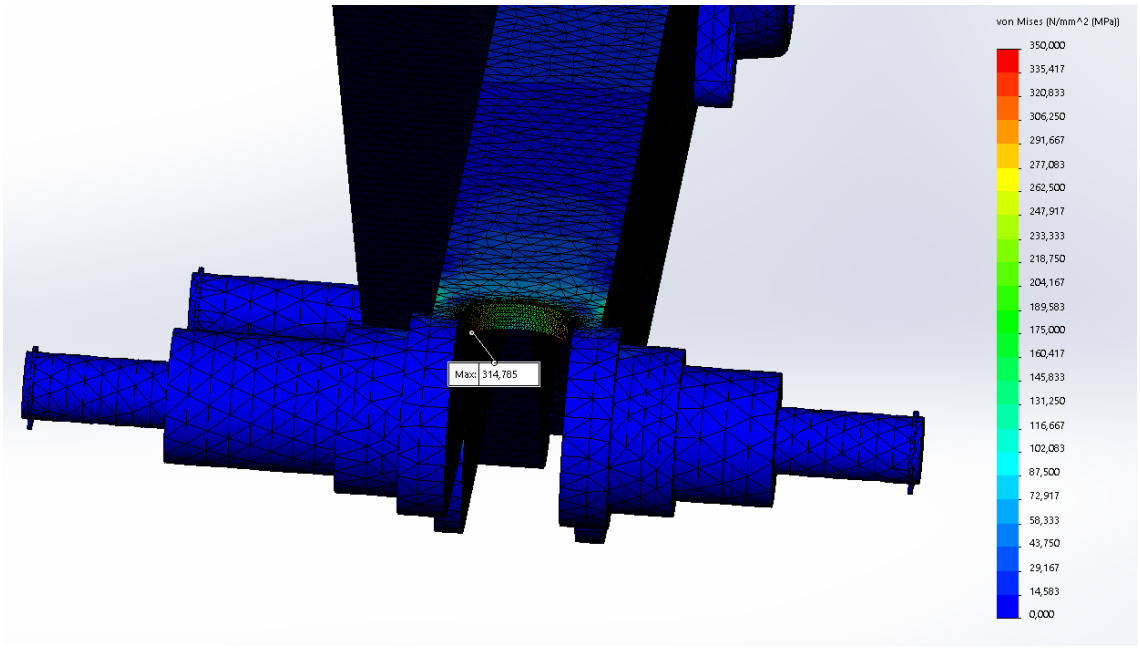
Etter flere analyser kan vi se at spenningene samler seg opp nederst ved den ene hulprofilen (se figur 14 og 15). Dette gjør at vi vil ha en mesh-forfining som kontrollerer hver millimeter rundt dette punktet. Vi bruker også mesh-forfining på akslingene. Mesh-størrelsen vi bruker i saksebeina generelt ser ut som dette:

Study name	Static 1 (-Høyre-)
Mesh type	Solid Mesh
Mesher Used	Curvature-based mesh
Jacobian points	4 points
Mesh Control	Defined
Max Element Size	7,37195 mm
Min Element Size	2,73815 mm
Mesh quality	High
Total nodes	285520
Total elements	167461
Maximum Aspect Ratio	40,627
Percentage of elements with Aspect Ratio < 3	96,4
Percentage of elements with Aspect Ratio > 10	0,528
% of distorted elements (Jacobian)	0
Remesh failed parts with incompatible mesh	Off
Time to complete mesh(hh:mm:ss)	00:00:15
Computer name	

Tabell 5: Mesh-størrelsen som ble brukt i styrkeberegningen

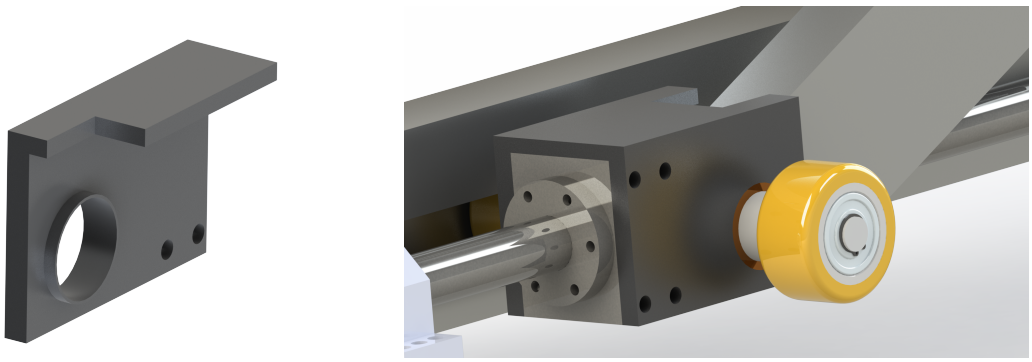


Figur 14: Styrkeberegning på saksebeina



Figur 15: Spenningskonsentrasjoner på saksebeina med meshforfining

Kraften er satt på bronseforingene med de egenproduserte delene av aluminium (se figur 16) . Den blir festet i samme retning som planet som ligger imellom de to saksebeina. I startposisjon er vinkelen som nevnt 16,32°. Dette blir derfor vinkelen som vi setter imellom saksebeina og dette planet. Etter beregningene av sammenhengen imellom kraft og vinkel (likning 10) blir kraften maksimalt 20kN i starten av bevegelsen når pallen veier 1 tonn. Dette er grunnen til at størrelsen til kraften som er satt på bronseforingene har denne størrelsen. Fixtures har vi satt som "Fixed hinge" på akslingene hvor det er skrudd på kulelagere og der hvor hjulene er montert. Overordnet kontakt i analysen er "Bonded", men kontakt imellom kulelagre (UCF-204 og bronseforingene) og akslinger er satt som "No penetration". Akslingene vil bli sveiset på i hulprofilene som gjør at de også i simuleringene har kontakt som "Bonded".



(a) Egenprodusert del i aluminium

(b) Egenprodusert del i sammenstilling med kuleskruene

Figur 16: Egenprodusert del som brukes til å overføre kraften fra kuleskruene til saksebeina

Figur 14 og 15 viser det endelige resultatet vi kom frem til når pallen med deler har en vekt på 1 tonn. Vi får en maks spenning på 315 MPa, som er under flytegrensa til stålet som blir brukt. På figur 16b kan man se hvorfor vi har valgt å maskinere hulprofilene slik vi har gjort. Vi ønsker å få plass til kuleskruene som skal igjennom hulprofilen for å få fordelt kraften fra kuleskruene jevnt over hulprofilene. En annen grunn til at vi ønsker å føre kuleskruene igjennom og ikke på siden, er for å bruke mindre plass i horisontal retning som kan føre til at vi faller utenfor standardene for ergonomisk håndtering av delene på pallen. En annen ting å bemerke seg er at hulprofilen er maskinert i en avrundet form der hvor kuleskruene passerer igjennom hulprofilen. Dette må gjøres for at spenningskonsentrasjonene ikke blir for høye, som gjør at hulprofilen blir plastisk deformert under maksimal last. Den største utfordringen med å designe saksebeina var å finne den hulprofilen som tålte belastningen men som samtidig tok minst mulig plass.

3.3.5 Kjøremekanisme

Valg av mekanisme

Etter 2.presentasjon måtte konseptet forandres på grunn av stolpene. Disse stolpene ga operatøren mindre tilgang til delene på pallen. På grunn av dette, ble saksebein valgt til å gi best mulig tilgang til delene. En mekanisme som kunne dra saksebeinene måtte fortsatt velges. Siden en del arbeid hadde allerede blitt gjort på kjeder, valgte vi å fortsette med kjede som drar mekanismen. Etter kraften til å dra saksebeinene ble beregnet (likning 10) ble festing av kjede til saksebeina en stor utfordring på grunn av store krefter. På grunn av dette valgte vi å bruke enten kuleskrue eller ledeskrue.

Både ledeskruer og kuleskruer konverterer rotasjon til bevegelse i en lineær retning.[39] Dette fører til at det er nødvendig med mindre moment for å kunne løfte store laster. Ved bruk av disse kan vi få en bevegelse i både vertikal og horisontal retning. I produktet vårt er bevegelse implementert i horisontal retning. Begge mekanismene kan brukes i to orienteringer. En orientering hvor mutteren roterer rundt en skrue som ikke kan rotere, og en orientering hvor skruen roterer og der mutteren er festet til lasten. Dette er orienteringen som blir brukt i produktet vårt.

Begge mekanismene er bygget opp ved bruk av følgende hovedkomponenter:

- Mutter
- Mutterhus
- Gjengestang
- Lagerbukke
- Kulelagere

Hovedforskjellen mellom kuleskrue og ledeskrue er at ledeskruen har direkte kontakt mellom spor på gjengestengene og mutteren. Kuleskruen har kuler som roterer i disse sporene. Denne forskjellen fører til både fordeler og ulemper:

Kuleskrue - fordeler

- Systemet får høyere effektivitet på minst 90 prosent. Dette er på grunn av de kulene som er plassert imellom sporene til gjengestengene. Dette fører til en mindre verdi for friksjonskoeffisient, som da senker kraftforbruket. På grunn av mindre friksjon blir momentet som er nødvendig for å kunne rotere skruen mindre. På grunn av dette kan vi ta i bruk en mindre kraftig motor iforhold til ledeskruemekanismen.
- Produktet får høyere nøyaktighetsgrad på grunn av høyere grader av toleranser ved produksjon av komponentene.
- Kuleskruen kan ta imot tyngre laster.
- Kuleskruen kan operere ved raskere rotasjons hastigheter.

Kuleskrue - ulemper

- Ved bruk at kuleskrue i vertikal retning må man ha bremser for å unngå bevegelse, som går motsatt av ønsket retning, på grunn av tyngdekraften. Motorbremser eller sikkerhetstapper kan fikse problemet. I produktet vårt kan vi se bort fra denne ulempen siden vi har bevegelse i horisontal retning (backlash).
- Mekanismen lager mer støy sammenlignet med ledeskruen. Dette er ikke et stort problem i vårt produkt siden produktet skal plasseres på et verksted med andre store maskiner.
- Smøring ved bestemte tidsintervaller er nødvendig for å ha en lang livssyklus.
- Kuleskruen er dyrere enn ledeskruen på grunn av høyere grad av toleranser ved produksjon.

Ledeskrue - fordeler

- Ledeskruen er billigere enn kuleskruen
- På grunn av høy friksjon er det ikke nødvendig med et bremsesystem til å motvirke "backlash". Det er grunnen til at ledeskrue er brukt oftere ved vertikal bevegelse.
- Ledeskruen lager mindre støy.
- Det er mulig å få tak i automatisk smøring til ledeskrue.

Ledeskrue - ulemper

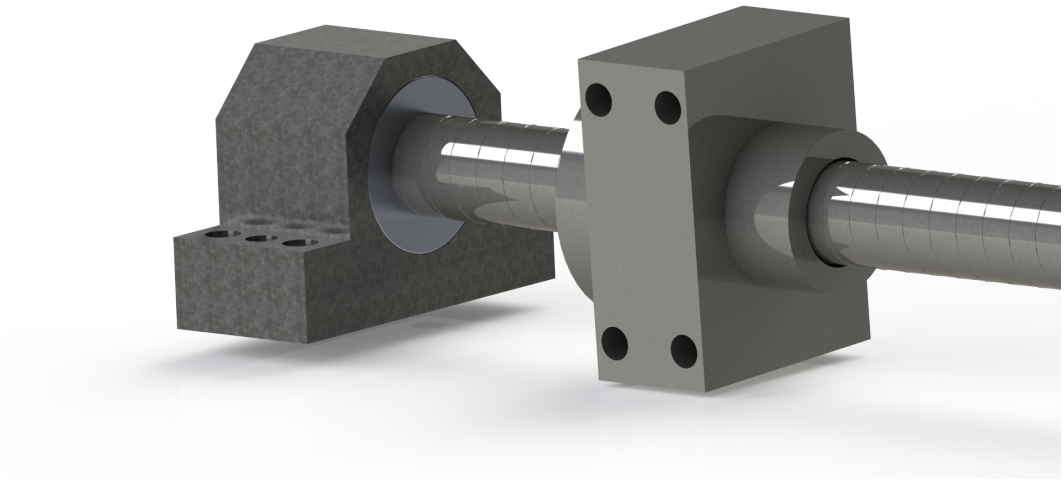
- På grunn av høy friksjon er det nødvendig at motoren klarer å levere et større moment enn hos kuleskruen for å kunne rotere gjengestengene.
- På grunn av høy friksjon opererer ledeskruen ved høyere temperaturer.
- Ledeskruene er dyrere å vedlikeholde over en lang tidsperiode.

Konklusjon av valget

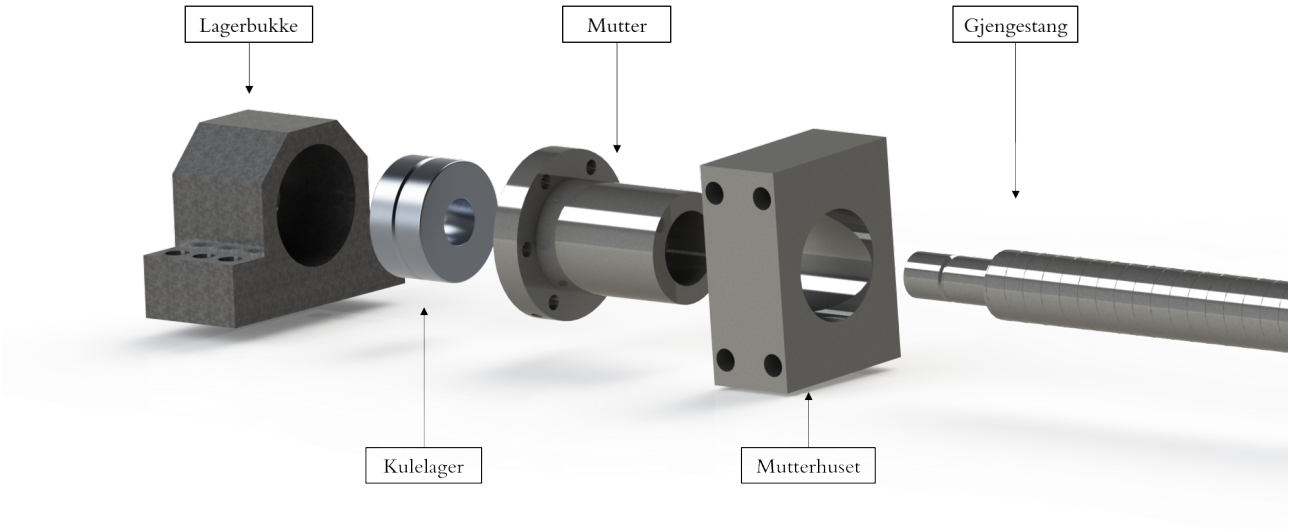
Etter vurdering av fordelene og ulempene ovenfor ble kuleskrue valgt som mekanisme til produktet vårt. Prisforskjellen hadde vært en veldig stor faktor hvis produktet skulle masseproduseres. Siden produktet skal produseres kun en gang valgte vi å se bort fra prisforskjellen. Høyere effektivitet ved bruk av kuleskrue ble prioritert, da det fører til mindre moment på motoren og derfor mindre kraftforbruk. Prisforskjellen blir balansert siden en billigere motor enklere kan rotere en kuleskrue. Dersom bevegelsen var i vertikal retning hadde ledeskrue blitt valgt. På grunnlag av lavere friksjon mellom komponentene får systemet lengre levetid ved bruk av kuleskrue.

Kuleskrue

Valg av komponenter til kuleskruemekanismen begynte med å analysere kreftene som blir påført på sammenstillingen. Den avgjørende faktoren for hvilken kuleskrue vi valgte var komponenter som tåler kraften i horisontal retning. Utifra beregningene med saksebeina ble det beregnet at 20kN var nødvending for å kunne løfte pallen. Rexroth, som er en del av Bosch, er en leverandør som leverer alle de nødvendige komponentene til et kuleskruesystem. Følgende komponenter ble valgt:



Figur 17: Kuleskrue sammensetning



Figur 18: Kuleksrue sammensetning - Exploded view

Mutter

Kulene skal rotere mellom sporene til gjengestangen og mutteren. Figur 17 og 18 viser hvor mutteren skal plasseres. Mutteren er festet videre til mutterhuset med 6 x 6.6mm bolter. Mekaniske egenskaper av skruen som ble valgt til produktet er listet nedenfor [39]:

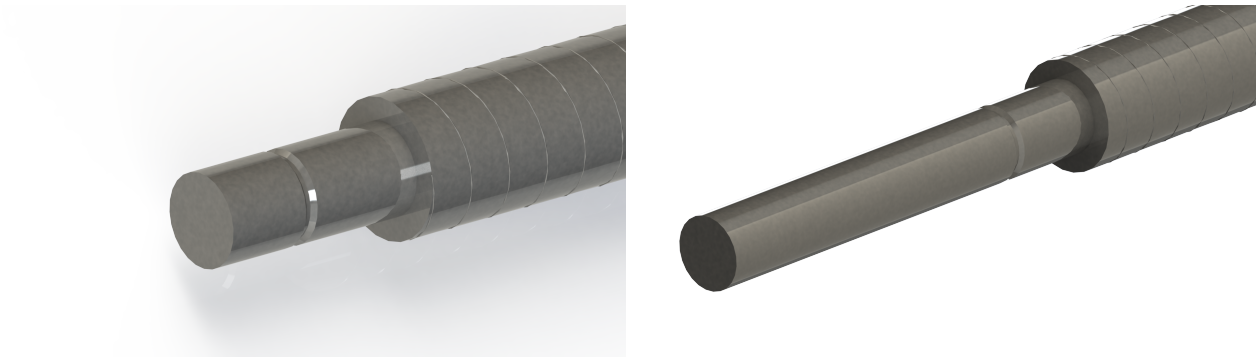
- 32 x 10R x 3.969 - 5
- R1512 340 13
- Dynamisk last: 38,000 N
- Statisk last: 58,300 N
- Maksimum rotasjonshastighet: 47 m/min

Mutterhus

Mutterhuset fungerer som mellomledd mellom mutter og saksebein. Mutterhuset er festet til saksebein ved bruk av 4 x M12 bolter. Mekaniske egenskaper av mutterhuset er listet nedenfor [39]:

- 32x10R x 3.969
- R1506 300 20
- Masse: 2.367kg
- M10 Screw (ISO 4762)
- Hull formasjon: BB4

Gjengestang



(a) Korte enden av gjengestang

(b) Lange enden av gjengestang

Figur 19: Maskinerte endene av gjengestang

Den kritiske komponenten i sammensetningen er lagerbukken. Lagerbukkene som var laget til gjengetenger under 32mm tålte ikke mer enn 20kN i aksial retning. Derfor måtte en 32mm gjengestang velges for å tilpasse mutter og mutterhus. På grunn av dette er mutter og mutterhus overdimensjonert i forhold til kreftene den blir utsatt for i vårt produkt. Da leddene kan bli utsatt til uforutsette krefter, blir overdimensjonering av disse komponentene akseptabelt. Endene av gjengestengene maskineres slik at de passer i lagerbukkene og kulelagrene. Den ene enden er lengre, da gjengestangen må fortsette videre til tannhjulet for å rotere stangen. Figur (19)

For å kunne definere nødvendig moment, kraft og hastighet av kuleskruene gjennomførte vi beregninger. Disse beregningene er vist nedenfor:

For å forklare beregningene defineres følgende parametre:

Parameter	**Forklaring**
Pitch (P)	Den lineære avstanden mellom to spor.
Lead (L):	Den lineære avstanden skruen beveger seg når et spor har tatt en full revolusjon.
Kraft (Fx) i den aksiale retningen:	Kraften kuleskruen må dra med, for å kunne løfte pallen med deler.
Treghetsmoment (J)	Hvor mye moment som trengs for å få en vist vinkelakselerasjon på et objekt.
Moment ved akselerasjon (Mc)	Moment nødvendig for å kunne akselerere til ønsket hastighet
Moment til å dra lasten (Md)	Moment nødvendig for å dra lasten ved konstant hastighet
Maksimum moment (Mt)	Maksimum moment motoren må kunne påføre
Vinkel akselerasjon (w)	Hvor raskt vinkelhastigheten blir forandret
Rotasjonshastighet (N)	Rotasjoner per minutt (rpm)

Tabell 6: Forklaring av parametere brukt til beregninger, [39]

Parameter	**Verdi**
Effektivitet (n)	0.91
Lead (L) [mm]	10
Kraft i den aksiale retningen (Fx) [N]	19400
Tid (t) [s]	10
Treghetsmoment (J) [kg/m^2]	6.40
Rotasjonshastighet (N) [rpm]	260
Total horisontal bevegelse (x) [mm]	432

Tabell 7: Verdier til parametere brukt til beregninger,[39]

Rotasjonshastighet

For å velge motor måtte rotasjonshastigheten beregnes i tillegg til momentet. Dette er viktig fordi systemet må kunne løfte pallen innen tidskravet satt av Tronrud Engineering. Ved å ta i bruk de definerte parametrene kan den nødvendige rotasjonshastigheten beregnes.

$$(\frac{\frac{x}{L}}{t}) \times 60 = rpm \tag{11}$$

$$(\frac{\frac{432mm}{10mm}}{10s}) \times 60 = 260rpm \tag{12}$$

Ut i fra disse beregningene kan en motor som kan gi 100Nm med 260 rotasjon per minute dimensjoneres.

