Høgskulen på Vestlandet

BACHELOROPPGAVE

Blandet virkelighet-basert tilbakemelding fra robot for virtuelle objekter

Mixed Reality-based feedback from robots for virtual objects

Fred Christiansen

Knut Erik Aspen

Krister Smelvær

Simen Farbu Swensen

Fakultet for ingeniør- og naturvitskap
Institutt for datateknologi, elektroteknologi og realfag

Informasjonsteknologi

Veileder: Per Christian Engdal

Innleveringsdato: 22.05.2023

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og

TITTELSIDE FOR HOVEDPROSJEKT

Rapport tittel:	*Dato: 18.01.2023*
Blandet virkelighet basert tilbakemelding fra robot for virtuelle objekter	
Forfatter(e):	*Antall sider u/vedlegg: 36*
Fred Christiansen, Knut Erik Aspen, Simen Farbu Swensen, Krister Smelvær	*Antall sider vedlegg: 52*
Studieretning:	*Antall disketter/CD-er:*
Informasjonsteknologi	0
Kontaktperson ved studieretning:	*Gradering:*
Per Christian Engdal	Ingen
Merknader:	
Ingen	

Oppdragsgiver:	*Oppdragsgivers referanse:*
HVL Robotics	*IF-1*
Oppdragsgivers kontaktperson(er):	*Telefon:*
Laurenz Elstner, Raquel Motzfeldt Tirach	

Sammendrag:

Målet med prosjektet er å lage en Blandet virkelighets basert programvare som kombinerer en fysisk robot og virtuelle objekter i et virtuelt miljø. Roboten skal kunne kollidere med disse virtuelle objektene og gi en fysisk tilbakemelding som om en fysisk kollisjon oppstod. Målet med dette er å gjøre det lettere å lære roboten i miljøer som er tidkrevende og vanskelige å lage til.

The goal of the project is to develop a mixed reality-based software which combines a physical robot and virtual objects in a virtual environment. The robot should be able to collide with the virtual objects and provide force feedback as if a real collision occurred. The goal of this is to make it easier to teach the robot in an environment that is time consuming and hard to set up.

Stikkord:

Blandet Virkelighet	Robot arm	Microsoft HoloLens 2

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og

Høgskulen på Vestlandet, Fakultet for ingeniør- og natuvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

FORORD

Rapporten dokumenterer utførelsen av bachelorprosjektet «Blandet virkelighets tilbakemelding fra robot for virtuelle objekter» ved HVL.

Vi ønsker å takke HVL Robotics for en spennende oppgave, god veiledning og hjelp under prosjektet. Vi ønsker også å takke intern veileder for tett oppfølging, gode råd og veiledning. Til slutt vil vi takke Campus Verftet for disponering av kontor plass, UR5e robot, Microsoft HoloLens 2 og diverse annet utstyr under utviklingen.

INNHOLDSFORTEGNELS

1. INNLEDNING

I dette kapittelet gis det en innføring i bakgrunnen til problemstillingen og videre forklaring på hvorfor den er nyttig å løse. Deretter introduseres prosjektets mål og eier.

1.1. Kontekst

I dagens løsning for å lære roboter nye oppgaver blir det laget fysiske miljø for å imitere miljøet oppgaven blir lært for. Det vil si at før roboten kan lære seg en ny oppgave nøyaktig, må det opprettes fysiske og presise konstruksjoner rundt roboten, for å tilnærme miljøet hvor roboten skal utføre oppgaven. Dette er svært tid- og ressurskrevende, spesielt ved komplekse miljøer med komplekse oppgaver.

Teknologien for samhandling av roboter med virtuelle miljøer er under stadig utvikling (Capgemini Research Institute, 2022), og nye metoder blir utviklet. Høgskulen på Vestlandet (HVL) sin robotikk gruppe HVL Robotics ønsker å utforske hvordan denne teknologien kan utnyttes for å spare tid og ressurser ved deres forskning.

1.2. Motivasjon

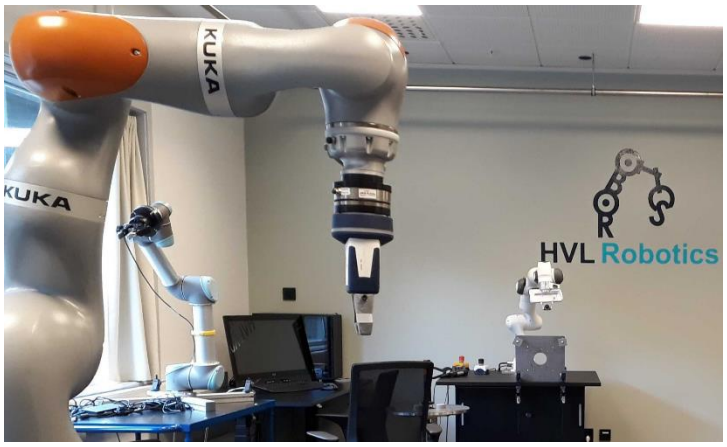
HVL Robotics er et forskerteam og vil kunne bruke så mye tid som mulig på forskningen, men slik det er nå er tilpasning av en robot sine bevegelser og operasjoner i et komplekst miljø tid og ressurskrevende i form av å bygge komplekse miljø rundt roboten. Simulering i et virtuelt miljø vil effektivisere og lette denne jobben betraktelig, og i ytterste konsekvens ikke kreve bygging av komplekse miljø i det hele tatt.

HVL Robotics må nå bruke riktige fysiske objekter, og sette opp riktig fysisk miljø før de kan begynne å lære roboter en oppgave. Dette krever mye tid og kan være ganske unøyaktig. De vil derfor ha hjelp til å lage en programvare som bruker blandet virkelighet til å kombinere en ekte robot og virtuelle objekter så dette problemet blir løst, slik at HVL Robotics kan bruke mer tid og ressurser til forskningen og jobbingen med robotene.

1.3. Prosjekteier

Oppdragsgiveren og prosjekteier for dette prosjektet er Laurenz Elstner og Raquel Motzfeldt Tirach fra HVL Robotics ved Høgskulen på Vestlandet. HVL Robotics er et forskerteam som fokuserer på robotikk. Mer spesifikt fokuserer de på samarbeidende og myke roboter. Deres forskning er både akademisk og rettet mot lokale bedrifter i Førde med behov for robotisering.

HVL Robotics holder stort fokus på menneskelig samarbeid og integrasjon med roboter. Dette inkluderer jobbing med sensorer, måling av menneske bevegelser og oversettelse av disse til robot bevegelse.



Figur 1.1: HVL Robotics (HVL Robotics Lab, 2023)

1.4. Problembeskrivelse og mål

Brukere av roboter og utviklere av robotprogramvare bruker i dag flere metoder for å implementere roboter i komplekse miljø med hindringer. Dette blir ofte svært tids og ressurskrevende, spesielt ved programmering av roboter og ved å sette opp riktig fysisk miljø rundt robotene.

Målsettingen med prosjektet er å lage en programvare hvor en bruker kan opprette virtuelle miljø med virtuelle objekter eller hindre rundt en fysisk robot, hvor roboten kan reagere ved kollisjon med, og benytte seg av de virtuelle objektene for å gjøre prosessen ved å lære roboter nye oppgaver mindre tidkrevende, ressurskrevende og enklere.

1.5. Oppbygging av rapporten

I de neste kapitlene vil det komme detaljer om prosjektbeskrivelse, design av prosjektet, valgt løsning, resultater, diskusjon og videre arbeid. Rapporten er delt inn i ni kapitler som har følgende oppsett:

Kapittel 1: Introduksjon til rapporten med mål, beskrivelse, og kontekst.

Kapittel 2: Her er det beskrevet i mer detaljer om bakgrunnen til oppgaven, avgrensinger og den initielle løsningen gruppen startet med.

Kapittel 3: Beskrivelse av brukstilfellet som representerer den initielle løsningen og valgene rundt dette.

Kapittel 4: Detaljert løsning av brukstilfellene hvor arkitekturen av programmet og koden blir beskrevet.

Kapittel 5: Evaluering av den endelige løsningen.

Kapittel 6: Diskusjon rundt valgene som ble tatt under prosjektet.

Kapittel 7: Konklusjonen av arbeidet og hvordan det kan utvikles videre.

Kapittel 8: Kilder og referanser.

Kapittel 9: Vedleggs kapittel.

Kapittel 10: Ordliste.

2. PROSJEKTBESKRIVELSE

I dette kapittelet blir det lagt frem en beskrivelse av prosjektets bakgrunn, initielle krav samt den initielle løsnings-idé. Kapitelet inkluderer også avgrensninger for prosjektet, ressurser og litteratur om problemstillingen.