Nødvending moment for å rotere gjengestengene:

$$M_t = M_d + M_a \tag{13}$$

$$M_d = \frac{Fx \times L}{2000 \times \pi \times \eta} \tag{14}$$

$$M_d = \frac{19400N \times 10mm}{2000 \times \pi \times 0.91} = 35Nm \tag{15}$$

$$M_a = J \times \omega \tag{16}$$

$$J = \frac{2\pi \times N}{60 \times t} \tag{17}$$

$$J = \frac{2\pi \times 260rpm}{60 \times 10s} = 1.815rad/s \tag{18}$$

$$M_a = 6.400kg/m^2 \times 1.815rad/s = 11.6Nm \tag{19}$$

$$M_t = M_d + M_a = 35Nm + 11.6Nm = \mathbf{46.616Nm} \tag{20}$$

Konklusjon av momentberegninger

Likning (20) viser at maksimum moment motoren må kunne påføre en gjengestang er lik 46.616Nm. Da vi har to gjengestenger må motoren kunne produsere **93.232Nm**. I virkeligheten vil friksjonskoeffisent øke nødvendig moment, og derfor blir en motor som kan løfte mer enn beregnet verdi valgt. Dette var nødvendig fordi friksjonskoeffisienten til systemet ikke var oppgitt ved leverandøren.

Lagerbukker

Lagerbukkene er brukt til å feste gjengestenger til bakken. Lagerbukkene skal ta imot reaksjonskrefter i horisontale retningen når saksebeinene er løftet opp. Derfor var det viktig å velge noe som tålte mer enn den beregnede horisontale kraften på 19400N (Likning (10)). De mekaniske egenskapene av lagerbukkene er listet nedenfor:

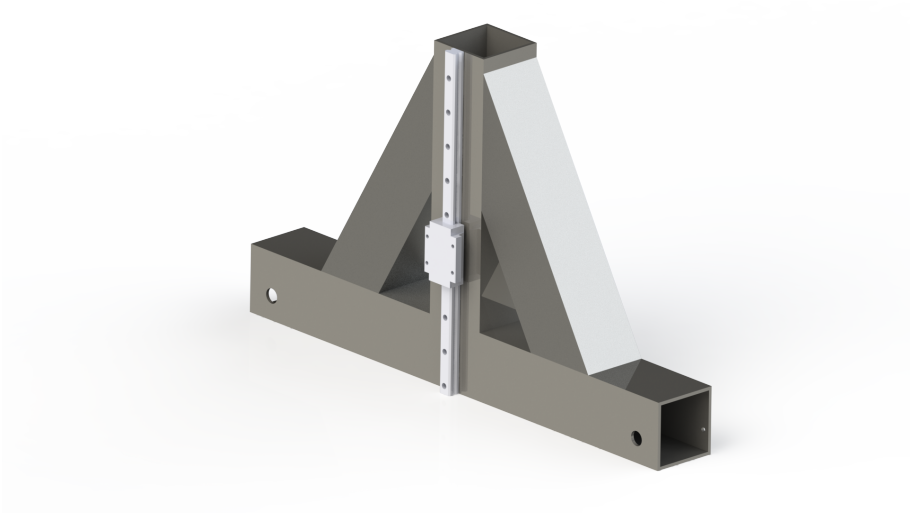
- SEB - F sammensetning
- 32x10
- R1591 120 30
- Dynamisk last: 26,000 N
- Statisk last: 47,000 N

Kulelager

Kulelagrene er festet til lagerbukker og er ansvarlig for å tillate rotasjon i gjengestengene. Informasjon om dette er tilgjengelig på leverandøren sin nettside [39] .

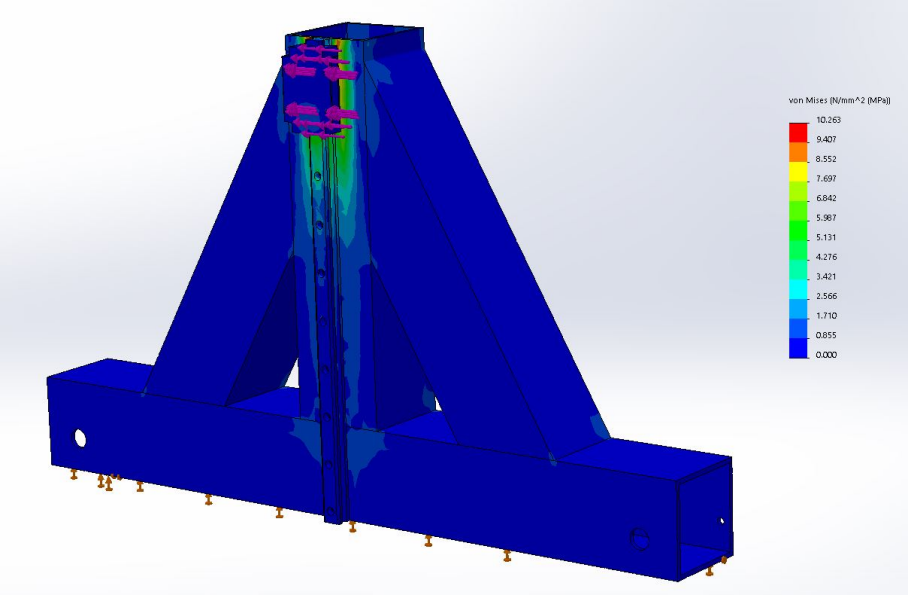
- R1590 120 30

3.3.6 Støttestruktur



Figur 20: A-profil

I et vanlig system som bruker saksebein som løftemekanisme er det støttestenger mellom beinene. I produktet vårt måtte disse støttestengene fjernes på grunn av et krav fra Tronrud Engineering som sier at trallen ikke skal kjøres opp på en plattform eller rampe. På grunn av dette får produktet en svakhet mot momentene fra sidene. Hvis en operatør lener seg på produktet når pallen er løftet opp, har ikke produktet god nok motstandsdyktighet mot disse kreftene og momentene. Strukturen vi har utviklet for å fikse denne svakheten har vi kalt A-profilen. A-profilen er bygget opp av fire firkantrør og en lineær skinneføring som er festet til baksiden av vinkelen (Figur 20). Firkantrørene er sveiset sammen for å lage en struktur som ligner på bokstaven 'A'. Det horisontale firkantrøret på bunnen er hvor kjeder og tannhjul som driver kuleskruene er plassert.



Figur 21: Styrkeberegning av A-profilen

Designprosessen av A-profilen begynte med å kjøre styrkeberegning på hele modellen. Hensikten var å simulere en operatør som lener på produktet fra siden. Simuleringen var kjørt ved å ha en kraft av 2000N som virker på vognen fra siden, når den er ved maksimum høyden. (Figur 21) Momentene er størst ved maksimum høyde på grunn av lengst momentarm. Ved å bruke denne verdien vil strukturen tåle momentene ved alle høydene. Simuleringen viser at strukturen blir utsatt til 10.263 MPa som maksimum spenning. Siden flytegrensen til S355NH er minst 470MPa, kan vi si at A-profilen klarer å støtte produktet ved å ta imot eksterne krefter fra sidene.

Lineær skinneføring

Figur 22: Lineær skinneføring

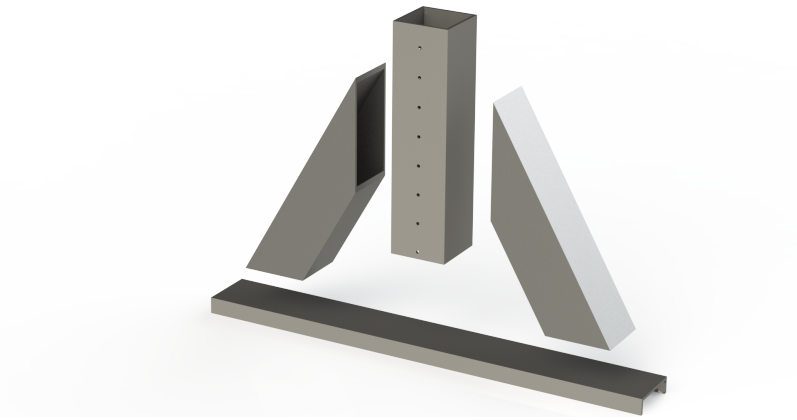
Lineære føringer inneholder bruk av en vogn som er festet til lasten og et spor hvor vogna kan bevege seg lineært. Hensikt av mekanismen er å sikre at bevegelsen av lasten følger en rett linje, enten vertikalt eller horisontalt. I produktet vårt sikrer mekanismen at pallen blir løftet vertikalt. Begge komponentene er hyllevarer som kan kjøpes inn fra Aratron.no.

Fysiske egenskaper av HGW 35HC [40]:

- Maksimum moment: 1.4 kNm
- Vekt: 2.06kg
- Mutter: M8x25mm
- Dynamisk last: 60.21 kNm
- Statisk last: 91.63 kNm

HGW 35HC ble valgt som vogn av hensyn til to grunner. Systemet kan motvirke opp til 1400Nm som moment fra sidene. I tillegg til dette er overflatearealet av vognen større enn de andre alternativene tilgjengelige gjennom Aratron. Dette betyr at kraftoverføringen fra lasten til den lineære føringen er fordelt over et større areal. Dette vil føre til mindre spenninger på grunn av bøying. Systemet tåler en statisk last av 90kN, som betyr at hvis det oppstår en feil i saksebeina, kan den lineære føringen støtte vekten til lasten ved å feste bremser til vognen.

Firkantprofil-bjelker

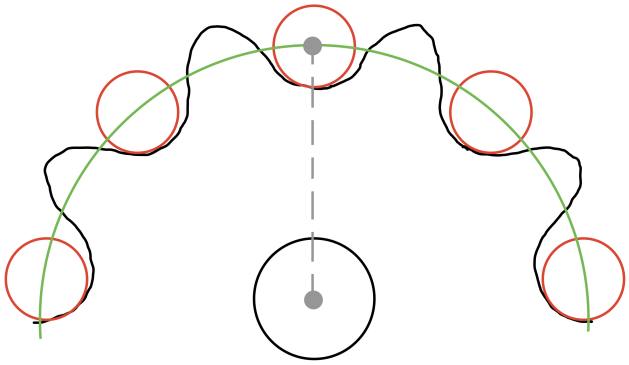


Figur 23: A-profil exploded view

Hovedstrukturen til A-profilen består av 4 hulprofil stenger laget av (S355NH) stål. (Figur 23) Disse stengene skal sveises sammen til å lage støttestrukturen. Norskstål leverer en 12m stang som standard, men kun 3.2m er nødvendig for støttestrukturen . Detaljer om maskinering er i 2D tegnigner i vedlegget.

3.3.7 Kjeder og kjedehjul

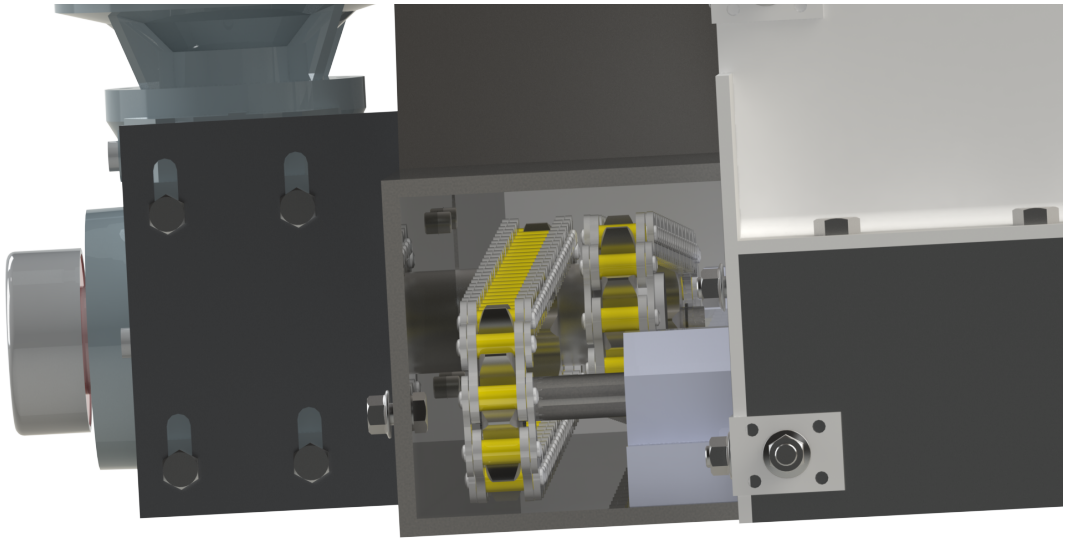
Etter at vi hadde valgt konsept fokuserte vi på å undersøke hvilke kjeder som var best egnet for vårt prosjekt. Vi valgte å gå for den europeiske standarden 16-b1-kjede. Dette valget ble basert på utregninger av når kjedet ble belastet som mest. Tidspunktet hvor systemet belastes mest er der hvor motoren akselererer fra startposisjon til topphastighet. Dette betyr at belastningen på kjedet vil være størst i dette tidspunktet. Hvis vi skal ta med dette i utregningen må vi først finne hastigheten vi ønsker å ha når systemet skal løfte delene fra bunnposisjon til topposisjon på 10 sekunder. Fra Solidworks har vi hentet ut hvor langt kuleskruene må gå i horisontal retning for at delene skal nå topposisjon. Utifra dette kan vi regne ut hvor mye motorkraft vi trenger for å rotere kuleskruene (Likning 20). Denne motorkraften vi kom frem til gjør at vi valgte motoren dere kan lese om i neste avsnitt (3.3.8)som leverer 153Nm moment. Dette kan vi bruke til å regne ut hvor mye kraft kjedet må tåle under maksimal last av motoren, ved å dele den på den effektive avstanden mellom kjedehjulet og der kjedet ligger i kjedehjulet.



Figur 24: Illustrasjon av effektiv avstand mellom kjedehjul og kjede

$$F = \frac{\tau}{\mathrm{d}} = \frac{153Nm}{0,037135} = 4120,1N \tag{21}$$

Kjedet som heter 16-b1 har en gjennomsnittlig bruddstyrke på ca 60kN, ifølge IWIS [28]. Bruddstyrken er noe vi vil holde oss langt unna, så selv om vi har multiplisert kreftene som påvirker kjedet med 2 som sikkerhetsfaktor tidligere, ender vi opp med en styrke som er veldig langt over hva som er nødvendig i forhold til beregningene over (likning 21). Dette vil sørge for at vi kan garantere at kjedet ikke vil ryke om systemet blir brukt på riktig måte av lagerarbeideren. I og med at vi valgte 16-b1 som standard er vi også nødt til å bruke tilhørende kjedehjul. Dette er grunnen til at vi valgte 16-b1 som kjede. Kjedehjulene som passer til 16-b1 er de minste kjedehjulene vi fant som passet på akslingene vi ønsker å bruke (motoraksling og aksling til kuleskruene). Siden vi ikke har rukket å lage et sikkerhetssystem som låser hele mekanismen uten motorkraft ser vi på det å overdimensjonere kjedene som en nødvendighet. Kjedehjulene har 9 tenner hver så det vil ikke foregå giring på kjedehjulene.



Figur 25: Bilde av kjedet i sammenstillingen (på innsiden av A-profilen)

Figur 25 viser at det går et kjede til hver side av pallen. Hvert kjede blir ført til hver side av pallen og drar hver sin kuleskrue i samme retning. Dette blir gjort ved at to kjedehjul er koblet til akslingen til motoren. På venstre side av figuren kan man se at motoren er montert til en egenprodusert del. Denne delen har avlange monteringshull som gir mulighet for å etterstramme kjedene.

3.3.8 Motor

For å finne nødvendig kraft for å løfte systemet vårt må vi se på systemet som en helhet. Dette er for å få en oversikt over hvor mye motorkraft vi trenger for å løfte delene på pallen. I tillegg til dette må vi ta høyde for vekten av delene til systemet som er med i løftebevegelsen og derfor belaster motoren. Vinklene som holder pallen er med på løftebevegelsen. Det samme er to hulprofiler som er boltet sammen på hver side av pallen. Andre parametre som kjeder, kuleskruer og saksebein vil også skape friksjon i systemet, men i belastningen av motoren regner vi dette som neglisjertbart fordi de ikke vil gjøre store utslag på resultatet i beregningene. Utifra likning 10 vet vi hvor mye kraft som kuleskruene blir belastet. Dette er tallet som motoren er nødt til å klare å dra til seg. Deretter kan vi finne ut arbeidet som motoren må yte for å forflytte kuleskruene den strekningen som er nødvendig:

$$W = F * L = 19400N * 0,424m = 8225,6J \tag{22}$$

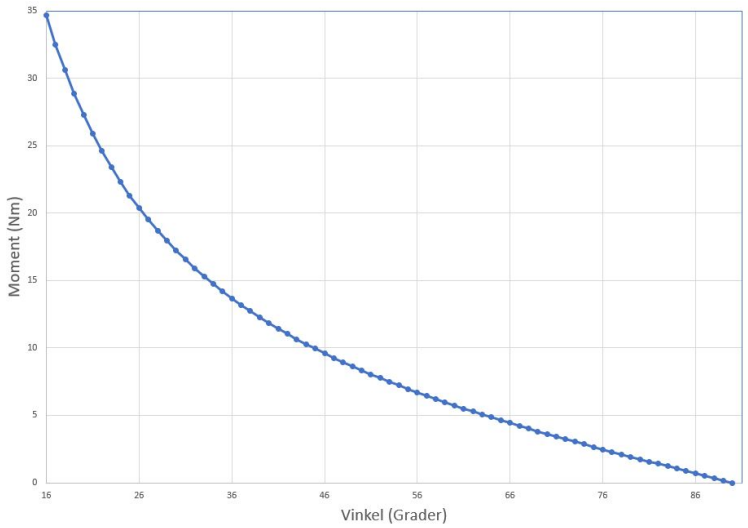
Effekten vi da trenger for å løfte pallen opp på 10 sekunder:

$$P = \frac{W}{t} = \frac{8225,6J}{10s} = 822,6W \tag{23}$$

Dette er altså ytelsen per kuleskrue, som vil si at vi trenger en motor med det dobbelte av dette som ytelse for å kunne dra til oss kuleskruene på både venstre og høyre side av pallen.

$$P = 822,6W * 2 = 1645W \tag{24}$$

En ting å bemerke seg er at kraften på 19400N ikke er konstant, men faller iløpet av bevegelsen. Vi har derfor laget en graf over hvor mye moment som er nødvendig i løpet av hele løftebevegelsen. Dersom momentet som motoren leverer følger denne grafen (26) oppnår vi konstant hastighet på løftebevegelsen.



Figur 26: Foranding av moment for å kunne rotere en gjengestang ved varierende vinkel av saksebein

Etter å ha beregnet hvor mye moment (100Nm) og rotasjonshastighet (260 rpm) som er nødvendig for å løfte pallen, begynte vi med å lese om step-motorer etter anbefaling fra vår intern veileder.

Steppermotor

Hovedfordelen med steppermotorer er at bevegelsesavstanden kan beregnes uten å bruke avstandssensorer. Antall rotasjoner i motoren kan programmeres og motoren kan derfor stoppes automatisk ved et satt antall steg. Dette kan hjelpe til med å unngå skader på operatør og maskinen. Dette gjør også at vi kan unngå bruk av eksterne avstandsensorer. Siden løftemekanismen skal styres manuelt må motoren overstyre brukerstyringen når pallen treffer bakken og når den kommer til maksimalt satt høyde.

Konklusjon

Etter vurderingen av alternativene og etter anbefaling fra Tronrud Engineering, ble en asynkron induksjonsmotor valgt til produktet. SEW-Eurodrive har blitt valgt som leverandør. Ved hjelp av nettsiden ble følgende spesifikasjoner valgt ut til motoren:

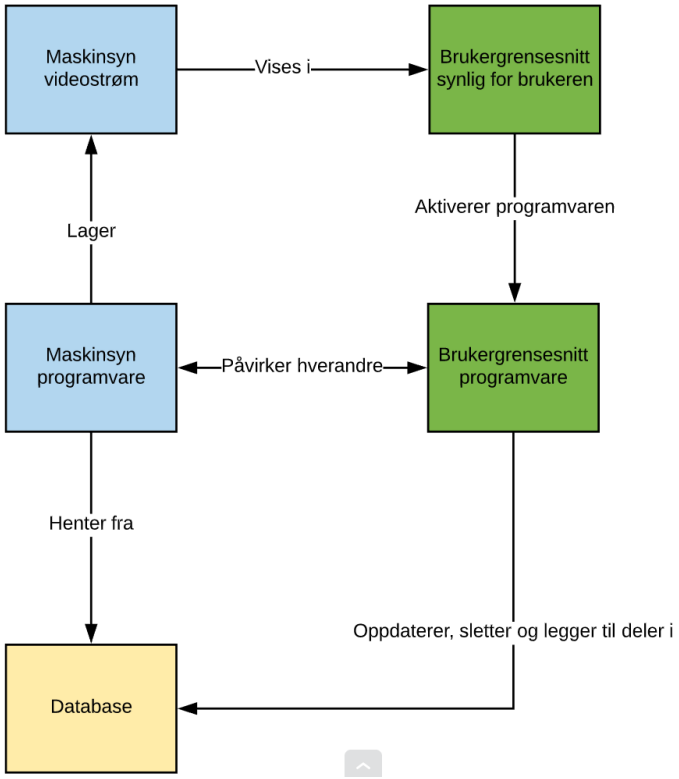
Spesifikasjon	**Verdi**
Moment (Nm)	153
Diameter på aksling (mm)	35
Motor spenning (V)	230/400
Strøm (A)	5,9/3,4
Vekt (kg)	39,00

Tabell 8: Tekniske egenskaper av valgt motor

4 Teknisk - Data

4.1 Modellering

Vår oppgave krever samspill mellom flere ulike programvarer, med mål om å utvikle en programvare som fungerer raskt og effektivt. Oversikt over programvaren vi utvikler til systemet er en viktig del av prosessen om vi ønsker å oppnå slik programvare. Modellering gir oss et startpunkt hvor vi kan begynne å sette oss inn i hvordan programvaren skal være før programmereringen starter. Siden utviklingsmetoden vår for prosjektet er evolusjonær så er det å navigere enkelt i gamle iterasjoner viktig for å utvikle og optimalisere systemet. Vi har modellert diagrammene basert på UML. Vi har lagt til eller fjernet detaljer etter behov. Det er nødvendig for oss og representere programvaren på en slik måte at det er mulig å identifisere hvilke funksjonaliteter vi trenger. Disse funksjonalitetene skal være sporbare slik at oppdatering av disse funksjonalitetene i programvaren er lett å finne. Når vi har produsert programvarearkitekturen får vi en bedre oversikt over programvaren i systemet, hvor brukersituasjon og sekvensdiagrammene gir oss ett mer detaljert perspektiv på funksjonene.



Figur 27: Programvarearkitektur

Figur 27 er programvarearkitekturen til systemet. Det øverste laget representerer det som er synlig for brukeren, mellomlaget representerer programvaren og databasen til systemet finner vi i bunn av programvarearkitekturen.

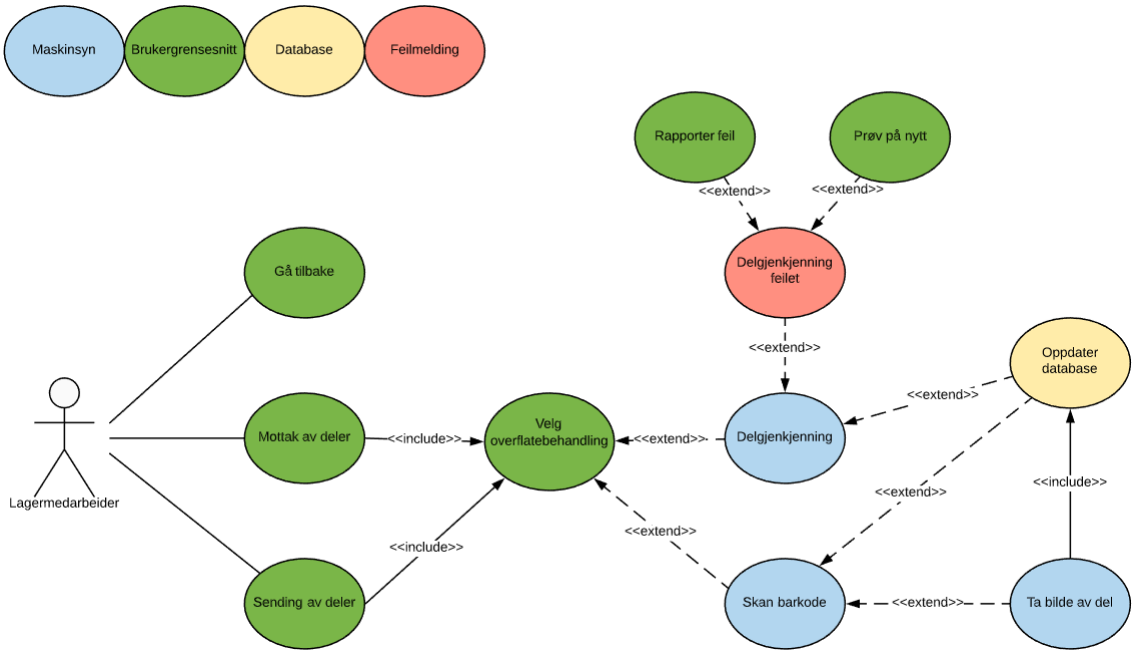
Diagrammet vist i figur 27 har visse detaljer som skiller seg fra en tradisjonell programvarearkitektur. Database komponenten i diagrammet er generisk. Istedenfor å dele opp datasett i egne kategorier har programvaren bare en datakomponent, database. Ved å gjøre dette minsker vi uorden i diagrammet ved å redusere antall komponenter. Et annet avvik fra en typisk programvarearkitektur er linjene. Linjene i diagrammet vårt har både retning ved bruk av piler og tekst. Det vi ønsker å formidle er flyten av informasjon i systemet og samhandlingen mellom komponentene systemet består av. Hensikten med disse endringene er å formidle informasjon på en slik måte at personer uten forkunnskap til UML kan forstå.

Metoden som er brukt for å oppnå oversikt over programvaren i systemet gjennom UML:

- 1. Lag brukersituasjon diagram
- 2. Lag sekvensdiagram ut fra funksjonaliteter identifisert i brukersituasjon diagrammet
- 4. Programvarearkitektur

Roten i UML er aktørene i et brukersituasjons-diagram. Vi begynte med å identifisere vår aktør, som er en lagermedarbeider. Denne aktøren påvirker systemet gjennom brukergrensesnittet. Funksjonalitetene av brukersituasjoner som implementeres i systemet finner man ved å sette seg i aktørens sko, i figur 35 har vi laget et brukersituasjons diagram for aktøren lagermedarbeider.

Etter gjennomgang av flere iterasjoner av brukersituasjons-diagrammer og sekvensdiagrammer identifiserte vi hvilke forskjellige funksjoner og programmer som var nødvendig for å oppnå et system som kunne løse problemstillingen. Dette kan vi derfor visualisere gjennom programvarearkitekturen vist i figur 27.

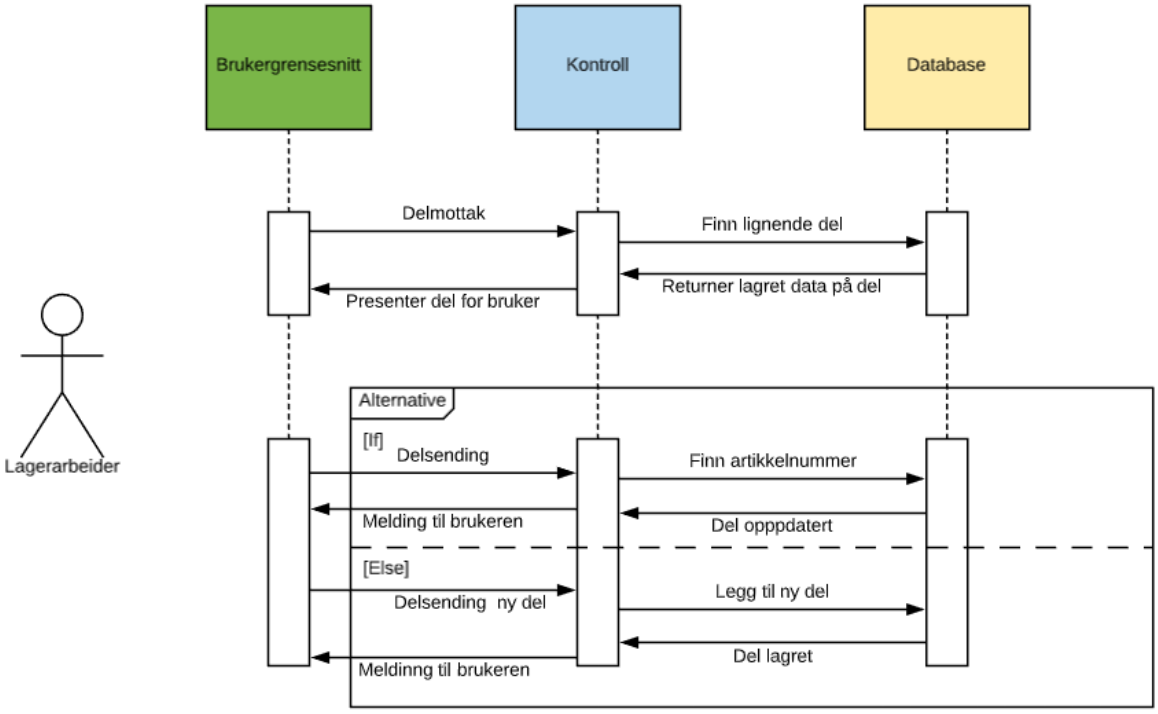


Figur 28: Brukersituasjon diagram: Brukergrensesnitt

I figur 35 vises brukergrensesnittet visuelt representert som et brukersituasjon-diagram. Vi har fargekodet de forskjellige brukersituasjonene. De grønne brukersituasjonene skal vise hvilke knapper lagermedarbeideren kan trykke på og dermed påvirke programmet. Ved å lese diagrammet fra venstre til høyre kan man se at lagermedarbeideren har tre brukersituasjoner å velge mellom.

Vi ønsker å forklare med et eksempel. La oss si at lagermedarbeideren har mottatt en pall med deler hvor de har blitt overflatebehandlet. Lagermedarbeideren vil da sette igang brukersituasjonen ”mottak av deler”. Festet til denne brukersituasjonen er en pil hvor det står ”include” på. I UML hvor en brukersituasjon har en pil med ”include” vil den brukersituasjonen med pilen festet til boblen settes igang. Lagermedarbeideren vil nå velge hva slags overflatebehandling delene har vært gjennom. Det er tilsynelatende irrelevant om man velger ”mottak av deler” eller ”sending av deler” siden begge inkluderer samme brukersituasjon, men ”velg overflatebehandling” har to andre piler festet til brukersituasjonen hvor linjene er stiplet og det står ”extend” istedenfor ”include”. ”Extend” betyr at brukersituasjonens flyt har blitt påvirket av en hendelse. Hendelsen i dette eksemplet er lagermedarbeiderens tidligere valg av brukersituasjon ”mottak av deler”, som forteller oss at vi har en del som skal gjenkjennes og derfor blir brukersituasjonen ”delgjenkjenning” satt igang. ”Extend” sine piler blir plassert på motsatt måte som ”include” og kan tolkes som ”include” sin motpart.

Videre ut fra en eller flere brukersituasjoner, tar man disse og utvikler et sekvensdiagram. Her undersøkes dynamikken mellom programvaren, brukergrensesnittet og databasen. I figur 29 ser vi nærmere på brukersituasjonene "mottak av deler" og "sending av deler, som er funksjoner identifisert fra brukersituasjonsdiagrammet vist i figur 35.



Figur 29: Sekvensdiagram

4.2 Database

Systemet vårt trenger en måte å lagre data og informasjon om dataen slik at systemet enkelt kan finne og bruke denne dataen. Vi valgte å bruke SQL for å organisere databasen. Dette er på bakgrunn av at vi er kjent med dette fra før av og på grunn av mulighetene for integrasjon sammen med Qt.

4.2.1 Struktur

Strukturen til databasen er laget ved hjelp av phpmyadmin for å modellere relasjonen mellom tabellene i databasen. Tabellene som er mest sentrale i databasen er Liste-tabellen og objekt-tabbelen. Liste-tabellen har oversikt over objekter som blir prosessert av gjenkjenningsprogrammet, slik at vi kan lage en liste av alle objektene som har blitt skannet i løpet av økten. Objekt-tabellen har informasjon om objektene som blir behandlet av gjenkjenningsprogrammet, slik som artikkelnummer og bildet av objektet. Bildet av objektet er viktig å lagre fordi det må brukes av gjenkjenningsprogrammet for å sammenligne.

4.2.2 Bruk

Databasen blir brukt i flere deler av programmet. Den blir brukt sentralt for å organisere dataen som blir brukt av andre deler av programmet. For å legge til ting i databasen bruker vi Insert SQL kommandoer, som enten blir gitt av brukergrensesnittet eller gjenkjenningsprogrammet. Brukergrensesnittet har kommandoer som har ansvar for å legge til informasjon om listene som blir generert under en av gjenkjenningsøktene. Denne informasjonen omhandler leverandører og hvilke type behandling delen skal få. Informasjonen databasen gir ut til brukergrensesnittet er informasjon som foreksempel hvilke deler som ble behandlet på en bestemt dag. Dette minimerer antall deler som sammenlignes i gjenkjenningsprogrammet. Listen av objekter som har blitt skannet blir generert i databasen som en tabell som skal kunnes skrives ut av systemet rett inn i et Excel-ark, slik at dataen er lett tilgjengelig. Denne funksjonen for å eksportere data blir ikke gjort i selve brukergrensesnittet, men må blir gjort via selve databasebehandler, som i vårt tilfelle er phpmyadmin.

4.2.3 Integrasjon i brukergrensesnitt

Databasen samhandler med brukergrensesnittet på flere måter. Den gir ut informasjon som blir forespurt av brukergrensesnittet, samtidig som den setter inn informasjon som blir gitt av brukergrensesnittet. Når GUI-en ber om informasjon for å sjekke hvilke deler som ble skannet på en spesifikk dato, eller for å hente ut bildet til en del, bruker databasen et SQL-kall basert på hvilken type informasjon som er ønsket. SQL-kallet blir generert ved hjelp av funksjoner som bruker Qt-klasser som er laget for å integrere SQL med Qt.