2.1. Praktisk bakgrunn

2.1.1. Tidligere arbeid

Våren 2022 ble det gjennomført et bachelorprosjekt ved Høgskulen på Vestlandet for Trolltunga Robotics kalt Robotarm baneplanlegging med blandet virkelighet (Benjaminsen & Dankel 2022). I dette prosjektet ble det demonstrert hvordan man kunne styre en robot gjennom en MR-enhet, dette skulle gjøres ved å lage et hologram av roboten i MR. Dette hologrammet skulle være justerbart med håndbevegelser og synkroniseres med den fysiske roboten, i tillegg skulle man kunne se den planlagte banen til roboten før utførelse. Prosjektet resulterte i en løsning som kunne utføre plukk og plasser oppgaver raskere enn den integrerte løsningen i robotarmen UR5e. Løsningen til Benjaminsen og Dankel 2022 benyttet mange av de samme verktøyene som dette prosjektet, og ga inspirasjon og hint til hvordan visse utfordringer med f.eks kommunikasjon mellom de ulike komponentene kunne løses. Ingen av Benjaminsen og Dankels egenutviklede løsninger er implementert i dette prosjektet, da vi var nødt til å utvikle våre egne tilrettelagte løsninger, basert på samme grunnlag som at den forrige rapporten måtte utvikle sine.

2.1.2. Initielle krav

Prosjektet blir utført for HVL Robotics med hensikt om å bli brukt til videre forskning. Oppdragsgiver har presentert noen krav for løsningen og produktet:

Krav for prosjektet:

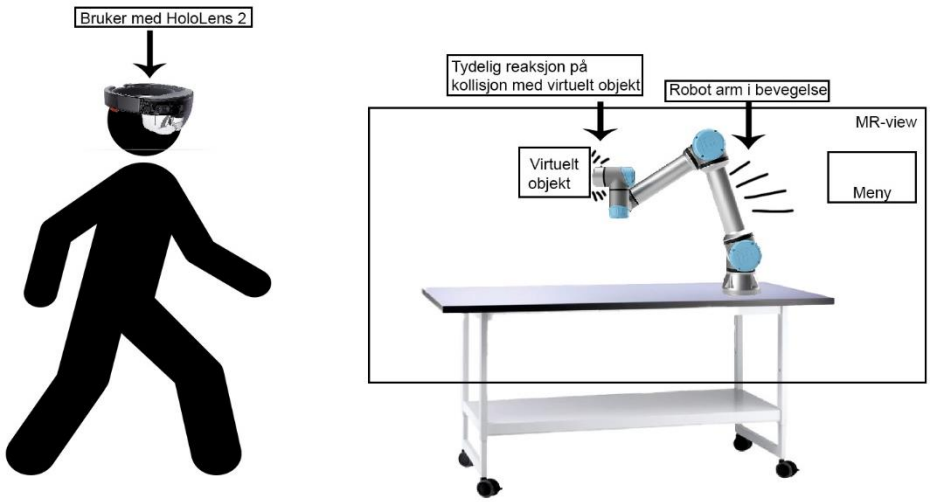
- Programmet skal utvikles for MR der robot og virtuelle objekter kommuniserer.
- Brukeren skal kunne opprette virtuelle objekter gjennom HoloLens2.
- Robot skal reagere med tydelig reaksjon på kollisjon med virtuelle objekter som om de kolliderte fysisk.

2.1.3. Initiell løsnings-idé

I figur 2.1 kan leseren se produktet i ett foreslått brukermiljø, dette vil vanligvis finnes på en robot lab. En bruker utstyrt med en MR- enhet, i dette tilfellet HoloLens 2, ser robotarm og bordet gjennom MR enheten. Brukeren oppretter ett virtuelt objekt i banen til roboten, når

roboten flyttes til dette objektet vil den gjøre en handling basert på simulert tyngde og størrelse hos objektet. Roboten kan flyttes rundt objektet, gjennom objektet eller spenne tilbake basert på forskjellige faktorer som fart, styrke og tyngde på objekt.

Ved tilstrekkelig tid er det tiltenkt at det blir utviklet et brukergrensesnitt der man kan velge en tidligere konfigurert robot, samt legge til og konfigurere ny robot i MR-synet. Dette kan gjøres gjennom en flytende meny i MR-synet, hvor brukeren kan nullstille roboten til standard posisjon. Det er også i denne menyen brukeren legger til nye virtuelle objekter, med forskjellige innstillinger som vekt, form og lengde.



Figur 2.1: Produktet i et brukermiljø.

2.2. Avgrensninger

Prosjektet avgrenses til at utvikling og testing blir bare utført på Microsoft HoloLens 2 og ikke andre AR-enheter da vi bare har tilgang på HoloLens 2 gjennom Campus Verftet AS.

Utvikling og testing blir også begrenset til en UR5e robot, da vi legger ett bra grunnlag ved å utvikle systemet for en type robot.

Det er fra oppdragsgiver blitt gjort begrensninger i hvordan roboten skal reagere på kollisjon med virtuelle objekter, som konkret blir begrenset til å stoppe ved kollisjon, eller rekylere basert på motstand. Dette er basert på tilgjengelig tid og ressurser, og det er gode muligheter for å utvikle dette videre.

2.3. Verktøy og ressurser

I dette prosjektet er de fleste virkemidler allerede bestemt i oppgaveteksten. Prosjektet planlegger å anvende ROS (Python) og Unity (C#/python) for utvikling av programvaren, Microsoft HoloLens 2 for visning av MR-hologrammer, og en robotarm som skal reagere på kollisjoner med virtuelle objekter.

GitHub vil bli brukt for kodedeling og versjonskontroll, og ferdige 3D-objekter fra Unity Store vil bli brukt for å spare tid og ressurser. Visual Studio vil være hoved IDE-en for utviklingen av programvaren, mens Clickup vil brukes for Gantt-planer og Sharepoint for deling av dokumenter.

2.3.1. Microsoft HoloLens 2

Microsoft HoloLens 2 er Microsofts løsning på blandet virkelighet kalt Mixed Reality (MR). HoloLens 2 kommer med funksjoner som hånd-sporing, øye-sporing, hode sporing, stemme aktivering, dybdesensor, kamera, mikrofon og rom kartlegging. Disse funksjonene lar hologrammer gli sømløst inn i omgivelsene, og lar brukeren samhandle med et virtuelt miljø. HoloLens 2 brillene blir lånt til gruppen fra Campus Veftet.

Figur 2.2: (Microsoft, 2023)

2.3.2. Unity

Unity er et multiplattforms utviklingsmiljø designet for 2D og 3D utvikling som bruker C# som programmeringsspråk. Unity lar brukeren bygge “scener” som inneholder alle de virtuelle objektene i systemet hvor de kan bli sendt til HoloLens 2 for å bli vist i et Mixed Reality (MR) miljø. Det brukes Unity versjon 2021.3 for utviklingen.

2.3.3. The Robot Operating System (ROS)

ROS Noetic brukes for å kontrollere roboten, ROS er en programvare med åpen kildekode som bruker C++ eller Python som programmeringsspråk. ROS blir brukt for å kommunisere mellom robot og Unity.

2.3.4. Universal Robot 5 e-series robot arm (UR5e)

UR5e er roboten som blir brukt til utvikling og testing da denne er tilgjengelig ved HVL campus og ved Campus Verftet AS. Denne robot armen er en fleksibel samarbeidende robot som kan utføre en rekke oppgaver og kan løfte objekter med en vekt opp til 5kg. For å kommunisere med roboten brukes UR RTDE.

Figur 2.3: (Universal Robots, 2023)

2.3.5. Operativsystem

For utvikling med HoloLens 2 kreves Windows 10, 11 eller Mac OS. Her bruker gruppen Windows 10 og 11 for utvikling. Linux er ikke kompatibel med HoloLens 2, men gruppen vil også bruke en virtuell maskin med Ubuntu versjon 20.04 installert for å kommunisere med roboten via ROS og UR RTDE.

2.3.6. Andre verktøy

2.3.6.1. Vuforia

En programvareutviklingspakke (SDK) som tilbyr funksjonalitet for AR. I dette prosjektet brukes funksjonene for bildegjenkjenning til å lese av AR-markører

2.3.6.2. UR RTDE (Universal Robots Real-Time Data Exchange)

Er et kommunikasjonsgrensesnitt som legger til rette for rask sanntidskommunikasjon mellom UR5e roboten og det eksterne operativsystemet.