4.2.4 Integrasjon i gjenkjenninsprogrammet

For at gjenkjenningsprogrammet skal kunne begrense bildene som er nødvendig for å gjenkjenne bilder, trenger databasen en måte å lagre bildene på slik at det skal passe med gjenkjenningsprogrammet. Bildene blir lagret i en tabell som kan brukes til å eksportere bildene til en mappe i programmet vårt som skal brukes av gjenkjenningsprogrammet vårt til å sammenligne bilder. Tabellen skal også ha mulighet til å importere bilder fra gjenkjenningsprogrammet for å lagre et nytt objekt med bilde inn i objekt-tabellen.

4.3 Brukergrensesnitt

Til dette systemet er det viktig med brukergrensesnitt slik at brukeren av systemet kan velge hvilke deler som skal bli skannet. Det er viktig at brukergrensesnittet er brukervennlig. Brukeren skal kunne ha muligheten til å dirigere seg fram og tilbake i brukergrensesnittet uten problemer. Et av kravene Tronrud stilte var at brukergrensesnittet skulle ha Windows design, noe vi har fulgt siden starten av prosjektet. Brukergrensesnittet skal være på norsk og være koblet til en database hvor informasjon blir registrert. Det skal også være et innloggingssystem for å ivareta sikkerhetsmessige hensyn. Etter innlogging skal brukeren få velge om pallen skal mottas eller sendes, og deretter velge overflatebehandligstype og tilhørende bedrift.

4.4 Gjenkjenning

4.4.1 Valg av gjenkjenning

Det finnes ulike måter å oppnå gjenkjenning av objekter ved hjelp av maskinsyn. De fleste metodene går ut på å behandle bildet slik at datamaskinen kan tolke og kategorisere bildene. De mest relevante algoritmene for å løse problemstillingen vår bruker de samme prinsippene når de behandler bilder. Faktorer vi tok i betraktning for å velge best egnet algoritme var brukervennlighet og dokumentasjon av funksjoner i algoritmen.

Systemet vårt måtte klare å gjenkjenne deler ved å sammenligne bildet av delen opp mot en database av referansebilder. Vi trengte derfor en metode som var fleksibel nok for å kunne bruke resultatet av bildebehandlingen til å identifisere objektene som blir tatt bilde av.

De fleste gjenkjenningsalgoritmer kommer ferdig bygget sammen med programvarebiblotek som f.eks OpenCV og TensorFlow [21]. Disse programvarebiblotekene kommer med verktøy som spesialiserer seg på forskjellige områder innen maskinlæring og maskinsyn. Vi har valgt å jobbe med OpenCV, da vi har mer kjennskap til OpenCV enn til TensorFlow.

4.4.2 Kriterier for gjenkjenning

Utvalgt algoritme for å gjenkjenne deler måtte inneha egenskaper som kunne oppnå ønsket hastighet og presisjon fra oppdragsgiver. Algoritmen måtte være robust nok til å ha minimalt med brukerassisterte-handlinger under normal funksjon av systemet. Tanken var at bruker kun skulle behøve å bekrefte at delen som ble avbildet faktisk var den riktige delen systemet vårt gjenkjente delen som.

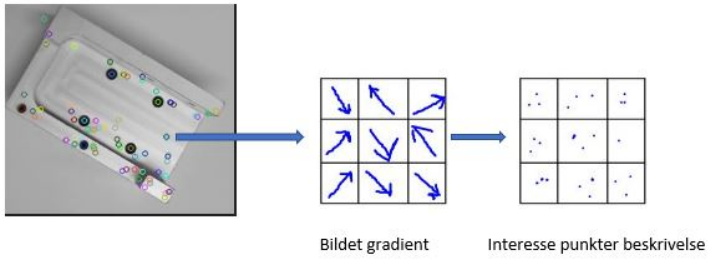
Oppdragsgiver ønsket en relativt rask gjenkjenning av delene på mellom 2-3 sekunder. Bildebehandlingen og gjenkjenningen måtte derfor foregå raskt, så vi måtte finne en måte å redusere ytelsestiden på. Vi baserte valg av bildebehandlingsmetode på en løsning som ville kunne prosessere resultatet fra bildebehandlingen slik at det kunne implementeres i en database med resten av informasjonen om hvert objekt som f.eks. artikkelnummer og navn. Forskjellige algoritmer genererer forskjellige resultater. Selv om ytelsen til f.eks SURF-algoritmen kan være bedre enn SIFT, kan det være mulig at SIFT-algoritmen ville være bedre egnet for systemet vårt pga. at resultatene fra SIFT lettere kan bearbeides inn i en database og bli sammenlignet med andre resultater.

4.4.3 Hvordan metodene fungerer teknisk

Surf-algoritmen har et filter som bruker Gaussian Smoothing for å oppdage firkanter på bildet, som senere kan bli analysert ved hjelp av en Gruppe-oppdager[22]. Dette er en metode for å finne konsentrerte områder i et bilde med punkter av firkanter som kan brukes til å gjenkjenne deler av bildet. Denne gruppe-oppdageren er basert på Hessematrisen for å lage en kvadratisk matrise som er 9x9.

Etter at områder på bildet er funnet trenger algoritmen en måte å sammenligne områdene med referanseområder på bildene som skal bli gjenkjent. Av og til er ikke referansebilde skalert likt som det bilder som skal gjenkjennes. Dette fører til at algoritmen trenger en måte å gjenkjenne uavhengig av hvordan områdene på referansebildene er orientert og skalert. SURF-algoritmen bruker et skaleringsområde for å finne riktige områder basert på referansebildet.

Etter at områder har blitt funnet på et bilde lager SURF en identifikator av området for å beskrive egenskapene, slik som intensiteten av piksler i området [23]. Etter at algoritmen har funnet identifikatorer fra bildet kan man sammenligne det direkte med andre bilder for å se om de samme områdene dukker opp på begge bildene. Dette gjør at vi kan bruke SURF til å sammenligne bildet av delen som systemet skal gjenkjenne opp mot en database med bilder. Dette resulterer i problem med tanke på tidsbruk for sammenligning av mange bilder. Vi trengte derfor en måte å redusere antall mulige kandidater som skulle sammenlignes.



Figur 30: identifikatorer som blir funnet i et bilde

Effektiv gjenkjenning av objekter kan foregå på ulike måter. Metodene har ulik effektivitet og implementasjonen kan bli vanskeligere basert på hvor komplisert løsningen er. Alle løsningene vi har vurdert for objektgjenkjenning har involvert en gjenkjenningsalgoritme for å finne identifikatorer som vi kan sammenligne for å finne de riktige objektene. Den neste utfordringen vi måtte løse var måten vi organiserte programmet etter behandling av bildet med en algoritme.

For å velge metode som drastisk kan redusere den totale tiden for gjenkjenning blir oppgaven å redusere antall kandidater av riktige alternativer. Brukeren av systemet kan skrive inn informasjon om hvilken behandling delen kommer fra og når den ble sendt til behandling.

En annen metode vi har vurdert er å lage en Bag of visual words. Dette er en måte å samle identifikatorer fra bilder ved hjelp av k-means gruppering for å bygge en ordbok av mønstrene av grupperingene. Mønstrene vil bli lagret i ordboken og histogram vil bli laget ut ifra hvor ofte forskjellige grupperinger oppstår på et bilde. Disse histogrammene kan bli sammenlignet med andre histogrammer fra en database. Histogrammene inneholder mindre informasjon enn det en gjenkjenningsalgoritme ville ha generert, men denne informasjonen kan lettere bli lagret i en database.

4.5 OpenCV

OpenCV[35] er et gratis bibliotek for maskinsyn og er under Berkeley Software Distribution (BSD) lisens. Dette biblioteket tilbyr over flere tusen algoritmer som kan brukes for objektdeteksjon, ansiktsgjenkjeninng eller andre formål. OpenCV støtter de fleste operativsystemer. Vi har valgt å bruke dette biblioteket først og fremst da det er gratis, slik oppdragsgiveren vår ønsket. Den andre grunnen er at OpenCV har god dokumentasjon på hvordan biblioteket fungerer.

4.5.1 Begrunnelse for bruk av SURF

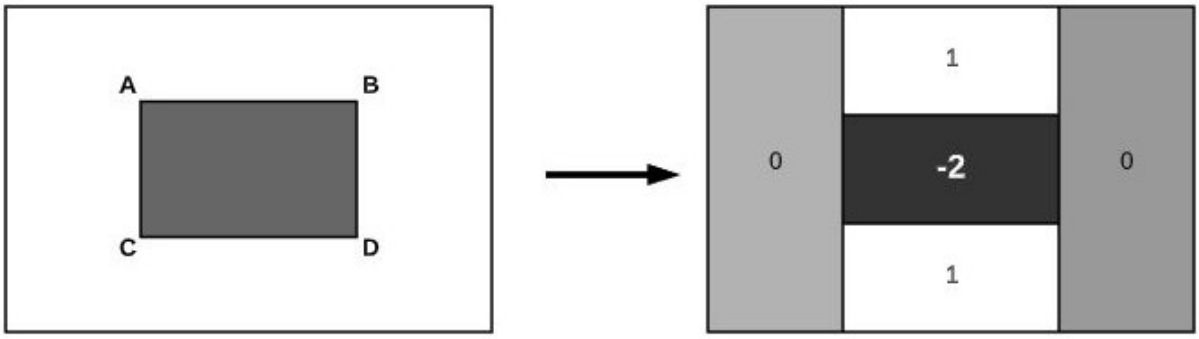
For å kunne velge riktig løsningsmetode måtte vi ha en god forståelse av styrkene og svakhetene til hver av gjenkjenningsalgoritmene. Vi har hovedsaklig arbeidet med SIFT og SURF for å undersøke hvilken som egner seg best til prosjektet vårt. Mange av prinsippene til SIFT er bygd på grunnleggende prinsipper i SURF. Vi har valgt å bruke SURF. Dette valget er basert på at programmet vårt har en fungerende SURF-funksjon, som sammenligner et bilde fra et kamera opp mot en database med flere bilder. Deretter returneres resultatet som har størst sannsynlighet for å være den samme delen.

SURF[36] algoritmen er en lokal funksjonsdetektor som brukes i ulike områder, som for eksempel i klassifisering av bilder. Denne algoritmen er en raskere versjon av SIFT- algoritmen. SURF bruker boksfilter, som er et kvadratformet filter for tilnærming av detekting områder på et objekt. Boksfilterne beregnes ved hjelp av bildeintegraler. Bildeintegral er pikselverdiene i det gitte bildet. Matematikkformelen for bildeintegral/arealsummen av tabell er følgende:

$$I(x,y) = \sum_{\substack{k \leq y \\ j \leq x}} i(j,k) \tag{25}$$

X og Y er punkter på bildet. I(j, k) er verdien av x og y. Videre beregner vi bildeintegralene ved hjelp av boksfilterne. Et boksfilter har 4 hjørner og vi starter alltid fra øverste venstre hjørne. Det blir foretatt en evaluerering av de forskjellige intensitetsverdiene i boksfiltrene og tegnet en beskrivelse ut ifra disse verdiene. La oss ta for oss bildeintegralen som er vist under. Vi beregner summen av intensitet slik: I = summen av intensitet og A, B, C og D er punkter.

$$I = D - B - C + A \tag{26}$$



Figur 31: Venstre viser bildeintegral og høyre side viser boks filter n x n

Hessian-matrise er en feature (egenskap) deteksjon som brukes mye i datamaskinsyn. SURF-algoritmen bruker denne matrisen grunnet god nøyaktighetsytelse. Denne matrisen er nesten identisk med Harris-matrisen. Begge matrisene er brukt for å detekte interessepunkter i objekter. Algoritmene bruker flere skalaer for å detekte interessepunkter i objektet. Hessian velger interessepunkter basert på Hessian- matrisen i det punktet. Under kan vi se Hessian- matrisen hvor $L_{xx}(\alpha)$ er andre ordens partielle deriverte og L_{xy} er en blanding av andre ordens partielle deriverte i x og y. Vi bruker Gaussian- matrisen for å uskarpe bildet.

$$Hessian(\alpha) = \begin{Bmatrix} L_{xx}(\alpha)L_{xy}(\alpha) \\ L_{yx}(\alpha)L_{yy}(\alpha) \end{Bmatrix} \tag{27}$$

I SURF-algoritmen bruker vi representasjon i skala av en bildepyramide. Denne skalaen øker fra en størrelse på 9x9 opptil 256x256. Bildepyramiden glattes ved hjelp av Gaussian-filter eller Gaussian-matrise som vi bruker i programmet. SURF- algoritmen reduserer ikke bildestørrelsen iterativt. Beskrivelsen av interessepunkter på bildet i SURF-algoritmen skjer i følgende steg; Først lager vi en kopi av informasjonen basert på interessepunkter i bildet. Deretter tegner vi en firkant på dette punktet, så tegner vi SURF- beskrivelen ut av det. Haar wavelet er en funksjon som brukes i SURF- algoritmen for å beregne x og y-retningen i interessepunktet i bildet med radius 6s, hvor s er størrelsen. Svaret på Wavelet kan beregnes raskere ved hjelp av bildeintegraler, som SURF bruker. For å få de primære orienteringssvarene i et punkt på bildet summerer vi opp alle svarene med en vinkel på rundt 60 grader. Interessepunkter deles inn i flere underregioner når de oppdages av programmet. Programmet tegner en vektor ut av x og y- svarene. De absolutte verdiene tas med i beregning. Dette gir SURF feature 64 dimensjoner. Dersom dimensjonene er lave går beregningen raskere og gir dårligere match. For å se om match er riktig eller feil bruker vi dobbeldimensjoner. Det vil si 128 dimensjoner. I dette programmet bruker vi også Laplace-matrise for å merke om interessepunkter matcher eller ikke, siden denne matrisen er brukt for å skille et objekt fra sin bakgrunn.

4.5.2 Hvilke funksjoner vi bruker fra OpenCV

I gjenkjenningsprogrammet bruker vi følgende funksjoner fra OpenCV[37]: **imread()** for å se bilder fra en fil og returnere bildet i filen. **imwrite()** funksjonen brukers for å lagre bilder, mens **imshow()** funksjonen brukes for å vise video og bilder på skjermen. **waitKey()** funksjonen venter på en handling når brukeren er ferdig med programmet. Når brukeren klikker på en knapp så vil programmet bli avsluttet. **DetectAndCompute()** funksjonen brukes for å detekte interessepunkter på bildet og beregne dem. **knnMatch()** funksjonen helper oss med å finne de beste matchene for hver beskrivelse i nærheten av interessepunkter. **DrawMatches()** funksjonen tegner matchene som bildet har fått og **GaussianBlur()** brukes for å gjøre bildene mer uskarpe når de blir tatt.

4.5.3 Klasser fra OpenCV

For at programmet skal fungere bruker vi innebygde klasser fra OpenCV. Følgende klasser er tatt i bruk; **cv::VideoCapture** er en klasse for å fange video fra webkamera og hvilken port som brukes må spesifiseres. **cv::xfeatures2d::SURF** klassen brukes for ekstrakte **SURF** sin feature fra et bilde. **cv::DescriptorMatcher** klassen er for å matche interessepunkter enten med et bilde eller med ett sett av bilder.

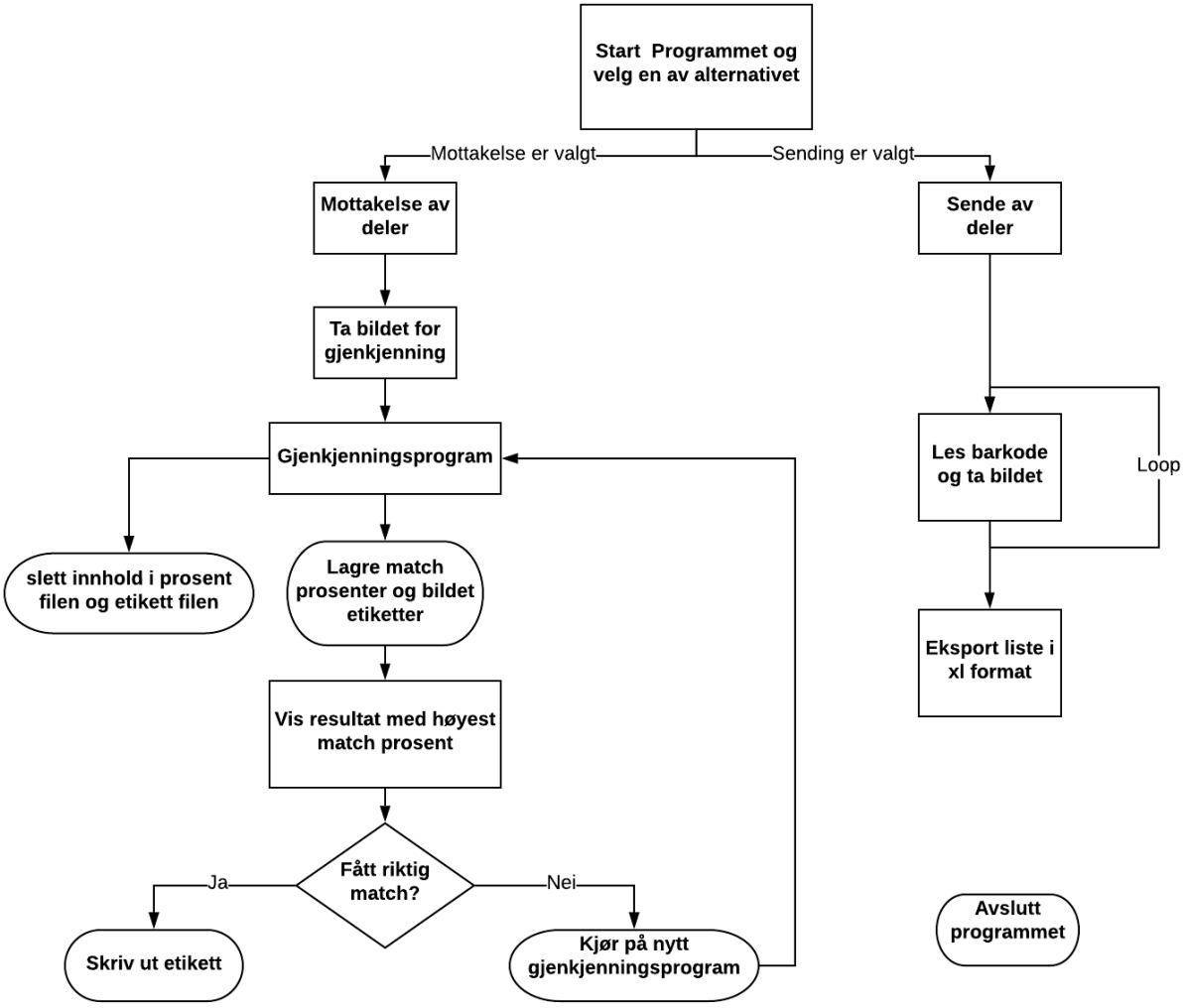
4.5.4 ZBar

ZBar[38] er et gratis biblioteket som vi bruker i programmet for å lese barkoder før vi tar bilder av deler. Dette biblioteket er laget for 32-bit operativsystemer. Den måtte bygges om til 64-bit siden den ikke fungerte med andre programmer som allerede var skrevet for 64-bit.