2.3.6.3. MRTK (Mixed Reality Toolkit)

Et åpent kildekodeprosjekt fra Microsoft med en samling av ferdig utviklede scripts og komponenter som skal legge til rette for utvikling av MR-applikasjoner. I dette prosjektet er kontrollpanel og menyer i stor grad hentet fra denne pakken. Interaksjon med objekter i MR kommer også fra denne pakken.

2.3.6.4. Unity Robotics Hub

Unity Robotics Hub er en ressurs designet av Unity for integrering av robotikk simulering i Unity. Unity Robotics Hub inneholder blant annet pakkene TCP-endpoint, TCP-connector og URDF

importer som var spesielt nyttig for prosjektet. TCP pakkene gjør det mulig å benytte seg av TCP basert kommunikasjon via endpoint på ROS siden og connector på Unity siden. URDF importøren er en pakke som gjør det enkelt å importere ferdige robot hologram i Unity ved hjelp av URDF filer.

2.3.7. Ressurser

Grunnet prosjektets natur kreves det tilgang på MR-enhet og robot. Tilgang til dette er gjort mulig gjennom Campus Verftet AS og eller robot lab på HVL campus. Robot er tilgjengelig både ved Campus Verftet og HVL campus, MR-enhet må lånes fra Campus verftet, men kan bli brukt både ved Campus verftet og HVL campus. Gruppen ble disponert kontorplass ved Campus Verftet, og gruppen valgte å jobbe der fremfor skole campus.

Gruppen har fått tett oppfølging og veiledning fra oppdragsgivere Raquel Tirach og Laurenz Elstner. I tillegg har gruppen kunnet rådføre angående Unity-relaterte utfordringer med Sivert Benjaminsen. Intern veileder fra HVL, Per Christian Engdal, har veiledet på spørsmål angående rapport og generell utvikling.

2.4. Litteratur om problemstillingen

2.4.1. Blandet virkelighet (MR – Mixed Reality)

En forklaring av Blandet virkelighet vil bli lettere forklart sammen med virtuell virkelighet og Utvidet virkelighet.

2.4.1.1. Virtuell virkelighet (VR – Virtual Reality)

Virtuell virkelighet er en datamaskingenerert simulering av et tredimensjonalt miljø som kan oppleves av en person gjennom VR-briller eller andre enheter som skaper et oppslukende miljø. Brukeren kan samhandle med miljøet og kan være fullstendig fordypet i det, og føle at de er til stede i en annen verden. VR brukes ofte i spill, utdanning, treningssimuleringer og andre applikasjoner der en fullstendig oppslukende opplevelse er ønsket (Sheldon, 2022.).

2.4.1.2. Utvidet virkelighet (AR – Augmented Reality)

Utvidet virkelighet er digital informasjon som er lagt over den virkelige verden. AR bruker kameraer og sensorer på enheter som smarttelefoner, nettbrett eller briller for å oppdage gjenstander fra den virkelige verden og legge til digital informasjon til dem. Dette kan inkludere alt fra å legge til digitale animasjoner eller bilder til den virkelige verden, til å gi tilleggsinformasjon om et objekt i brukerens miljø. AR brukes ofte i markedsføring, underholdning, utdanning og andre applikasjoner der å legge til digital informasjon til den virkelige verden kan forbedre brukeropplevelsen. (Arena et al., 2022).

2.4.1.3. Blandet virkelighet (MR – Mixed Reality)

Blandet virkelighet kombinerer elementer fra både VR og AR. Det innebærer å skape et fullstendig oppslukende digitalt miljø som er integrert med den virkelige verden, slik at brukeren kan samhandle med både digitale og fysiske objekter i sanntid. I motsetning til AR, innebærer

blandet virkelighet mer sømløs integrasjon av den digitale og fysiske verden, slik at brukeren kan samhandle med begge miljøene samtidig. MR brukes ofte i industrielle, medisinske og pedagogiske omgivelser, der en kombinasjon av det digitale og fysiske miljøet er nødvendig for trening eller andre applikasjoner. (Adobe, 2023).

2.4.2. Praktisering av MR teknologi

Det har vært en stor vekst i artikler om VR, AR og MR de siste årene. Ett av fokusene MR basert forsking har hatt er fjernstyrte operasjoner og terapi. I 'A Novel Framework for Mixed Reality-Based Control of Collaborative Robot: Development study' (Shahria et al, 2022), fokuserte de på utviklingen av ett rammeverk for samarbeidende roboter. Dette rammeverket har en del likhetstrekk med det tidligere bachelorprosjektet til Sivert og Håvar (Benjaminsen & Dankel, 2022). Det disse har til felles med dette prosjektet er at det praktiseres MR teknologi i kombinasjon med samarbeidende roboter.

2.4.3. Force feedback

Tilbakemelding eller force feedback fra samarbeidende roboter er en teknologi som kan være veldig nyttig innenfor en rekke arbeidsområder. Minimalt invasiv kirurgi er et eksempel der force feedback kan være nyttig for å identifisere unormaliteter i hudvev (Tholey et al, 2005). Det samme er sant for fjernstyrt kirurgi ved hjelp av samarbeidende roboter, derfor ser man fordelen med tilbakemeldingen som blir utviklet i dette prosjektet.

3. DESIGN AV PROSJEKTET

3.1. Tilnærmet løsning

Basert på oppgaveteksten og virkemidlene som er bestemt, planlegges følgende løsning:

1. Utvikling av en Mixed Reality (MR) programvare ved hjelp av Unity (C#) som simulerer et miljø for roboten, hvor den kan samhandle med virtuelle objekter og utføre spesifikke oppgaver.
2. Integrere ROS (Python) i bakgrunnen for å kommunisere med roboten gjennom UR RTDE og sørge for at den kan motta data om kollisjoner og for å motta data fra roboten.
3. Programvaren vil kreve en Microsoft HoloLens 2, som vil gi sluttbrukeren en MR-opplevelse og la dem samhandle med roboten og de virtuelle objektene.
4. En viktig del av programvaren vil være å sørge for at roboten reagerer på simulerte kollisjoner i MR-miljøet. Dette vil oppnås ved å utvikle algoritmer og sensorer som kan registrere kollisjoner og sende dette til roboten, som deretter vil reagere som om objektet eksisterer i virkeligheten.
5. Programvaren vil testes grundig på den virkelige roboten og Microsoft HoloLens 2 for å sikre at den fungerer som forventet og gir den ønskede MR-opplevelsen for sluttbrukeren.

Denne løsningen vil muliggjøre utvikling av en MR-programvare som kombinerer en ekte robot og virtuelle objekter i et simulert miljø. Programvaren vil være i stand til å registrere og reagere på simulerte kollisjoner.