4.5.5 Funksjonen og programmet

Den første versjonen av gjenkjenningsprogrammet er skrevet i python. Noen av funksjonskravene er for eksempel at programmet skal lese barkode før den tar bilde og lagrer i databasen. Dersom barkoden ikke blir lest av kameraet, så vil den ikke ta bilde og lagre et ugyldig bildeformat. Vi bruker barkoden som label/navn på bildet. Dette gjør det lettere for oss når vi skriver ut etiketter på delene senere. Programmet lagrer ikke duplikater av bilder, så lenge barkoden er unik. Programmet kjører raskere ved sendingsfasen enn i mottakelsefasen. I sendingsfasen leses barkoden og det blir tatt bilde som lagres i databasen. Det eksporteres en liste med xl format. I mottakelsesfasen skjer det flere steps og programmet må sammenligne et bilde med flere bilder for å få match.

4.5.6 Hvordan programmet fungerer



Figur 32: flow chart: Viser hvordan programmet fungerer

Når programmet starter så har vi to valgmuligheter å velge mellom. Disse valgmulighetene består av enten sending til leverandør eller mottakelse av deler fra en leverandør. Hvis "sending" er valgt av lagermedarbeideren så får hun/han noen instruksjoner på skjermen for hvordan vedkommende skal gjennomføre handlingen. Lagermedarbeideren skal legge hver enkelt del under kameraet på en naturlig måte slik at programmet får flest mulig nøkkelpunkter. Dette øker match-prosenten når delen blir skannet senere for gjenkjenning.

Først og fremst skal lagermedarbeideren skanne barkoden som er produsert av TE. Etter dette tar lagerarbeideren bilde av delen og det lagres i en database. Denne prosessen går i en loop og når alle barkode-etikettene og alle delene er skannet så vil lagerarbeideren ha mulighet for å eksportere en liste med informasjon om delene. Etter dette kan brukeren avslutte programmet.

Dersom mottakelse er valgt i programmet, så tar lagermedarbeideren bilde av delen. Deretter starter gjenkjenningsprogrammet å matche bildet med bilder som ble tatt da delene ble sendt til leverandøren. Programmet lagrer alle prosentene og barkodenavnet på hvert enkelt bilde til en fil. Når programmet har kjørt gjennom alle bildene, vil den vise resultatet som har fått høyest match- prosent. Dersom matchen er riktig kan lagermedarbeideren skrive ut en etikett med barkode. Hvis resultatet er feil så kjøres gjenkjenningsprogrammet på nytt, så bildet kan tas ved behov siden det er forskjell på sidene.

4.5.7 Hvordan vil systemet prestere

Systemet har en grei ytelse når det kommer til hvor raskt den leser barkoden. Den gjennomsnittlige tiden det tar for å lese en barkode er på 4 sekunder. Den tiden er selvfølgelig avhengig av brukeren og hvor effektivt han jobber. I gjenkjenningsprosessen så vil systemet bruke noe mer tid. Den tiden øker dersom vi har store mengder med bilder som programmet skal sammenligne bildet med. Vi har tenkt til å redusere denne tiden. Dette skjer enten ved at vi flytter bilder som har fått match til en ny fil eller sletter dem for godt. Dette kan både gjøre at vi øker treffprosenten og effektiviserer programmet på samme tid. Den første versjonen av programmet er en test og gjenkjenner delene med en treffprosent på 60-70 prosent.

4.5.8 Testing av gjenkjenningsprogrammet

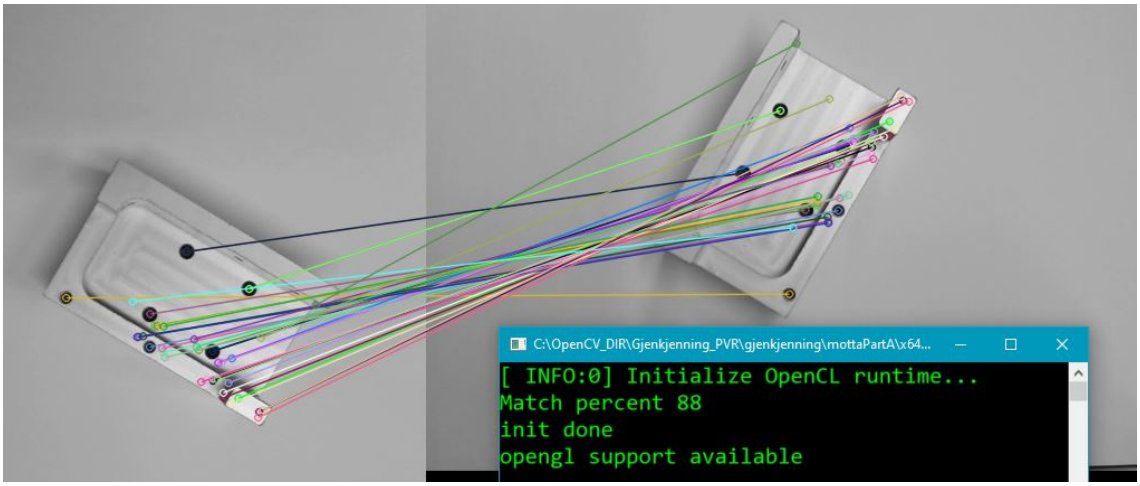
Vi har testet den første versjonen av programmet med deler som vi har fått fra TE, for å se hvor nøyaktig programmet kan gjenkjenne delene. Programmet har fungert bra. Match- prosentene varierer med lysstyrken. Dette er fordi metalldelene er lysreflekterende. Når lyset blir reflektert mot kamera fanger ikke programmet opp alle de viktige parameterene. Da går en del av de viktige punktene på bildet tapt. Dette gjør det vanskelig å få en bra match. Vi har diskutert dette med TE, så vi kommer til å finne en type lys som ikke sender så sterke lysstråler. I tillegg til dette vil vi bruke et svart underlag, slik at den absorberer mest mulig av lysstrålene. Under viser vi to testresultater av programmet hvor programmet gjenkjenner delene og har fått riktig match. Bildene under viser to bilder av samme del, hvor den på høyre side ligger i filen med de andre delene og den andre er tatt nylig for å bli gjenkjent. De små fargede ringene viser nøkkelpunktene på bildet og linja viser at bildet har fått match.

4.5.9 2. versjon av gjenkjenningsprogrammet

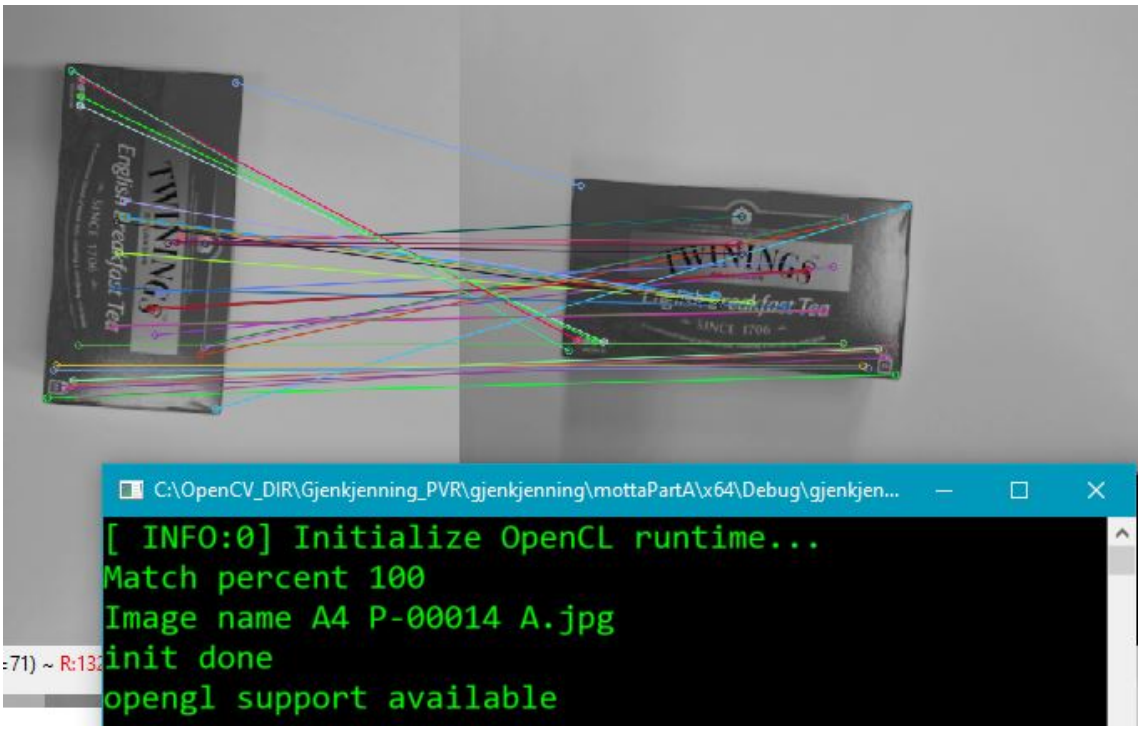
Den andre versjonen av gjenkjenningsprogrammet er skrevet i C++ hvor vi tenkte å bruke samme programmeringspråk både i brukergrensesnitt og i bildebehandling. C++ er et standard programmeringspråk på et høyt nivå og det er et robust språk. I programmet har vi laget 4 klasser. Hver av klassene inneholder både offentlige og private medlemmer som variabler og funksjoner. Alle funksjonene er laget utenfor klassen i en egen cpp-fil og funksjonene har tilgang til alle dataene i klassen.

4.5.10 Testing av versjon 2

Den andre versjonen av programmet er forbedret i match-nøyaktighet og er raskere enn første versjonen. Programmet leser barkoden først og lagrer den i en midlertidig tekstfil. Programmet bruker denne informasjonen som bildenavn når bilder blir tatt. Når programmet skal gjenkjenne delene så vil programmet ta et bilde av delen som skal gjenkjennes og lagrer det i en separat fil. Programmet vil matche det bildet mot andre bilder som er i en annen fil. Programmet viser match-prosenten og den som har fått høyest match-prosent vises på skjermen. I figurene 34 og 33 kan vi se noen test-resulter fra programmet. Vi har testet programmet med ulike gjenstander. Siden vi ikke hadde deler (produsert av Tronrud) med oss fra skolen da denne ble stengt på grunn av koronaviruset. Deler som programmet gjenkjenner har dårlige features og te-pakken som vi ser på figur 34 har bedre features og derfor har fått 100 prosent match.



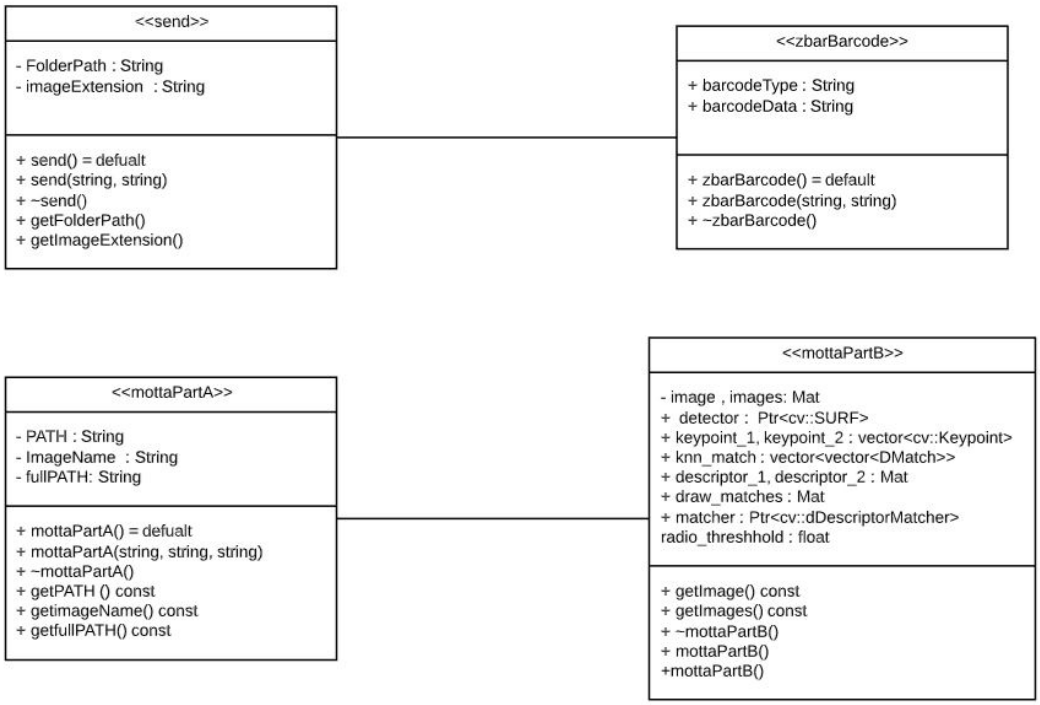
Figur 33: Tester programmet med deler



Figur 34: Tester programmet med te-pakke

4.5.11 Klassediagram

Klassediagrammet under er designet for å vise strukturen av gjenkjenningsprogrammet i C++. Diagrammet viser hvilke funksjoner som er avhengig av andre funksjoner og hvilke variabler av datatyper som brukes i programmet. Noen av variablene inneholder ulik informasjon. For eksempel er "Mat" en klasse i OpenCV som inneholder en matrise hvor matrisehode består av størrelsen på matrisen og hvor matrisen er lagret. Matrisekroppen består av pikselverdiene fra bildet.



Figur 35: klassediagram

4.6 Database

Databasen i systemet vårt blir brukt til å lagre data slik at brukergrensesnittet eller gjenkjenningsprogrammet kan bruke det til å utføre sine oppgaver. Databasen er laget og organisert ved hjelp av MySQL og integrert med brukergrensesnittet ved hjelp av Qt-biblioteker som gjør det mulig å bruke SQL med Qt.

Qt bruker flere SQL-metoder for å hente data fra databasen og for å legge inn ny data. Dataen som skal settes inn i databasen blir generert basert på hva brukeren av systemet gjør i brukergrensesnittet.

Databasen består av flere tabeller som skal holde styr på flere dataelementer. Den skal inneholde liste over objekter som har blitt skannet i løpet av økten og hvilken bruker som har utført skanningen. Dataen blir lagret slik at den kan bli eksportert til en Excel-fil fra SQL-databasen.

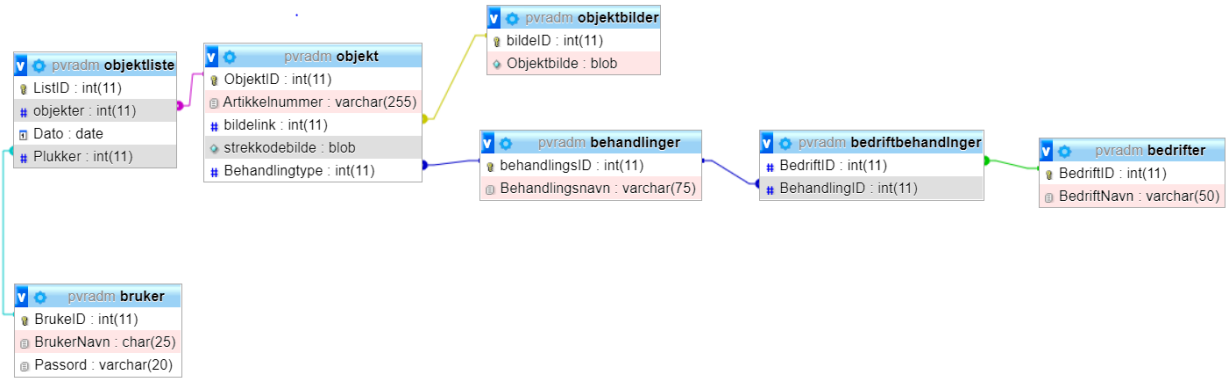
SQL-spørringene som er av interesse er SELECT, INSERT og UPDATE. Disse spørringene vil gjøre det mulig å manipulere dataen slik at gjenkjenningsprogrammet kan redusere bildene den trenger å sammenligne for å gjenkjenne bilder. Dette fører til andre utfordringer som kan redusere effektiviteten til systemet basert på andre variabler, som størrelsen på datasettet og overføringshastigheten.

4.6.1 Struktur

Tabellene i databasen er koblet sammen med forskjellige typer forhold. Hele databasen er koblet sammen på en sånn måte at ingen tabeller står helt uten en relasjon til andre datatabeller. Databasen er bygd opp rundt objektene som skal bli lagret og informasjonen om objektet. Informasjonen om objektet er viktig å lagre slik at dataen er lett tilgjengelig. Dette kan være tidsparende for gjenkjenningsprogrammet og for arbeideren som skal ha muligheten til å manuelt skrive inn artikkelnummer.

Brukertabellen er laget for å ta vare på informasjon om brukeren av systemet. Informasjonen som er lagret i tabellen er brukernavn og passord, slik at det kun er den korrekte brukeren som kan logge inn i systemet. Brukertabellen har en til en relasjon til listetabellen. Dette er fordi en bruker kan ha jobbet med flere lister. Det er viktig å se hvem som har jobbet med en liste, slik at hvis en feil oppstår på grunn av brukerfeil kan man søke etter den ansvarlige personen.

Objekttabellen har en til en relasjon med objektbildetabellen. Dette er fordi vi vil at et objekt bare skal ha et bilde koblet til seg av gangen. Det ble valgt å lagre bildedataen i egen tabell, slik at bildet i tabellen kan



Figur 36: Tabeller i databasen

bli eksportert til en mappe for å bli brukt i gjenkjenningsprogrammet. Den skal også kunne hente bilder fra gjenkjenningsprogrammet og sette informasjonen inn i tabellen. Behandlingtabellen har en mange til mange relasjon med leverandørtabellen. Dette er fordi en behandlingstype kan bli gjort av flere bedrifter og en bedrift kan utføre flere typer overflatebehandling. Denne tabellen har informasjon om hvilke bedrifter som utfører hvilken type overflatebehandlinger. Databasen kan dermed plassere dataen inn i objekttabellen for å registrere hvilken type behandling objektet skal få.

Listetabellen har en til mange relasjon med objekttabellen. Objekter er fremmednøkkelen slik at det kan være flere objekter som tilhører en liste, men et objekt kan bare være i en liste. Tabellen er designet slik fordi objekter skal ut og inn i produksjon og behandling raskt, slik at et spesifikt objekt bare kommer til å bli behandlet en gang. Objektene i listen er ikke unike, dette betyr at det kan være flere av det samme objektet i listen. Dette betyr at vi må ha mulighet til å legge til flere av det samme objektet samtidig for å spare tid for brukeren og for å gjøre listen mer ryddig når den blir skrevet ut.

4.6.2 Integrasjon i brukergrensesnitt

Vi har brukt flere forskjellige type klasser for å integrere databasen med brukergrensesnittet via Qt. Disse klassene er inkludert i Qt, men det er nødvending å definere hvilke QSQL klasser som skal brukes i Qt brukergrensesnitt-klassene. Noen av SQL-klassene trengs for å manipulere data i databasen og noen andre trengs for å hente ut data fra tabellene inne i databasen.

QSqlDatabase

Denne klassen har ansvar for å koble databasen til Qt, og gjør det mulig å hente ut data som vi senere skal bruke ved hjelp av andre klasser. Når vi først kobler til databasen ved å definere hvilken database vi skal bruke. Her oppsto det mange problemer på grunn av at windows 10 versjonen av Qt versjon 5.14 ikke inkluderte QMYSQL-driveren som er nødvendig for å koble til databasen. Dette problemet ble løst ved å laste ned frittstående MySQL istedenfor å kjøre MySQL via XAMPP. Valget av å starte med XAMPP istedenfor bare MySQL var at dette var det programmet vi var mest kjent med, men etter at det viste seg at flere problemer oppsto på grunn av dette valget bestemte vi oss for å jobbe med bare MySQL.

QSqlQuery

For å kjøre SQL spørringer via Qt er denne klassen nødvending. INSERT, DELETE, SELECT og UPDATE er spørringer som er mulig for programmet å bruke når denne klassen har blitt definert i Qt. I programmet vårt bruker vi INSERT til å legge til nye objekter til listen og hva slags behandling objekter skal ha. SELECT blir brukt til å hente ut informasjon om hvilken leverandør som utfører forskjellige typer behandlinger. For å manipulere spørringene slik at vi får ønsket data ut av databasen må vi spesifisere variabler som skal bli brukt i spørringen. For eksempel i brukergrensesnittet når det kommer til behandlinger vil navnet på behandlingen bli registrert og brukt til å registrere det i objekt-tabellen via INSERT spørringen. SELECT spørringer blir viktig når programmet skal sjekke om brukernavnet og passordet som er skrevet inn er gyldig. Dette er relativt lett fordi QSqlQuery klassen inkluderer en metode for å sjekke om SELECT spørringen som er utført gir resultater. Denne metoden gir et binært resultat som er enten True eller False. Dette gjør det lett å definere bn SELECT spørring som bare gir resultater hvis både Brukernavn og passord er gyldig.

QSqlError

Denne klassen blir brukt for å gjøre det mulig å utføre spørringer og sjekke om de er gyldig. Dette gjør det lettere å programmere fordi da kan vi få kontroll over når man har lov til å utføre spørringer og om alt går som det skal i programmet. Forsikring om at databasen har koblet seg riktig og om spørringen er gyldig og dermed har lov til å bli utført og gi resultater.

4.7 Brukergrensesnitt

Brukergrensesnittet er lagd i Qt-creator med fokus på brukervennlighet. Lagerarbeideren skal kunne gjøre valg og skanne deler ved å bruke brukergrensesnittet. Valgene lagerarbeideren tar skal lagres i databasen, slik at ved skanning så vet lagerarbeideren hvilke valg som tilhører delen som skal skannes. Det har vært mange diskusjoner og vurderinger på hvor avansert brukergrensesnittet skal være. Valget falt på at brukergrensesnittet skal være på norsk, brukervennlig og ha et Windows-design.

Mesteparten av tiden har blitt brukt på å gjøre undersøkelser og finne løsninger på hva vi skal bruke, ikke bruke og hvordan vi skal bygge dette opp. Målet har alltid vært å bygge opp brukergrensesnittet med de kravene Tronrud har gitt oss, og koble dette opp mot databasen og integrere gjenkjenningsprogrammet til brukergrensesnittet. Akkurat som ting er nå, så er statusen slik at brukergrensesnittet, databasen og gjenkjenningsprogrammet virker kun for seg selv, altså som 3 separate systemer.

4.7.1 Qt Creator

Qt Creator er en kryss-plattform som er integrert og lagd for maksimal utvikleropplevelse i utviklermiljøet. Denne type kryss-plattform er bedre kjent som **IDE**. Qt Creator kan kjøres på Windows, macOS skrivebord-operativsystemer og Linux. Den gir utviklere muligheten til å lage applikasjoner som er egnet til smarttelefoner, PCer og andre integrerte plattformer. Qt Creator har muligheten til å koble seg direkte til GitHub, integrert visual editorer for utvikling av C++ baserte widget applikasjoner og har sin egen test og debug applikasjon integrert slik at utvikleren kan gjøre test og debugge kode uten å måtte bruke en ekstern program.

Vi valgte å bruke Qt Creator (inkludert designer vindu) da det er mulig å laste ned åpen kilde versjonen. At den kan kjøres på Windows, ettersom det var et krav fra bedriften, og at det er et av de mest brukte programmene i verden som vil si at det finnes mange gode dokumentasjoner. Det finnes også andre versjoner av Qt, som for eksempel Qt Quick, men dette egner seg for mindre skjermer som for eksempel smarttelefon. I Qt Creator har utvikleren mulighet til å ikke bare programmere et brukergrensesnitt, men også se hvordan den visuelt vil se ut ved hjelp av designervinduet. De første ukene jobbet vi med testing av Qt og hvordan Qt fungerer, da ingen i gruppa har tidligere erfaring med Qt. Etter omfattende undersøkelser, dokumentasjonlesing og testing skjønte vi etterhvert hvordan vi kunne bruke Qt Creator til å bygge opp et brukergrensesnitt.

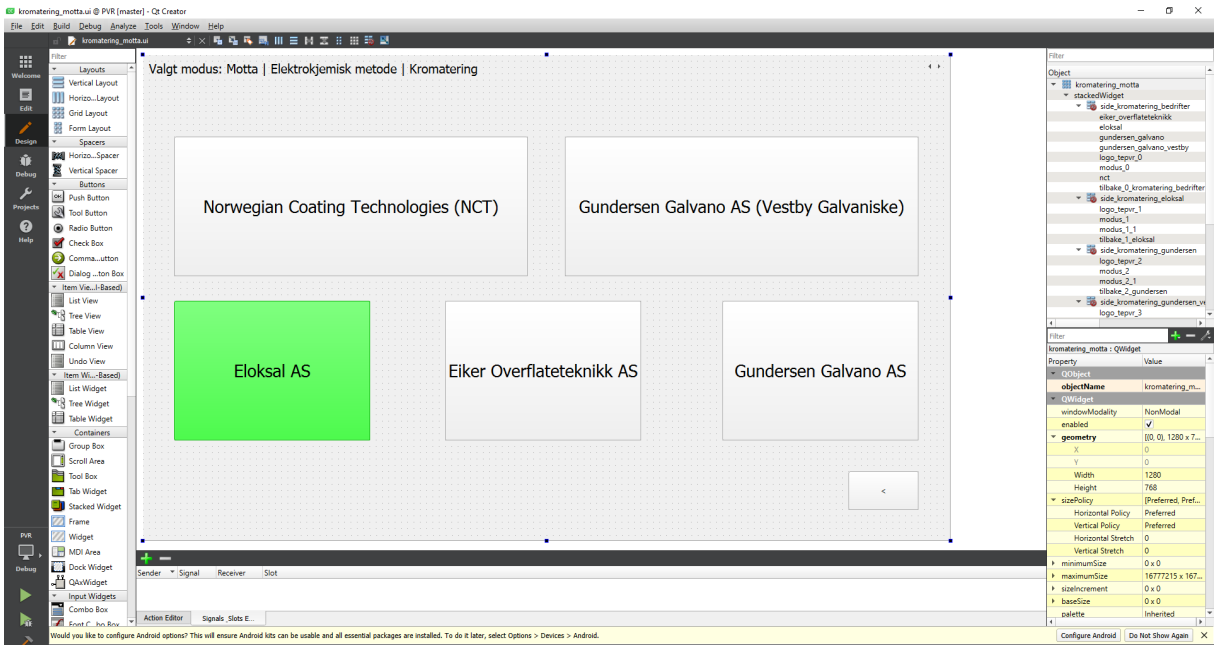
4.7.2 Pugh Matrise av verktøy

Kriterier	Qt Creator	Qt Quick	React Native	wxWidgets
Designer vindu	✓	✓	✓	✓
Windows-operativsystem	✓	✓	✓	✓
IOS/Android	X	✓	✓	✓
Datamaskin	✓	X	X	✓
Smarttelefon/Tablet	X	✓	✓	✓
Integrert debugger	✓	✓	✓	X
Dokumentasjon	✓	✓	✓	✓
Moderne	✓	✓	✓	X
Passer vår produkt	✓	X	X	X
Totalt	**7 av 9**	**7 av 9**	**7 av 9**	**5 av 9**
I prosent	**77.77%**	**77.77%**	**77.77%**	**55.55%**

Tabell 9: Pugh Matrise av verktøy til brukergrensesnitt

Her ser vi en Pugh matrise av de forskjellige brukergrensesnittverktøyene som finnes. Som nevnt er Qt Creator og Qt Quick av samme leverandør, men baserer seg på forskjellige felter. Pugh-matrisen er for å kunne sammenligne mellom de forskjellige verktøyene, og deretter velge den vi mener passer oss best. I matrisen har vi to verktøy som er fra Qt, en med navnet "React Native" og "wxWidgets". React Native er en åpen kildekode for mobilapplikasjoner og er funnet opp av Facebook. For å kunne jobbe med React Native, er vi nødt til å skjønne Javascript. wxWidgets er skrevet i C++ og stammer helt tilbake fra 1992. Som vi kan se i tabellen har de alle positive og negative sider ved seg. For oss, er det et par ting som er viktigere enn andre. Et av de er at brukergrensesnittet skal fungere på en Windows datamaskin. Siden det finnes smarttelefoner med Windows operativsystem, er det viktig å spesifisere at vi fokuserer på Windows operativsystem som er til datamaskiner. Dokumentasjon på dette er lett tilgjengelig, slik at vi til en hver tid har mulighet til å slå opp ting vi lurer på. Det er også viktig at det er moderne slik at vi alltid har de nyeste funksjonene og den nyeste programeringsmetodikken til systemet. Til slutt er det en totalvurdering på om dette passer til produktet vårt. I følge tabellen kan vi se at Qt Creator, Qt Quick og React Native ligger likt. Selv om disse tre verktøyene ligger likt, så er forskjellen på de at Qt Quick og React Native er egnet for mobilapplikasjoner. Utviklerprogrammet for Brukergrensesnitt wxWidgets faller under disse tre i matrisen da vi ikke følte den var moderne nok for produktet og at det ikke var så veldig mye dokumentasjon

å finne på disse programmene. Som nevnt er Qt Creator et av de mest brukte verktøyene i verden, så med det er det mer dokumentasjon og ofte nyere og mer effektive løsninger.



Figur 37: Qt Creator - Designer vindu

4.7.3 Widget

Brukergrensesnittet til PVR systemet vårt er en *Qt widget applikasjon* som er for datamaskiner, og inkluderer designervindu. Der vil klassebiblioteket til programmet få en egen *.ui* fil, hvor utvikleren kan designe hvordan brukergrensesnittet skal se ut. Deretter kan de forskjellige knappene få funksjoner, som må programmeres i de respektive filene. Det finnes en annen type applikasjon som heter *Qt konsoll applikasjon*. Denne er også egnet for datamaskiner men kommer uten designervindu og kun en enkelt *main.cpp* fil. Siden vi valgte å gå for *Qt widget applikasjon* fikk vi også muligheten til å bruke designervinduet til å kunne visualisere brukergrensesnittet. I dette designervinduet finnes det forskjellige widgets.

En widget er en liten modul som er nyttig. Det finnes forskjellige widgets til forskjellig bruk. På venstre side i designervinduet vil vi finne forskjellige widgets som vi ønsker å bruke i brukergrensesnittet. Det kan være knapper, lister, tekstbokser og mye mer. For å kunne bruke disse widgetene i designervinduet, så er det bare å dra de fra menyen og slippe den i designervinduet der utvikleren ønsker å ha widgeten plassert. Etter å ha plassert widgeten, er vi nødt til å programmere hva som skal skje med den valgte widgeten. Designer

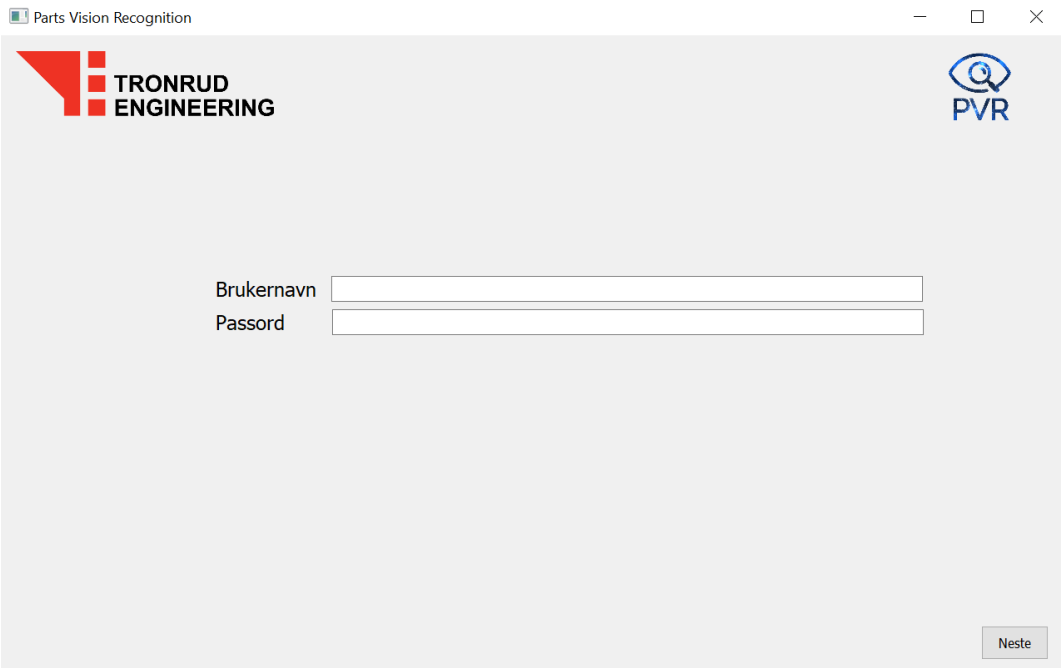
vinduet er veldig godt egnet for de som enten har høy interesse eller erfaring med UX. Dette er noe ingen av datastudentene har erfaring med på fra før av, så dette er en ting vi har måtte satt oss inn i.

4.7.4 Klasser

Det er mange funksjoner som er mulig å få til ved å bruke Qt Creator. Fokuset har vært å lage et brukergrensesnitt som møter kravene fra bedriften. Brukervennlig og mulighet for videreutvikling har også vært høyt prioritert. Derfor startet vi med å lage et design av brukergrensesnittet, slik at vi og bedriften kunne se hvordan utkastet av brukergrensesnittet skulle bli. Vi fikk positive tilbakemeldinger samtidig som at at noen endringer som måtte gjøres. Qt krever en spesiell måte å utplassere en ferdig programmert programvare på, og dette har vært en problemstilling helt siden vi startet med å bygge opp brukergrensesnittet.

I starten hadde vi samlet alt under en klasse, noe som begynte å bli veldig rotete ettersom det begynte å bli flere klasser. For å ikke blande alle klassene, splittet vi de opp i forskjellige klasser som senere ble til et klassebibliotek. Dette ga oss oversikt over de forskjellige klassene vi jobbet med, og en mulighet for å gjøre endringer, mye mer effektivt. Vi har også tatt hensyn til at bedriften skal få muligheten til å videreutvikle systemet hvis de ønsker det. Det finnes også mange andre metoder å bygge opp et brukergrensesnitt på. En mer effektiv måte å gjøre det på er å ha mindre klasser men dette er første gangen vi har vært med på utvikling med Qt. I senere tid har vi funnet ut at brukergrensesnittet kunne ha vært utviklet på en annen måte. Det hadde muligens vært mer fornuftig å starte med databasen først og deretter bygge opp brukergrensesnittet ut ifra det. Vi kunne også ha redusert antall klasser ved å kategorisere de på en annen måte fra starten av. Det vi kan se nå etter flere timer og måneder med brukergrensesnitt og Qt Creator, er at det er rom for utvikling og forbedring. I PVR-prosjektet har vi nå 167 klasser, hvor hver klasse representerer både "send" og "motta" av en overflatebehandlingstype. Det vil si at hver overflatebehandlingstype har to klasser som tilhører den. En klasse for sending av pall, og en klasse for mottagelse av pall.

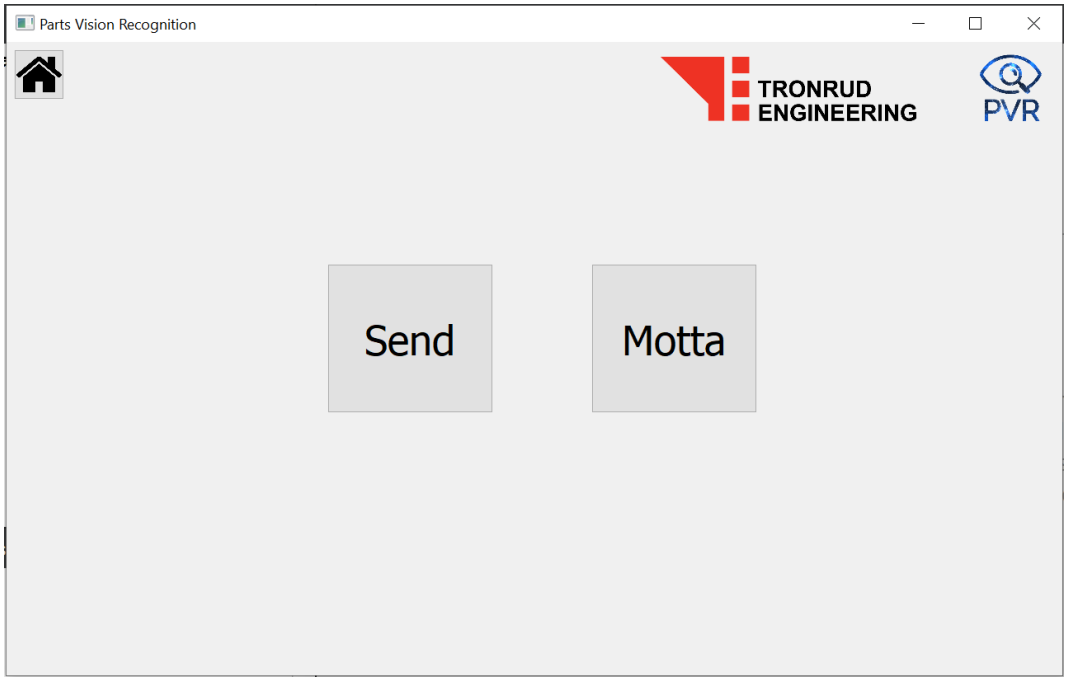
4.7.5 Visualisering av brukergrensesnittet



Figur 38: Første side - Logg inn (Hjem)

Hjem vinduet skal gi brukeren muligheten for å logge inn slik at administrasjonen vet hvem som har operert systemet. Når innloggingen er vellykket vil brukeren sendes videre til vindu nummer 2, som er valg mellom "send" og "motta". Dette gjør det lettere å kunne skille mellom deler som skal sendes for overflatebehandling og deler som har kommet tilbake fra overflatebehandling.