3.2. Prosjektmetodikk

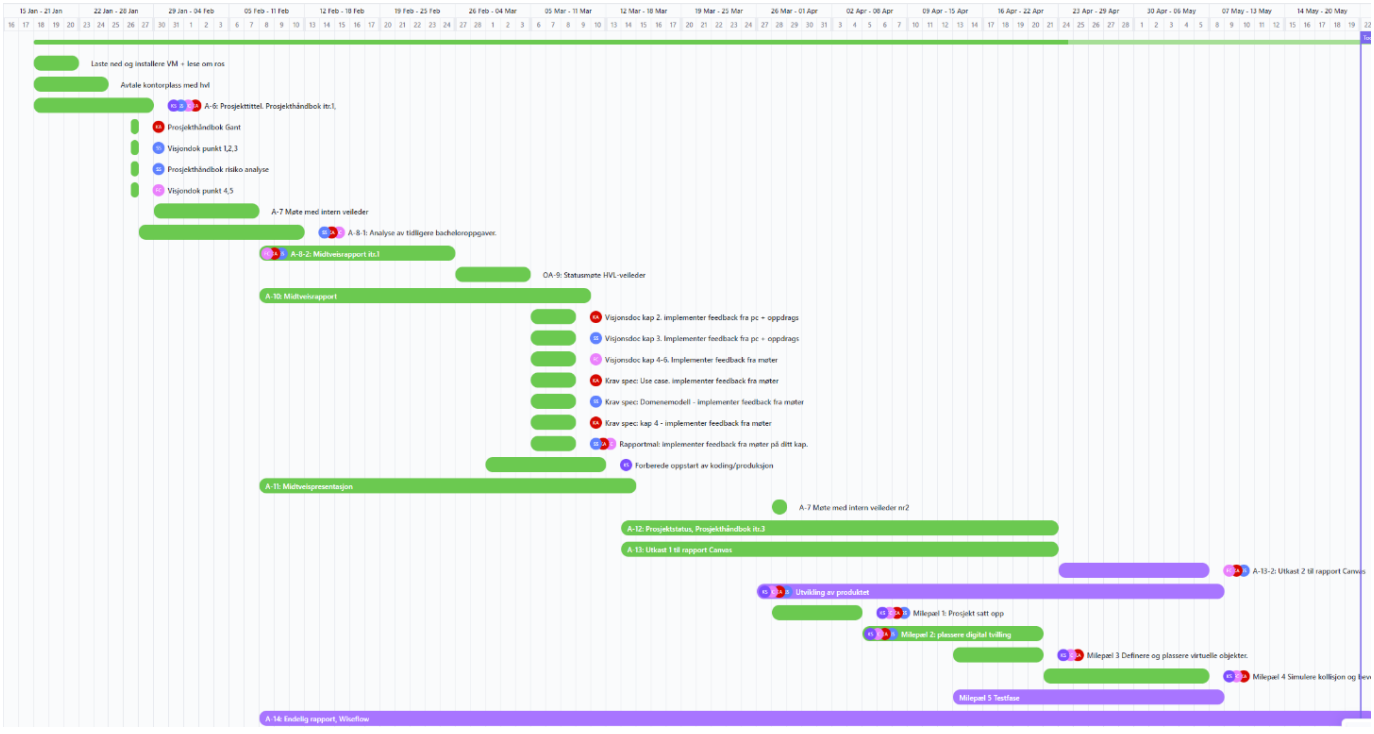
3.2.1. Utviklingsmetodikk

For å jobbe systematisk og effektivt har gruppen har valgt å anvende Scrum under utviklingsdelen av prosjektet. Scrum er en agil utviklingsmetode rettet mot systemutvikling og vil gi oss ett rammeverk for å organisere og prioritere oppgavene i prosjektet, samtidig som det sikrer en jevn flyt i utviklingen ved å dele arbeidet inn i sprinter med klare mål. Dette vil gjøre det enklere å holde oversikt over fremdriften, og sikre at vi leverer resultater i tide.

For rapportskriving og leveranser i Canvas har gruppen valgt å anvende en iterativ og inkrementell prosess som vil tillate oss å justere og tilpasse arbeidet etter hvert som vi lærer mer om prosjektet, og sikre at det leveres resultater som er tilpasset brukerens behov. Dette vil også gi mulighet til å identifisere og løse problemer tidlig i prosessen, og sikre at større feil og misforståelser unngås senere i prosjektet.

Samlet sett vil anvendelsen av disse metodene gi en fleksibel og tilpasningsdyktig metode for å håndtere kompleksiteten i prosjektet, samtidig som det sikrer en jevn fremdrift og levering av resultater til brukeren.

3.2.2. Prosjektplan



Figur 3.1: Fremdriftsplan illustrert via GANTT-diagram

Prosjektets fremdriftsplan er representert som et GANTT-diagram i figur 3.3 siden dette er et kjent verktøy for å illustrere plan for prosjekt. Prosjektet varer fra 18. januar til 22. mai, og vil omfatte flere milepæler som er avgjørende for en vellykket gjennomføring av prosjektet. Blant de viktigste milepælene finnes første iterasjon av rapport ferdig og utvikling ferdig. Til slutt vil vi levere inn den endelige rapporten den 22. mai. GANNT-planen hjelper å sikre at alle milepæler og leveranser blir oppnådd innenfor de angitte tidsrammene.

3.2.3. Risikovurdering

I dette prosjektet vil risikohåndtering bli tatt svært alvorlig. Det er blitt identifisert en rekke potensielle risikoer som kan oppstå i prosjektet, og det er viktig å redusere disse så mye som mulig. En risikomatrise benyttes for å evaluere risikoen for hvert enkelt problem, basert på sannsynlighet og alvorlighetsgrad. Kritiske funksjoner prioriteres i utviklingen, da det er viktig å sikre at de er robuste og pålitelige. Dette inkluderer testing av systemet under ulike forhold og situasjoner, for å sikre at det fungerer som forventet og at det ikke oppstår uforutsette problemer. Det vil også vurderes alternative løsninger og backup-planer i tilfelle av feil eller problemer oppstår. Ved å ta en proaktiv tilnærming til risikohåndtering, vil risikoen for problemer håndteres og sikre at prosjektet går så smidig som mulig.

	Hendelse /Risiko	***Årsak***	***Sannsynlighet***	***Konsekvens***	***Risiko-produkt***	***Tiltak***
1	*Programvare vanskelig å bruke*	*Ineffektiv og komplisert*	*Høy (4)*	*Høy (4)*	*16*	*Gjennomføre brukertesting + pri under utvikl.*
2	*Programvare fungerer ikke i henhold til mål*	*Tidsprioriterin g/ dårlig planlegging*	*Lav(1)*	*Høy(4)*	*4*	*Komme tidlig i gang med utvikling, godt kjent med hardware*
3	*Bryter tidsfrister*	*Tidsprioriterin g/ dårlig planlegging*	*Lav(2)*	*Middels (3)*	*6*	*Prioritere planlegging og bruk av gant plan*
4	*Misforstår av hvordan roboten skal lære*	*Misforstår konseptet "learning by doing"*	*Svært lav(1)*	*Svært høy (5)*	*5*	*Utstrakt bruk av veileder og produkteier, diskusjon i gruppa*
5	*Feil i kode*	*Manglende testing*	*Middels (3)*	*Svært høy (5)*	*15*	*Prioritere oppsett av git, god kommunikasjon og testing*

S a n n s y n l i g h e t	Svært Høy (5)	5	10	15	20	25	
	Høy (4)	4	8	12	16	20	
	Middels (3)	3	6	9	12	15	
	Lav (2)	2	4	6	8	10	
	Svært Lav (1)	1	2	3	4	5	
		Svært Lav (1)	Lav (2)	Middels (3)	Høy (4)	Svært Høy (5)	
	Konsekvens						

Tabell 1: Risiko vurdering

3.3. Evalueringsplan

I dette prosjektet blir det brukt en evalueringsplan som vil hjelpe gruppen med å evaluere progresjonen gjennom hver iterasjon. Det vil følges en agil og iterativ metode, som vil kreve at det kontinuerlig evalueres fremgang og gjør justeringer der det er nødvendig. Det vil bli brukt Gantt-diagrammer for å planlegge og spore hver iterasjon, og det sørges for at tidsfrister og milepæler blir holdt. Det vil også vurderes og dokumenteres resultater fra hvert trinn av prosessen, og bruke denne informasjonen til å justere fremtidige planer og prioriteringer. Det vil bli samarbeidet tett med gruppemedlemmene, veileder og det vil kommuniseres regelmessig med produkteier for å sikre at alle er på samme side og at det jobbes mot de samme målene. På denne måten vil det sikre at prosjektet går så smidig som mulig, og at produktet oppnår de ønskede resultatene.

3.3.1. Testing

Testing blir en sentral del for kvalitetssikring av oppgaven, og vi har diskutert en del metoder dette kan gjennomføres på.

Funksjonell testing: Dette er en type testing som gjøres på programvaren som helhet, og som sjekker at alle funksjoner og krav i programvaren fungerer som forventet. Funksjonell testing kan gjøres manuelt eller automatisk, og kan hjelpe til med å sikre at programvaren fungerer som den skal.

Integrasjonstesting: Dette er en type testing som sjekker at de ulike delene av programvaren fungerer sammen som forventet. Integrasjonstesting kan bidra til å identifisere eventuelle feil og mangler som oppstår når de forskjellige delene av programvaren integreres.

Brukertesting: Dette er en type testing som gjøres for å sjekke om programvaren oppfyller kravene og behovene til sluttbrukeren. Akseptansetesting kan bidra til å sikre at programvaren fungerer som ønsket og at sluttbrukeren er fornøyd med produktet.

Disse testene kan gjøres på forskjellige nivåer av utviklingsprosessen, fra tidlig utvikling til ferdig produkt.

4. DETALJERT LØSNING

4.1. Arkitektur

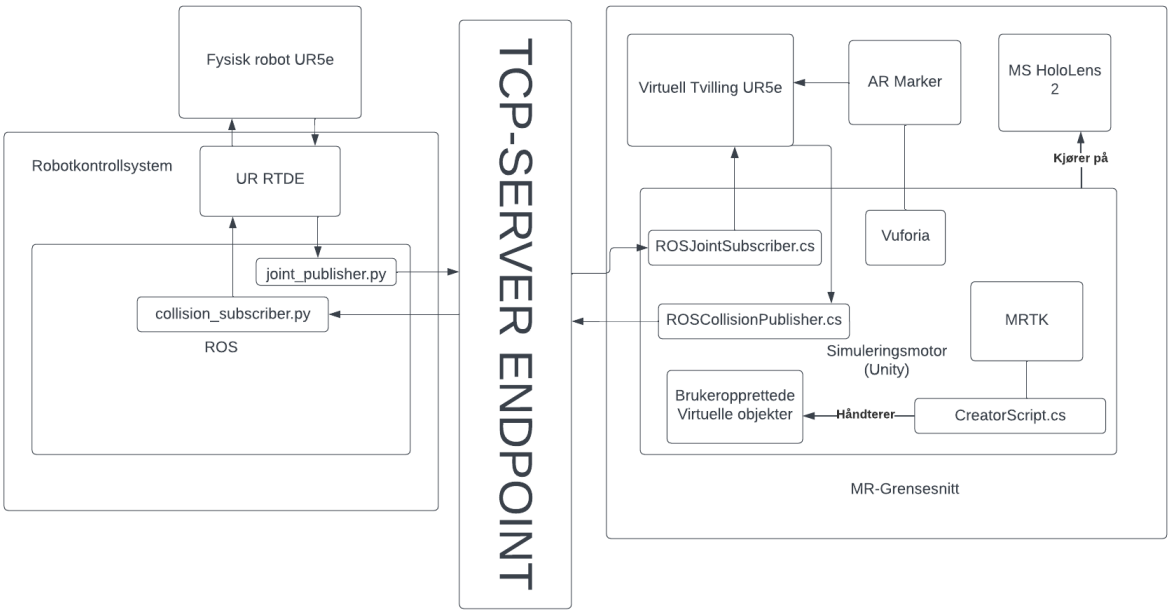
Arkitekturen til løsningen har blitt nøye utformet for å støtte de ulike brukseksemplene og for å sikre at systemet er skalerbart og vedlikeholdbart. Vi har strukturert systemet i tre hovedmoduler: MR-grensesnittet, robotkontrollsystemet og simuleringsmotoren.

MR-grensesnittet er implementert ved hjelp av Microsoft HoloLens 2 og bruker Unity som utviklingsplattform. Dette modulære systemet håndterer brukerinteraksjon og visualisering av virtuelle objekter i det virkelige miljøet. Det kommuniserer også med robotkontrollsystemet for å sende kommandoer og motta oppdateringer om robotens status.

Robotkontrollsystemet er ansvarlig for å kommunisere med den fysiske roboten og håndtere alle aspektene ved dens bevegelser. Dette inkluderer koordinering av robotens bevegelser basert på input fra MR-grensesnittet, samt håndtering av eventuelle feil eller unntak som kan oppstå under robotens drift.

Simuleringsmotoren, Unity, kjører på MS Hololens 2 og er ansvarlig for å generere og oppdatere den digitale tvillingen av roboten. Den bruker data fra MR-grensesnittet og robotkontrollsystemet for å lage en simulering av den fysiske verdenen, som gir brukerne muligheten til å interagere med roboten og miljøet på en trygg og intuitiv måte.

Hver av disse modulene består av flere mindre komponenter som samarbeider for å oppnå det overordnede målet om å gi brukerne en intuitiv og effektiv måte å interagere med roboten på.



Figur 4.1: System arkitektur

4.2. Løsningsdesign

Løsningsdesignet for prosjektet er basert på en kombinasjon av brukerinteraksjon og teknisk design, og er utformet for å oppfylle prosjektets overordnede mål. Vi har identifisert flere brukseksempler for systemet, og for hver av disse har vi utviklet et løsningsdesign som omfatter både brukerdialog og teknisk løsningsdesign. Dette inkluderer kontrollpanelet, plassering av virtuell robot, definering og plassering av virtuelle objekter, kollisjonssimulering, og valg av tidligere definerte miljøer.

I utviklingen av løsningsdesignet ble det lagt vekt på brukervennlighet og at det skal føles intuitivt for brukeren å samhandle med systemet. Vi ville sørge for at brukerne kunne interagere med systemet på en naturlig og effektiv måte, og at de kunne få mest mulig ut av funksjonene og funksjonaliteten som systemet tilbyr. Dette innebar blant annet at vi valgte teknologier og løsninger som støttet direkte manipulasjon av virtuelle objekter, og som tillot brukerne å bruke naturlige bevegelser for å anvende funksjoner i systemet.

Samtidig var det viktig å sikre at det tekniske løsningsdesignet var solid og robust. Vi ville sørge for at systemet kunne håndtere et bredt spekter av situasjoner og utfordringer, og at det kunne levere pålitelige og nøyaktige resultater uavhengig av brukerens spesifikke behov eller omstendigheter. Dette innebar blant annet at vi anvendte teknologier og løsninger som var velprøvde og pålitelige (eks. Unity og ROS), og som ga oss den fleksibiliteten og kontrollen vi trengte for å realisere prosjektets mål.

Figur 4.2: Overordnet bilde av brukeropplevelsen

4.2.1. Kontrollpanel

4.2.1.1. Brukerdialog

Kontrollpanelet er den primære brukergrensesnittkomponenten som tillater brukerne å interagere med systemet og dets funksjoner. Kontrollpanelet er designet som en flytende meny som kan følge brukerens synsfelt eller plasseres av brukeren. Det gir en intuitiv og brukervennlig opplevelse, da brukeren kan navigere og bruke funksjonene på kontrollpanelet gjennom bevegelser og håndgesturer som oppleves som naturlig.

Kontrollpanelet inneholder et utvalg av knapper, hver med sin egen funksjon. Disse inkluderer muligheter for å lage forskjellige figurer (som kuber, sfærer og kapsler), fjerne tidligere opprettede figurer, og velge mellom ulike ferdig oppsatte miljøer. Hver knapp er tydelig merket og lett å forstå, noe som gjør det enkelt for brukerne å finne og bruke funksjonene systemet tilbyr.

Figur 4.3: Implementert Kontrollpanel

4.2.1.2. Teknisk løsningsdesign

Kontrollpanelet er implementert ved hjelp av Mixed Reality Toolkit (MRTK) for HoloLens, som tilbyr en rekke ferdiglagde knapper og objekter for bruk i applikasjoner med blandet virkelighet. Vi har brukt MRTKs "Near menu", en flytemeny spesielt designet for HoloLens.

I tillegg til MRTK, har vi også brukt konseptet med prefabrikkerte former i vårt design. Dette innebærer at vi har opprettet en serie forhåndslagde former med bestemte komponenter og størrelser. Disse formene kan deretter brukes av systemet til å generere identiske kopier av den prefabrikkerte formen etter behov. For dette prosjektet har vi en kube, sfære, kapsel og en sylinder. Det er enkelt å legge inn mer avanserte prefabrikkerte former i framtiden.

Det er viktig å merke seg at selv om formene opprettes med en forhåndsdefinert størrelse, kan de gjøres mindre eller større etter ønske gjennom HoloLens. Dette gir brukerne fleksibilitet til å tilpasse størrelsen på de virtuelle objektene i henhold til deres spesifikke behov.

Funksjonaliteten bak de forskjellige knappene og skyveknappene implementeres av forskjellige scriptkomponenter:

- Button Methods: implementerer enkle funksjonaliteter for knapper som åpning av meny og låsing av miljøet.
- Offset Adjuster: håndterer funksjonaliteten til offset skyveknappen og endrer objekter basert på skyveknappens verdi.
- Preset Opener: åpner forhåndsinnstilte miljø, og passer på at bare et miljø er åpent om gangen.
- Restore Default Slider Value: Setter skyveknapper tilbake til sine originale verdier.
- Slider Link: kobler sammen skyveknapper, slik at alle alltid har samme verdi.

```
public class ButtonMethods : MonoBehaviour
{
    public void OpenMenu(GameObject menu)
    {
        menu.SetActive(!menu.activeSelf);
    }

    public void LockEnvironment(GameObject imageTarget)
    {
        ImageTargetBehaviour script = imageTarget.GetComponent<ImageTargetBehaviour>();
        script.enabled = !script.enabled;
    }
}
```

Figur 4.4: Button Methods komponenten.