Foreløpig er det satt inn en knapp "Neste" for å kunne klikke seg videre i brukergrensesnittet, da databasen ikke er har blitt satt sammen med brukergrensesnittet enda. Så fort dette er gjort, vil "Neste" blir byttet ut med "Logg inn" og brukeren kan få klikke videre så fort brukeren logger seg inn med et gyldig brukernavn og passord. For å gjøre endringer i selve logg inn informasjonen, så må dette gjøres manuelt i databasen. Hvis firmaet ønsker å operere systemet uten å logge seg inn, så er det mulig å deaktivere "logg inn" funksjonen via Qt Creator.

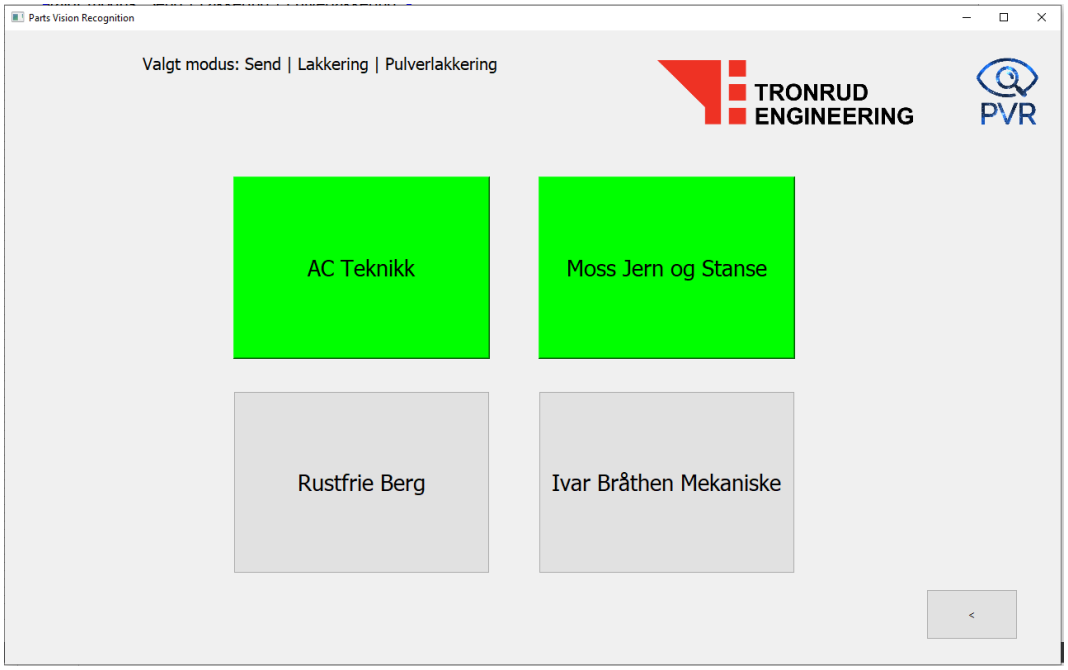


Figur 39: Valg mellom "Send" og "Motta"

Brukeren starter med å logge inn og deretter velge pallen skal sendes eller mottas fra overflatebehandling. Når dette valget er tatt, skal brukeren velge hvilken type overflatebehandling dette gjelder og deretter spesifisere behandlingstypen. Overflatebehandlingene er kategorisert utifra om det er *lakkering, elektrokjemisk metode, kjemiskmetode, herding* eller *annet*. Deretter splittes de inn i spesifikke typer.

I "Send" og "Motta" vinduet ser vi logoen til bedriften **Tronrud Engineering AS** og **Parts Vision Recognition**. I starten designet vi brukergrensesnittet uten logo, og fikk fort vite at det er noe bedriften ønsket å ha, og da ble den implementert inn sammen med logoen vår. Det samme gjaldt "hjem" knappen som vi kan se øverst til venstre i figur 39. Da vi designet brukergrensesnittet i starten hadde vi ingen "hjem" knapp. Etter å ha testet brukergrensesnittet litt, skjønte vi at det lønner seg med "hjem" knapp som skal kunne avslutte hele prosessen i tillegg til å logge ut brukeren. Da vi implementerte dette uten kobling til database lagde vi en knapp i designervinduet og programmerte den til å gå tilbake til "hjem"-vinduet, som vi kan se i figur 38.

Tronrud likte tanken bak "hjem" knappen, men ønsket at vi skulle ha at et "hjem" ikon i knappen, istedenfor at det står "hjem". Dette kan vi som sagt se i figur 39.

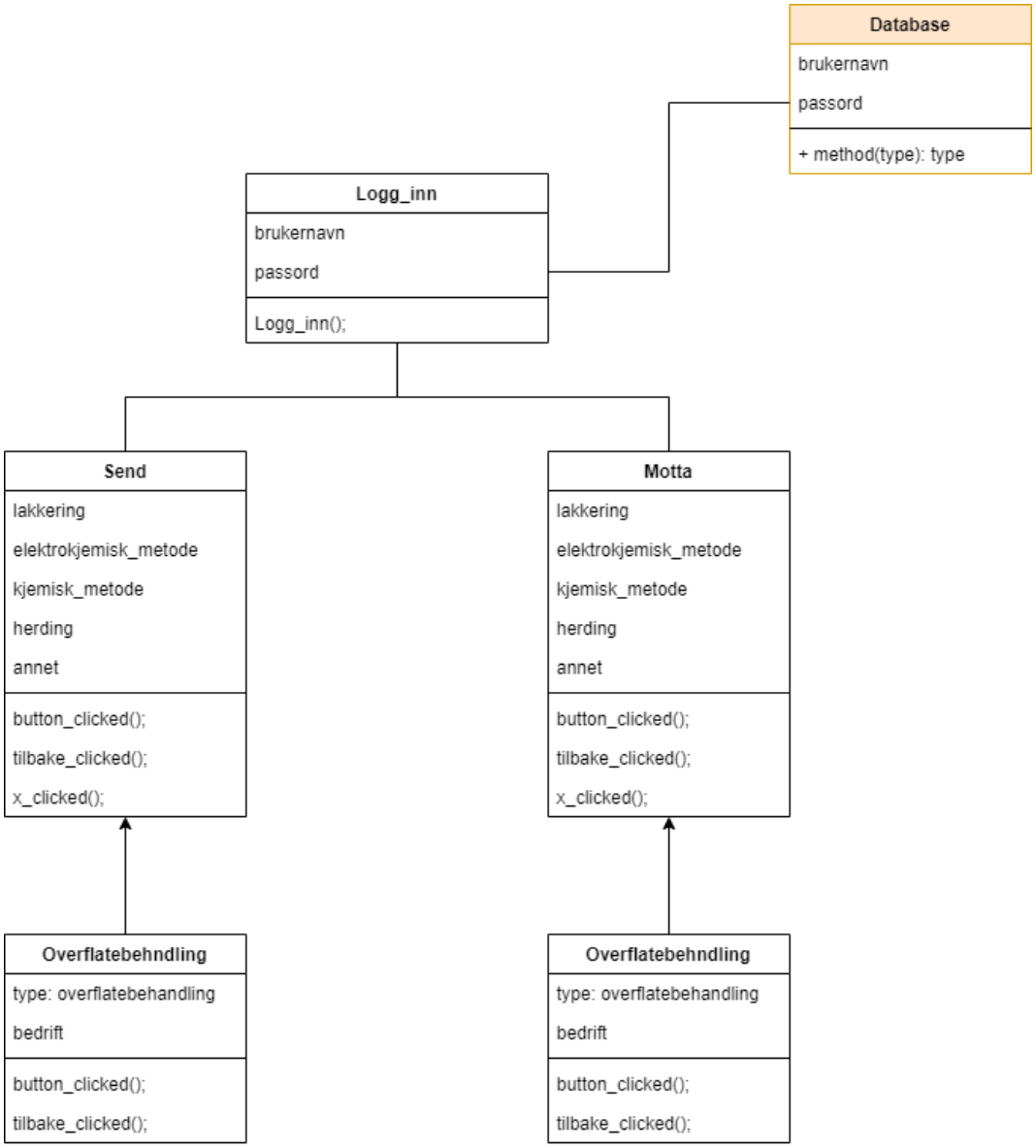


Figur 40: Ved valg av bedrifter

Det vi kan se indirekte er at det blir brukt elimineringsmetoden for å nøyaktiggjøre valgene brukeren tar, og for at det skal bli mindre feil ved valgene som blir gjort. Øverst i brukergrensesnittet kan vi se at det står "**Valgt modus: Send — Lakkering — Pulverlakkering**", dette er for å vise brukeren hvilke tidligere valg brukeren har gjort. Her ser vi at brukeren har valgt "Send" siden det er en pall med deler som skal sendes til overflatebehandling. Etter dette har brukeren valgt kategorien "Lakkering" siden delene skal lakkeres og tilslutt spesifisert om at delene skal pulverlakkeres. Når disse valgene har blitt valgt, gjenstår det kun å velge hvilken bedrift delene skal sendes til. Det er forskjellige bedrifter til forskjellig type overflatebehandlinger. Noen firmaer utfører flere type overflatebehandlinger, så det er viktig å spesifisere hvilken overflatebehandling det gjelder og hvilken bedrift delene skal sendes til. I figur 40 vil vi se at to av fire bedrifter er markert i grønn farge. Grunnen til dette er for å vise brukeren hvilke bedrifter som er de mest prioriterte av bedriften. På denne måten vil det være lettere for brukeren å velge riktig. Liste over prioriterte bedrifter varierer utifra hvilken type overflatebehandling det er snakk om.

En annen viktig detalj som er viktig å bemerke seg, er at vi har integrert en "Tilbake"-knapp. I første utkast av brukergrensesnittet sto det "Tilbake" i knappen, men bedriften ønsket at vi heller skulle bruke tegnet <, som vi kan se nederst til høyre i figur 40. Grunnen til at vi har en "tilbake"-knapp, er for å gi brukeren mulighet til å gå et steg tilbake og endre på valget, i tilfelle det første valget var feil. Så, for å komme seg tilbake til "Send og "Motta" siden, så trenger brukeren bare å klikke seg 2 steg tilbake.

4.7.6 Diagrammer

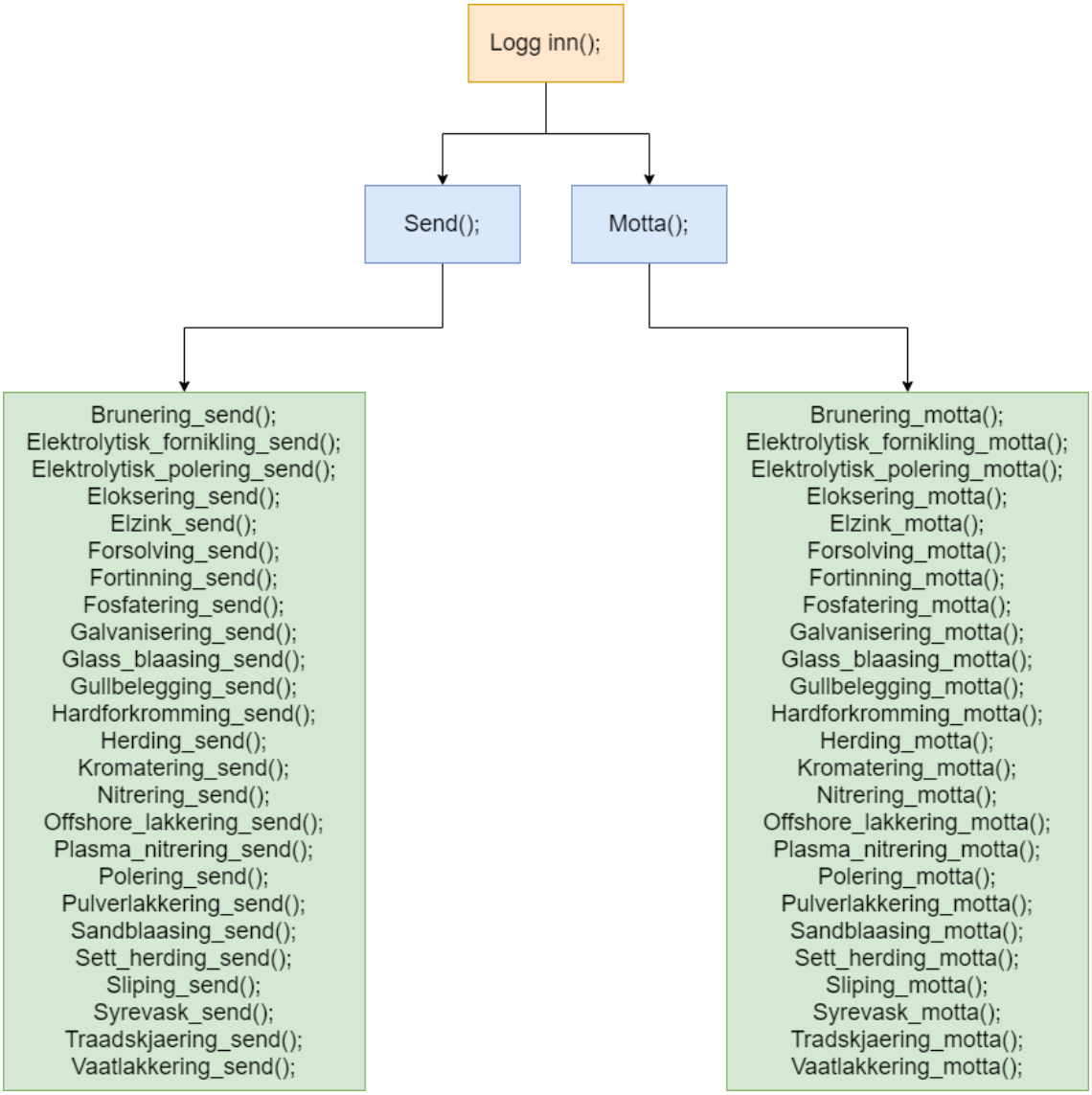


Figur 41: Klassediagram av brukergrensesnittet

Klassediagram til brukergrensesnitt er nødvendig, og derfor har vi lagd et veldig abstrakt klassediagram som viser noen viktige variabler og funksjoner i brukergrensesnittet. Det er viktig å bemerke seg at databasen skal gjøre en mye større oppgave enn kun å logge inn. Men da dette er relevant til første vindu i brukergrensesnittet, er den koblet direkte dit. Noen funksjoner som gjentar seg ofte er "button_clicked();", "tilbake_clicked();" og "x_clicked();". "Button_clicked" er en funksjon som vil gå igang når knappen har blitt trykket på, og det samme gjelder "tilbake_clicked" knappen. "x_clicked();" representerer "avslutt/tilbake til hjem", som vil si at brukeren blir flyttet direkte til første "logg inn" vindu hvis denne knappen blir trykket. I brukergrensesnittet har denne knappen en "hjem-ikon". "Tilbake" og "x" knappene i programmet har et tall som er inkludert i variabelen. Hvis variabelen er "tilbake_0_send", så betyr det at knappen er i index 0 i "send" klassen. Index 0 betyr side 1. Variablene er navngitt på denne måten på grunn av at ved videreutvikling, vil det være lettere å forstå hva denne variabelen gjør. Disse index-nummerene vil man kunne se ofte i de fleste variablene, nettopp for å skjønne hvilken side widgeten er på.

Diagram 42 viser en oversikt over hvordan de forskjellige klassebibliotekene er koblet sammen med "Send" og "Motta". For at bedriften skal få gjøre endringer ved eventuell videreutvikling, har vi lagd et klassebibliotek for hver av overflatebehandlingsmetodene. Dette gjør at de kan gjøre endringer i den tilhørende filen. Klassebibliotekene er navngitt utifra navnet på overflatebehandlingstypen. Hvert klassebibliotek er markert med **_send** eller **_motta** for å kunne skille mellom "Send" og "Motta". Klassen som tilhører "*Logg inn();*" er **mainwindow();** i koden.

Noen av navnene til overslatebehandlingmetodene har bokstaver som *æ, ø, å*, og dette er bokstaver som vi ikke kan bruke når vi skal navngi klassene, da programmet ikke aksepterer andre bokstaver enn de standard 26 engelske bokstavene. Så, for å kunne navngi klassene på norsk, har vi brukt "*aa*" for "å", "*o*" for "ø" og "*ae*" for "æ". Et eksempel på dette er overflatebehandlingen "*Trådskjæring*" som har fått navnet "*Traadskjaering*" i klassen.



Figur 42: Diagram av klassebiblioteket

Disse klassene er koblet ved hjelp av widgeten *QStackedWidget*. Overflatebehandlingsklassene i de grønne boksene er koblet til "Send" og "Motta" klassen, som i gjen er koblet til hovedklassen "Logg inn".