4.2.2. Brukseksempel: Plassere virtuell robot

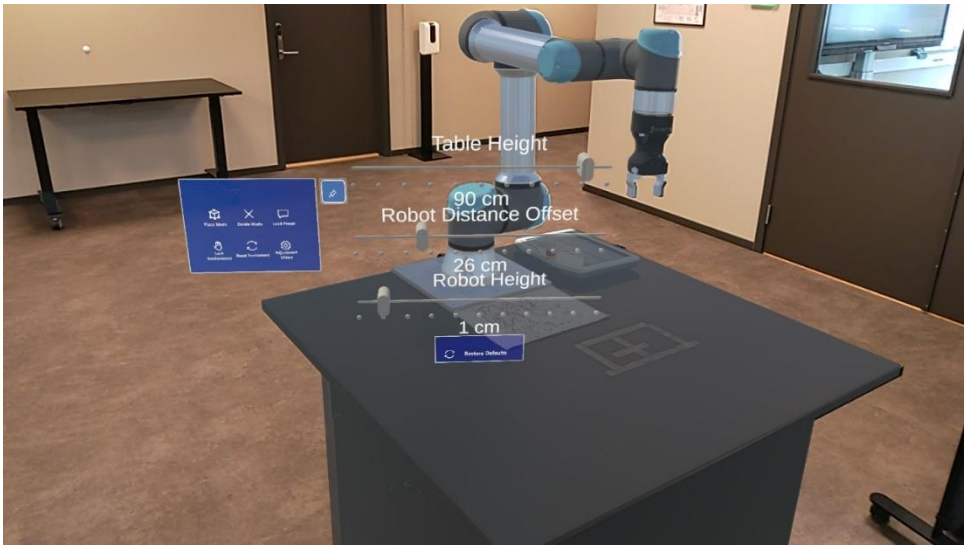
4.2.2.1. Brukerdialog

En sentral funksjon i løsningen er evnen til å plassere den virtuelle roboten i det digitale miljøet på en måte som gjenspeiler den fysiske robotens plassering i det virkelige miljøet. Dette gjøres ved hjelp av en AR-markør. Ved å plassere AR-markøren nær der den faktiske roboten befinner seg, kan brukeren posisjonere den virtuelle roboten nøyaktig i det digitale miljøet ved hjelp av skyveknappene. Dette er avgjørende for å sikre at simuleringene og manipulasjonene som utføres på den virtuelle roboten korrekt reflekterer hva som vil skje i det virkelige miljøet.

Figur 4.5: Miljøet før AR-markør er riktig plassert og kalibrert

Figur 4.6: Virtuell tvilling plassert på fysisk robot ved hjelp av AR-markør

Dersom applikasjonen skal brukes på en robot som er satt opp på en annen måte, kan verdiene av hvor roboten står i forhold til AR-markøren justeres i kontrollpanelet ved hjelp av justerings-skyveknappene.

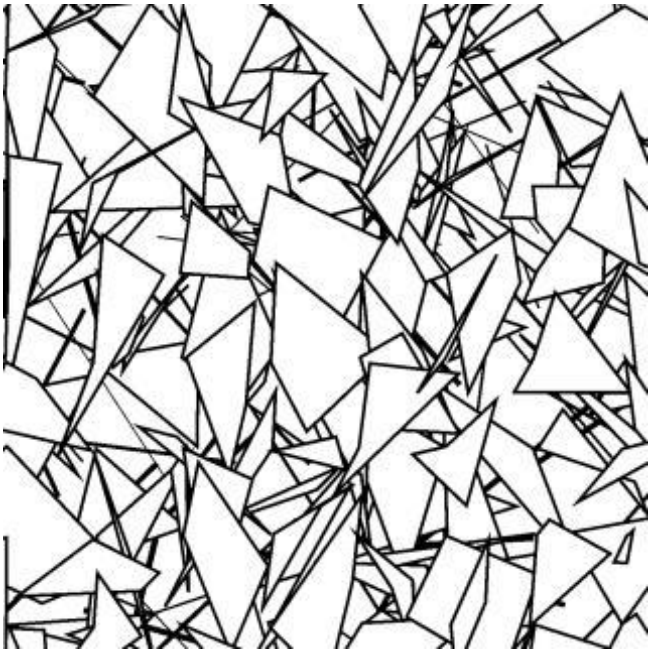


Figur 4.7: Justerings skyveknapper for posisjon av robot i forhold til AR-markør

4.2.2.2. Teknisk løsningsdesign

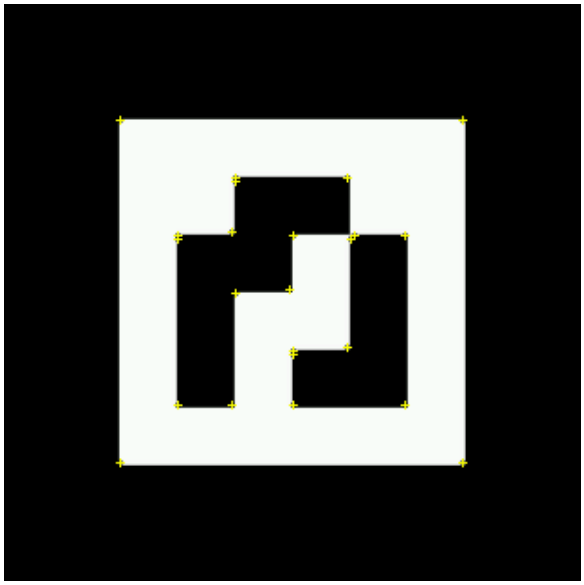
Programvaren bruker Vuforia som er et populært AR-rammeverk for å implementere AR-markørfunksjonalitet. Når en bruker peker HoloLens-kameraet mot AR-markøren, oppdager Vuforia markøren og returnerer objektets posisjon og orientering i det virkelige miljøet.

Det er importert en digital tvilling av den fysiske roboten i Unity som kan utplasseres i det virtuelle miljøet. Når AR-markøren oppdages, plasseres den digitale tvillingen på samme sted og med samme orientering i det digitale miljøet, som det den fysiske roboten har i den virkelige verden. Dette sikrer at den virtuelle robotens posisjon og orientering nøyaktig speiler den fysiske robotens posisjon og orientering i det virkelige miljøet.

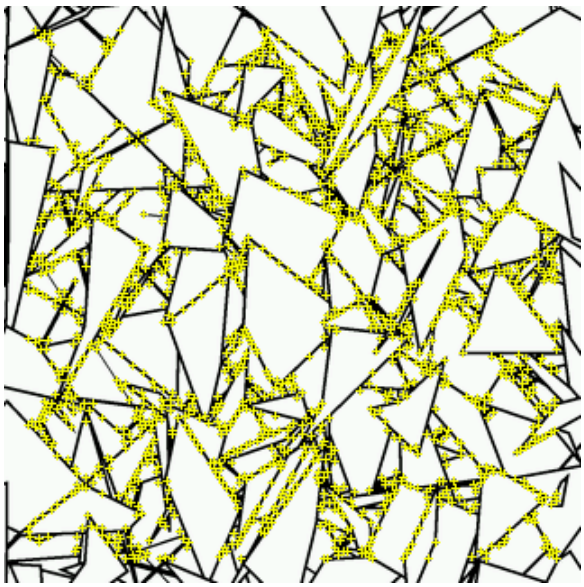


Figur 4.8: AR-markør

Vuforia brukes til å oppdage AR-markører ved å se etter spesifikke punkter på et bilde som kalles "features". I utgangspunktet var planen å bruke en enklere AR-markør (figur 4.6). Dette fungerte ikke godt nok, fordi den hadde få punkter som Vuforia kunne gjenkjenne bildet på. Derfor ble en mer detaljert AR-markør valgt (figur 4.5), fordi den hadde flere gjenkjennbare punkter og god kontrast, som dermed gjorde det enklere for Vuforia å gjenkjenne markøren. Under er "features" på de to markørene visualisert som gule punkter.



Figur 4.9: "Features" i dårlig AR-markør



Figur 4.10: "Features" i god AR-markør

Funksjonaliteten i brukseksempelet er stort sett håndtert av Vuforia sine komponenter, men disse komponentene fungerer ikke med Articulation Body komponenter, som den virtuelle tvillingen bruker. Av denne grunn måtte Robot Teleporter komponenten lages, som flytter roboten til AR-markøren sin posisjon så lenge AR-markøren er observert.

```
public void TeleportRobot(GameObject gObject)
{
    baseObject.GetComponent<ArticulationBody>()
        .TeleportRoot(gObject.transform.position, gObject.transform.rotation);
}
```
Figur 4.11: Metoden som flytter den digitale tvillingen.

4.2.3. Brukseksempel: Flytte på fysisk og virtuell robot

4.2.3.1. Brukerdialog

For at systemet skal fungere er en digital tvilling en viktig del. Den digitale tvillingen skal alltid følge bevegelsene til den ekte roboten. Fra brukerens side gjøres dette enkelt ved å bevege den fysiske roboten. Brukeren vil da se den virtuelle roboten følge etter i sanntid.