5 Videreutvikling

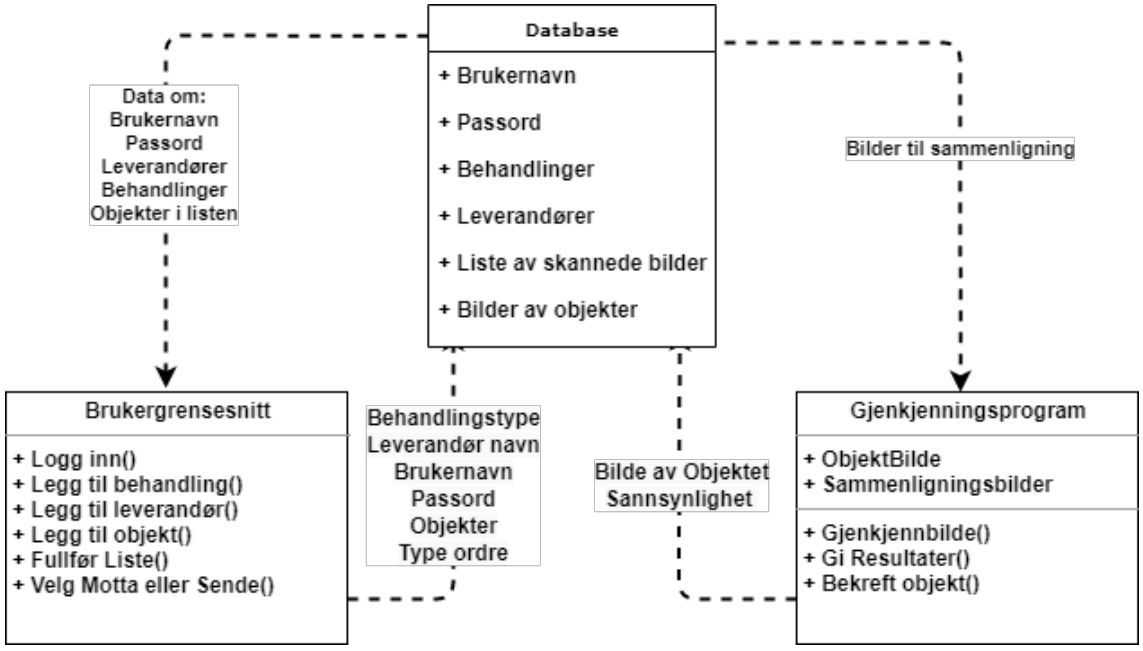
5.1 Maskin

Som videreutvikling av systemet vårt mener vi at et mekanisk sikkerhetsystem må på plass for at Tronrud sine krav skal bli tilfredsstilt. Dette betyr at vi skulle gjerne ønsket å designe et system som også kan bremse mekanisk dersom motorkraften går bort, kjedene ryker eller kuleskruene svikter. Dette kan f.eks gjøres ved hjelp av den den lineære føringen eller ved en av akslingene til saksebeina. Vi ser på det som nødvendig å ha mulighet til at systemet kan bremse uavhengig av motorkraft. Et annen ting som går under sikkerhet er å videreutvikle systemet som unngår klemfare. Dette kan gjøres ved at dekslinga av systemet gjøres på en annen måte, eller ved føring av klemlistene over et større område. Når disse sikkerhetsaspektene har blitt jobbet med er det også mulighet for å gå ned på kjedestørrelse. Kjedene er overdimensjonert nettopp på grunn av sikkerhet. Det å enten jobbe med å spare vekt eller å forsterke saksebeina, ville det gitt muligheten for at systemet kan løfte delene høyere enn det systemet kan gjøre per dags dato.

5.2 Data

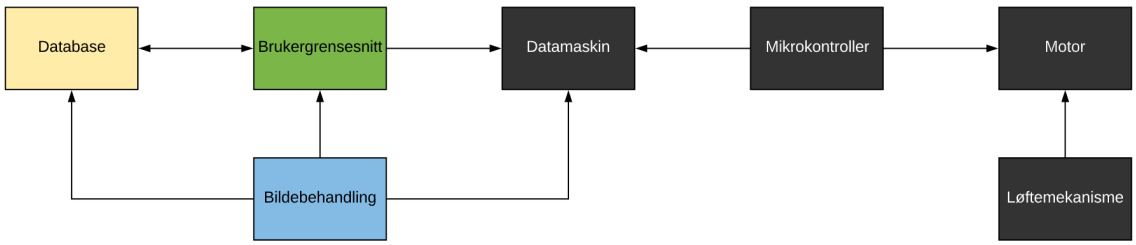
Vidreutvikling av systemet bør skje på styring av løftemekanismen, hvor det å finne ut av alle deler som hører til den delen av systemet er kritisk for å kunne fortsette arbeidet med styringen. Etikett utskriving har vi ikke kommet noe vei på, så det må plasseres grundig fokus på denne funksjonen dersom det skal være en del av sluttproduktet. Gjenkjenningsprogrammet bør stress testes slik at avvik i det vi anser som god gjennomførelse av oppgaven til programmet blir fikset. Ved tanke på database må eventuell videreutvikling fokusere på å være sikker på at SQL spørring gir riktig data som har blitt forespurt. Tilkobling av database bør bli forklart basert på hvilke operativsystem som blir brukt på maskinen. Tabeller bør også bli tilpasset hvis det viser seg at det er behov for mer data.

Brukergrensesnittet vikrer for seg selv, men trenger å få integrert de andre systemene inn. Det finnes andre deler av brukergrensesnittet som også kan videreutvikles som nevnt tidligere. Etter integrering av de andre systemene, trenger det å legges inn knapper for å starte for eksempel gjenkjenningsprogrammet. Dette blir en del av videreutvikling. De 167 klassene kan også kanskje slås sammen og kategoriseres på en annen måte, slik at det blir mindre klasser å forholde seg til. Etter integrering av systemene, kan brukergrensesnittet utplasseres som .exe fil.

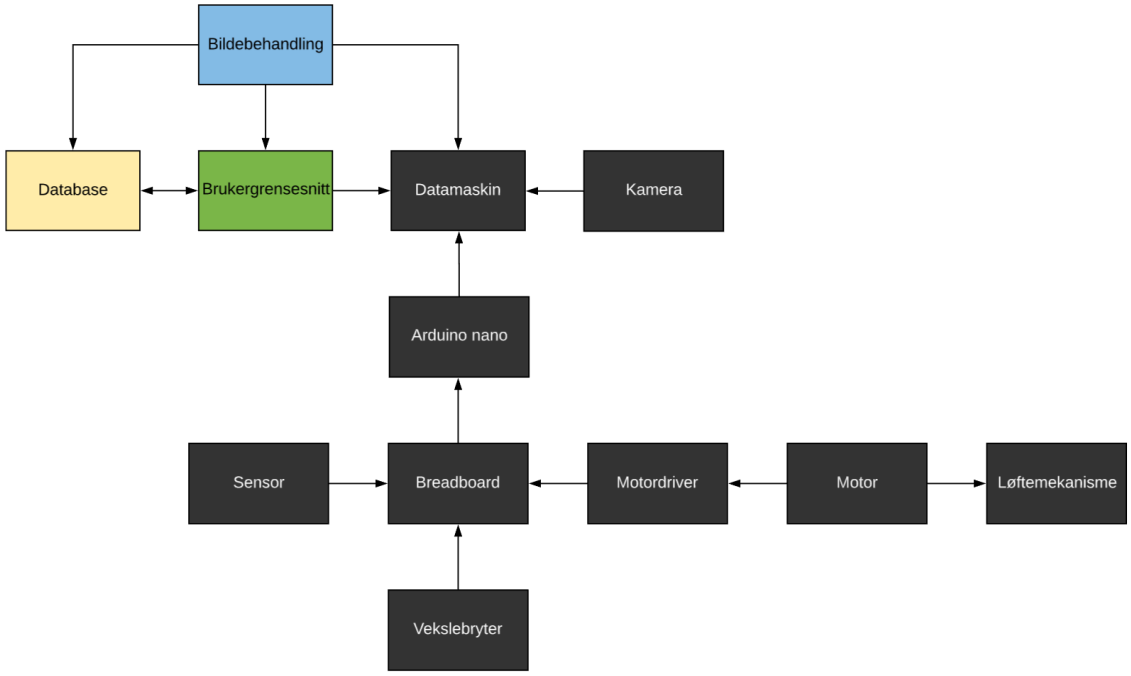


Figur 43: Abstrakt klassediagram av hele systemet

I figur 43 kan vi se klassediagrammet på hvordan de tre systemene bør kobles opp mot hverandre. Systemene virker hver for seg, men må integreres i hverandre for at det skal bli et helt system. For å få et bedre bilde av hvordan den fysiske oppkoblingen blir, har vi lagd blokk diagram som skal viser oppkoblingen abstrakt og litt mer utvidet. Vi har også inkludert løftemekanismen som en fjerde system, da den er koblet til datamaskinen for at Arduinoen skal få strøm. Programvaren til arduinoen må programmeres i C i Arduino IDE og utplasseres enten på en arduino Nano/MEGA eller permanent programmeres i en integrert kretskort. Styringen av løftemekanismen skal ikke kontrolleres av datamaskinen. Løftemekansimen kan selvfølgelig programmeres inn i en sterkere kretskort som for eksempel FPGA (VHDL), men da dette er enkle signaler som "løft opp" og "løft ned", så bør det holde med Arduino. Arduiono alene kan ikke gi nok strøm til løftemekanismen, og trenger derfor en strømforsyner. Arduinoen er kun "hjernen" til løftemekanismen.



Figur 44: Abstrakt blokkdiagram



Figur 45: Utvidet blokkdiagram

Blokkdiagrammet i figur 44 viser oss en veldig abstrakt versjon av hvordan hele system skal være koblet. Som nevnt er det foreløpig tre systemer, database, brukergrensesnitt og gjenkjenning (bildebehanling). Det er derfor de er markert i tre forskjellige farger. De mørke blokkene representerer komponenter som vi trenger. Vi har fått utlevert en Microsoft Surface av bedriften, men om den er sterk nok til å kjøre et komplett system er usikkert. Hvis vi ser på figur 45 vil vi se de komponentene vi mener det trengs for å bruke systemet PVR. Kameraet vi har brukt for testing er Logitech C920 HD PRO webkamera som har fungert helt utmerket. Så vi mener at bedriften ikke trenger å bruke et større budsjett på kamera.

5.2.1 Programvare koden

Koden til gjenkjenningsprogrammet:
`https://github.com/kirisanm/Parts_Vision_Recognition/tree/master/Gjenkjennings%20programmet`

Koden til databasen:
`https://github.com/kirisanm/Parts_Vision_Recognition/tree/master/Database`

Koden til brukergrensesnittet:
`https://github.com/kirisanm/Parts_Vision_Recognition/tree/master/UI/PVR`

6 Konklusjon

Målet vi hadde satt for dette prosjektet var å ferdigstille et produkt etter bestilling fra Tronrud. Vi la ned mye arbeid i utviklingen av et konsept som tilslutt måtte forkastes etter at Tronrud bestemte seg for at de ønsket bedre tilgang på pallen for lagerarbeideren. Dette resulterte i at vi måtte begynne utviklingen av et nytt konsept relativt sent i prosjektfasen som medførte at vi ikke har kommet like langt i produktutviklingen som ønsket. Som nevnt tidligere har den globale pandemien ført til at ferdigstilling av produktet ikke hadde latt seg gjennomføre allikevel.

Prosjektet bærer preg av at arbeidsoppgavene har blitt gjort separat, noe som har ført til at sammenslåingen av programvaredeler har vært vanskelig. Uten å ha kunnet jobbe fysisk sammen har det vært vanskeligere å forstå hvordan deler av systemet skulle bli integrert sammen. For å lage et komplekst system som skal gjenkjenne deler basert på et egen definert dataset har ført til utfordringer som vi har jobbet hardt med å løse. Selve gjenkjenningsproblemstillinger har også vært komplisert, ofte pleier man å kunne trene om maskiner til å gjenkjenne bilder, men i vårt tilfelle har oppgaven blitt spesifisert slik at gjenkjenningsprogrammet skal kunne oppdage deler basert på ett bilde.

Gruppen har fått til å gjenkjenne deler basert på ett bilde og finne det riktige bilde utifra ett sett med bilder av forskjellige deler. Bildebehandling trenger mye tid til å prosessere bilder, derfor har gruppen brukt en database til å lagre bildene fra deler som har blitt skannet på en bestemt dato, dette har ført til at vi kan redusere potensielle kandidater for deler som trenger å bli sammenlignet noe som kan redusere tiden for bildebehandlingen. Databasen har blitt designet rundt dette men problemene med integreringen av programvarene har gjort dette alternative ikke har blitt implementert.

Prosjektgruppen har tilrettelagt for videreutvikling av prosjektet eller ferdigstille produktet om arbeidsgiveren ønsker. Prosjektet har vært lærerikt for oss alle, da vi har måtte lære oss å jobbe som en gruppe mot ett konkret mål. Videre fram til presentasjonen kommer prosjektgruppen til å jobbe videre med prosjektet hvor vi ønsker å ferdigstille elementer av systemet.

7 Referanser

[1] Kohara Gear Industry, (2015), *Gear Rack and Pinion*
https://khkgears.net/new/gear_rack.html

[2] Marano et al., (juni 2017), *Effects of Gear Manufacturing Errors on Rack and Pinion Steering Meshing*
https://www.researchgate.net/publication/319058875_Effects_of_Gear_Manufacturing_Errors_on_Rack_and_Pinion_Steering_Meshing

[3] Maloney, (november 2018), *Mechanisms: Lead Screws and Ball Screws*,
https://hackaday.com/2018/11/13/mechanisms-lead-screws-and-ball-screws/

[4] Skf.com, (april 2016), *SKF high-perfomance miniature ball screws - series SP*
https://www.skf.com/binaries/tcm:12-268714/0901d1968049c09f-New-Miniature-Ball-ScrewBrochure.pdf

[5] Eitel, (desember 2015), *Lead Screw Basics*
https://www.motioncontroltips.com/lead-screws/

[6] Slocum,(januar 2008), *Fundamentals of Design: Power Transmission Elements II*
https://web.mit.edu/2.75/fundamentals/FUNdaMENTALs%20Book%20pdf/FUNdaMENTALs%20Topic%206.PDF

[7] Ilango og Soundararajan, (2007), *Introduction to Hydraulics and Pneumatics*, Tilgjengelig på USN-Kongsberg Bibliotek

[8] Pop, Corina og Mogan, (oktober 2017),*Real-Time Object Detection and Recognition System Using OpenCV via SURF Algorithm in Emgu CV for Robotic Handling in Libraries*, International Journal of Modeling and Optimization, Vol. 7, No. 5, ss 265-269

[9] Boehm, Jan, Guehring og Brenner, (januar 2000), *Automated Extraction of Features from CAD Models for 3D Object Recognition*

[10] Han og Zhao, (august 2015), *CAD-based 3D objects recognition in monocular images for mobileaugmented reality*, Computers Graphics Volume 50, ss 36-46

[11] Musso, (16. mai 2017), *Image recognition with C#: and Emgu libraries*
https://www.codeproject.com/Articles/1187512/Image-recognition-with-Csharp-and-Emgu-libraries

[12] WhyDoMath, *Digital image basics*
https://www.whydomath.org/node/wavlets/index.html

[13] Fisher, Perkins, Walker og Wolfart, (2000), University of Edinbrugh.
https://homepages.inf.ed.ac.uk/rbf/HIPR2/

[14] University of Auckland, *Morphological Image Processing*
https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm

[15] Lars Aular, (9. august 2005), *Matematisk morfologi*, Norsk Regnesentral
https://www.uio.no/studier/emner/matnat/ifi/nedlagte-emner/INF3300/h05/undervisningsmateriale/handout-inf3300-2005-8-6pp.pdf

[16] Raju og Neelima, (januar 2012) *Image Segmentation by using Histogram Thresholding*, IJCSET
https://ijcset.net/docs/Volumes/volume2issue1/ijcset2012020103.pdf

[17] Duygulu, (2006), *Image Processing*, Bilkent University
http://www.cs.bilkent.edu.tr/~duygulu/Courses/CS554/Notes/ImageProcessing.pdf

[18] Cattin, (19-26. april 2016), *Image restoration*, University of Basel
https://miac.unibas.ch/SIP/06-Restoration.html

[19] Hitoshi, Masato og Izumi, (26. juni 2017), *Tree Crown Size Estimated Using Image Processing: A Biodiversity Index for Sloping Subtropical Broad-Leaved Forests*, Tropical Conservation Science Volume 10, ss 1–12
https://doi.org/10.1177/1940082917721787

[20] Moallem og Razmjooy (oktober 2012) *Optimal Threshold Computing in Automatic Image Thresholding using Adaptive Particle Swarm Optimization*
https://doi.org/10.1177/1940082917721787

[21] Lars Aular, (1. mars 2005), *Matematisk morfologi I*, Norsk Regnesentral
https://folk.uio.no/inf5300/morpho1ny.pdf

[22] Karami, Ebrahim og Prasad, (6. november 2015), *Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images*

[23] Haddad og Akansu, (mars 1991), *A Class of Fast Gaussian Binomial Filters for Speech and Image Processing*, IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 39, ss 723-727

[24] Bay,Ess, Tuytelaars og Van Gool, *Speeded Up Robust Features*, ETH Zurich, Katholieke Universiteit Leuven, Belgia

[25] Astrup, (2016), Metallkatalogen
https://astrup.no/content/download/5079/18525/version/9/file/1_Aluminium_LowRes.pdf.

[26] Tsubaki, About Us, (2020)
https://www.ustsubaki.com/about.html
Lest: 02.02.2020

[27] DMA Europa, "Tsubaki Zip-Chain Lifter Wins Energy Efficiency Award", 2010
: http://www.dmaeuropa.com/Default.aspx?TabId=1245&itemid=523 Lest: 15.02.2020

[28] IWIS, Industrial roller chains (2020)
https://www.iwis.com/en-gb/print-product/3222
Lest:20.02.2020

[29] Belastningsergonomi, (2019)
https://www.av.se/globalassets/filer/publikationer/foreskrifter/
belastningsergonomi-foreskrifter-afs2012-2.pdf

[30] NTN-SNR, (2020), *UCP-204 - Technical data*
https://eshop.ntn-snr.com/en/UCP-204-2249630.html

[31] Bickle, (2020), *VSTH 80/20K - Teknisk data*
https://www.blickle.no/produkt/VSTH-80-20K-535872

[32] QStackedWidget Class, (2020)
https://doc.qt.io/qt-5/qstackedwidget.html

[33] Signals og Slots, (2016)
https://doc.qt.io/archives/qt-4.8/signalsandslots.html

[34] Signals og Slots, (2020)
https://doc.qt.io/qt-5/signalsandslots.html

[35] OpenCV biblioteket
https://opencv.org/
Lastet ned: 20.01.2020 (version 3.4.10)

[36] SURF algoritmen basic tutorial
https://docs.opencv.org/master/df/dd2/tutorial_py_surf_intro.html
Lest: 13.04.2020

[37] OpenCV dokumentasjon
https://docs.opencv.org/3.4.10/index.html

[38] Brown, (2007-2010), *ZBar code reader*
http://zbar.sourceforge.net/ Lastet ned: 10.05.2020

[39] Rexroth, (01-03-2020), *Produkt katalog, R999001185*
https://www.boschrexroth.com/various/utilities/mediadirectory/index.jsp?language=en-GB&publication=NET&filterMediatype=1593&search_query=R999001184&search_action=submit&edition_enum=R999001185 Lastet ned: 15.04.2020

[40] Aratron.no, *Produktkatalog, G99TE14-1006*
https://aratron.no/wp-content/uploads/2014/12/HGW-serie.pdf Lastet ned: 15.03.2020

[41] IWIS, Industrial roller chains (2020)
https://www.iwis.com/en-gb/print-product/3425
Lest:20.02.2020