4.2.3.2. Teknisk løsningsdesign

For at den virtuelle roboten kan gjenspeile den ekte roboten må den oppføre seg som en tvilling av den ekte roboten. Under forrige brukseksempel ble det forklart hvordan den virtuelle roboten ble plassert i miljøet, men ikke hvordan leddene til den virtuelle roboten allerede var i riktig posisjon. For å oppnå denne funksjonaliteten brukes scriptene "Joint Publisher" på ROS-siden, og "ROS Joint Subscriber" på Unity-siden til å kontinuerlig sende data om posisjon fra hvert av den fysiske robotens ledd, til den digitale tvillingen. Dette gjør at den virtuelle roboten alltid vil finne posisjonsdataen fra den ekte roboten og oppdatere i sanntid, selv om den ekte roboten rører på seg.

```
void JointUpdate(JointStateMsg jointData)
{
    if (scriptEnabled && !manualJointValues)
    {
        List<float> positions = new List<float>();
        for (int i = 0; i < jointData.position.Length; i++)
        {
            positions.Add((float)jointData.position[i]);
        }

        baseObject.GetComponent<ArticulationBody>().SetDriveTargets(positions);
    }
}
```
Figur 4.12: Den digitale tvillingen beveger seg hver gang den mottar ny ledd-informasjon.

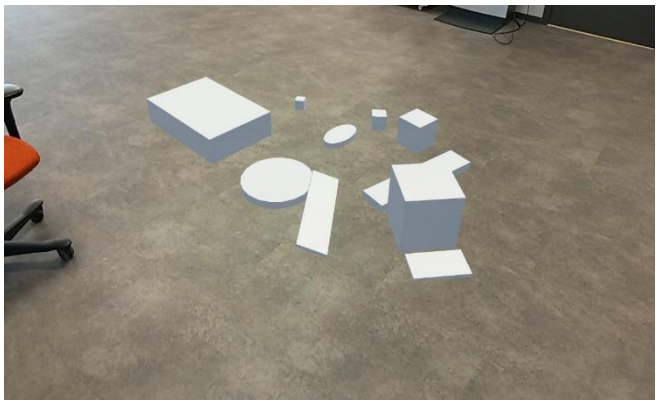
4.2.4. Brukseksempel: Definere og plassere virtuelle objekt

4.2.4.1. Brukerdialog

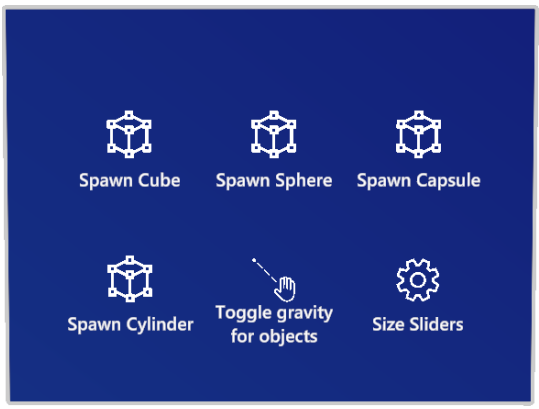
Programvaren er designet for en interaktiv brukeropplevelse når det kommer til plassering og definering av et virtuelt objekt i miljøet. Prosessen starter med at brukeren velger et objekt fra menyen på kontrollpanelet. Så snart et objekt er valgt, opprettes en virtuell representasjon av det valgte objektet, synlig for brukeren gjennom HoloLens.

Brukeren har deretter mulighet til å justere objektets plassering i det virtuelle miljøet ved hjelp av håndbevegelser. I tillegg tillater systemet brukeren å rotere objektet for å oppnå ønsket posisjon i rommet. Objektets størrelse kan også justeres, enten dynamisk med brukerens håndbevegelser eller forhåndsbestemt i kontrollpanelet for økt presisjon og kontroll.

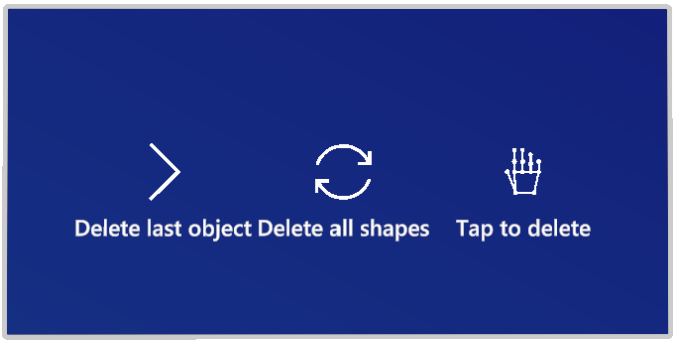
Denne interaktive prosessen gir brukeren mulighet til å manipulere og definere virtuelle objekter med høy grad av presisjon og kontroll. Dette bidrar til å skape en nøyaktig gjengivelse av det fysiske miljøet i det virtuelle rommet.



Figur 4.13: Utplasserte objekter i miljøet



Figur 4.14: Plasserings-undermeny



Figur 4.15: Slette-undermeny

4.2.4.2. Teknisk løsningsdesign

Når det kommer til definering og plassering av virtuelle objekter, involveres en blanding av brukerinteraksjon og programvarelogikk. Ved valg av objekt fra menyen i brukergrensesnittet, initieres opprettelsen av det virtuelle objektet. Objektene er forhåndsdefinerte som 3D-modeller og genereres ca. 20 cm foran der brukeren ser.

For å gi brukeren en mest mulig virkelighetsnær opplevelse, kan brukeren endre plasseringen og størrelsen på det virtuelle objektet direkte gjennom HoloLens 2 sitt brukergrensesnitt. HoloLens 2 sine sensorer oppfatter brukerens håndbevegelser og sender denne informasjonen til applikasjonen, som deretter oppdaterer objektets plassering og størrelse i det virtuelle miljøet.

Med denne implementeringen kan brukeren aktivt forme sin virtuelle verden, noe som gir en realistisk og nøyaktig representasjon av det fysiske miljøet i det virtuelle rommet. Dette gir muligheter for økt forståelse og interaksjon med både det virtuelle og det fysiske miljøet.

For å oppnå dette brukseksempelet ble Creator Script komponenten opprettet, og er ansvarlig for å håndtere opprettelse og oppdatering av virtuelle objekter opprettet av brukeren.

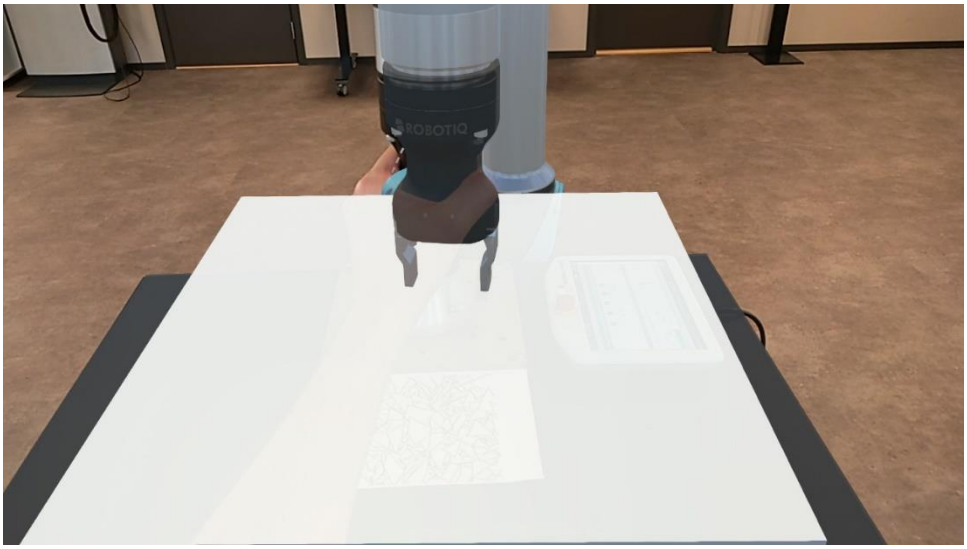
```
// Function for spawning the selected prefab
public void SpawnObject(GameObject prefab)
{
    prefab.transform.localScale = new Vector3(widthScale, heightScale, lengthScale);
    // Sets spawnpostion at end of linerenderer
    Vector3 spawnPosition = Camera.main.transform.position + Camera.main.transform.forward * 2f;
    // Should add scaling based on current slider values here.
    GameObject newObj = Instantiate(prefab, spawnPosition, Quaternion.identity);

    // Add an Interactable component and set its OnClick event to OnObjectClicked
    Interactable interactable = newObj.AddComponent<Interactable>();
    interactable.OnClick.AddListener(() =>
    {
        if (deleteMode)
        {
            Destroy(newObj);
            prefabList.Remove(newObj);
        }
    });
    // Checks if gravity should be toggeled on or off
    CheckGrav(newObj);
    prefabList.Add(newObj);


}
```

Figur 4.16: Et objekt valgt av brukeren blir opprettet i det virtuelle miljøet.

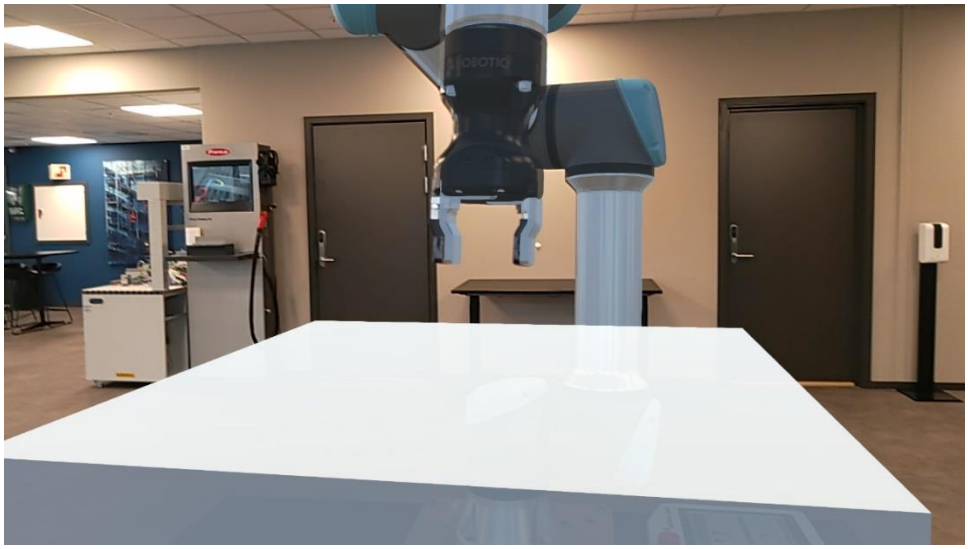
4.2.5. Brukseksempel: Simulere kollisjon

4.2.5.1. Brukerdialog

En annen kritisk funksjon som er implementert er kollisjonssimulering mellom roboten og de virtuelle objektene i miljøet. Dette gir brukeren mulighet til å teste robotens veibeskrivelse og manipulasjonsoperasjoner i et trygt, virtuelt miljø før de implementeres i et miljø med virkelige hindringer. Brukeren kan observere simuleringen i sanntid, og eventuelle kollisjoner vil bli tydelig indikert også på den virkelige roboten. Dersom brukeren beveger på armen fysisk, så vil brukeren kjenne at robotarmen gjør motstand når den treffer et virtuelt objekt.



Figur 4.17: Roboten treffer et virtuelt objekt



Figur 4.18: Roboten rekylerer retningsbestemt og viser klar reaksjon på det virtuelle objektet.

4.2.5.2. Teknisk løsningsdesign

Roboten og de virtuelle objektene er utstyrt med Unity sine innebygde komponenter for fysikk- og kollisjonshåndtering - Rigid Body og forskjellige Collider-komponenter. Disse komponentene tillater simulering av realistisk interaksjon mellom den virtuelle roboten og objektene i miljøet.

Når en simulert kollisjon skjer mellom den virtuelle roboten og et virtuelt objekt, vil retningen av kollisjonen kommuniseres til roboten. Denne interaksjonen er muliggjort gjennom robotens "force mode" funksjon, som brukes til å få roboten til å gi en motkraft i motsatt retning av kollisjonen.

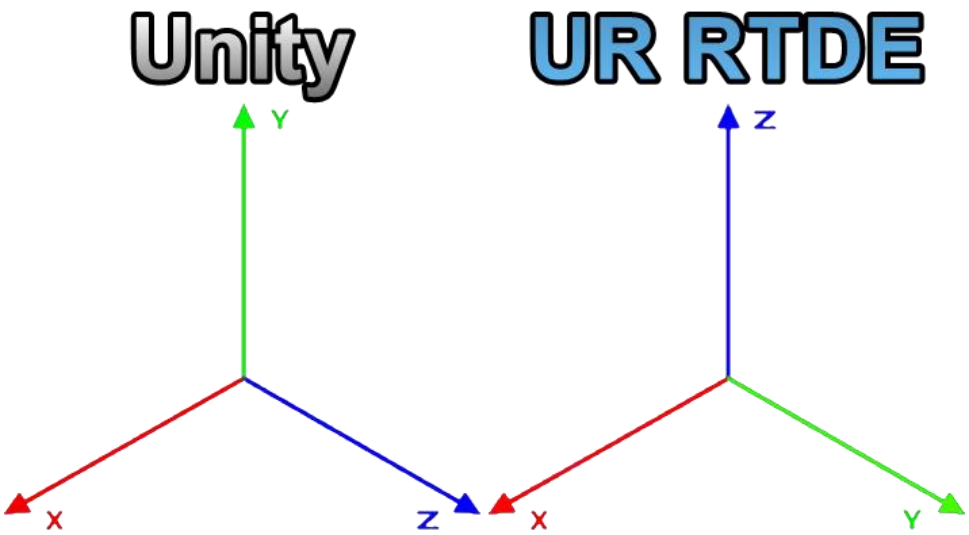
I dette brukseksempel er disse scriptene sentrale:

- Collision Detector: En komponent som kaller på Events når kollisjoner starter og slutter.
- ROS Collision Publisher: En komponent som publiserer normalvektor av kollisjonen.
- Collision Subscriber: Et script i ROS som mottar meldinger om kollisjoner og bruker normalvektor til å få roboten til å gi motkraft i motsatt retning.

Siden Unity har et annerledes koordinatsystem, måtte *z* og *y* byttes ut i koden der man setter kraftvektorene.

```
if not self.force_mode_enabled:
    # Sets wrench values
    self.wrench[0] = self.force * msg.x
    self.wrench[1] = self.force * msg.z
    self.wrench[2] = self.force * msg.y
    # Enables force mode
    self.force_mode_enabled = True

    self.rtde_c.forceMode(self.task_frame, self.selection_vector, self.wrench, self.force_type, self.limits)
    self.rtde_c.forceModeSetDamping(self.damping)
```

Figur 4.19: Force Mode skrus på etter kollisjon

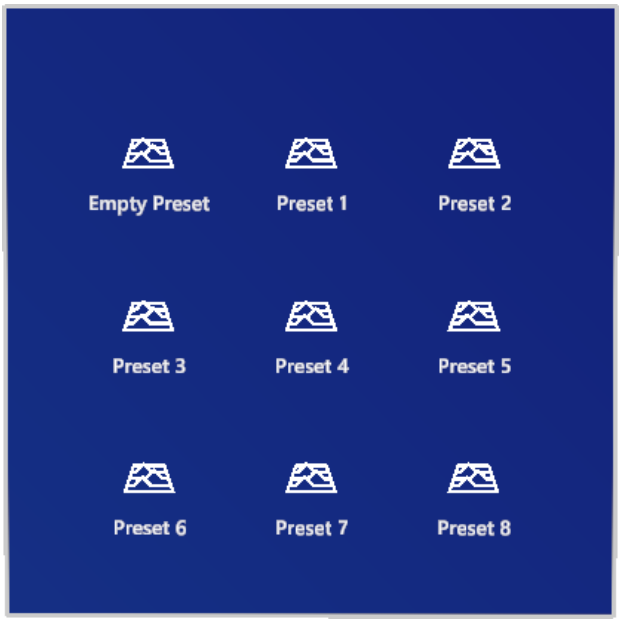


Figur 4.20: Forskjell i koordinatsystem mellom Unity og UR RTDE

4.2.6. Brukseksempel: Velge tidligere definert miljø

4.2.6.1. Brukerdialog

Et sentralt brukstilfelle i applikasjonen er evnen til å bygge miljøer i Unity Editor for å bruke i applikasjonen. Dette er en viktig funksjon for brukere som trenger å jobbe med nøyaktige og avanserte oppsett over flere økter, da det er vanskelig å bygge disse i det virtuelle miljøet hver gang. For å kunne bygge miljøer, må man legge inn objektene i miljøet i en av preset objektene under miljøet i ImageTarget objektet, slik at miljøet plasserer seg i forhold til AR markøren.



Figur 4.21: Preset-undermeny


Figur 4.22: Miljøet etter et preset er valgt

4.2.6.2. Teknisk løsningsdesign

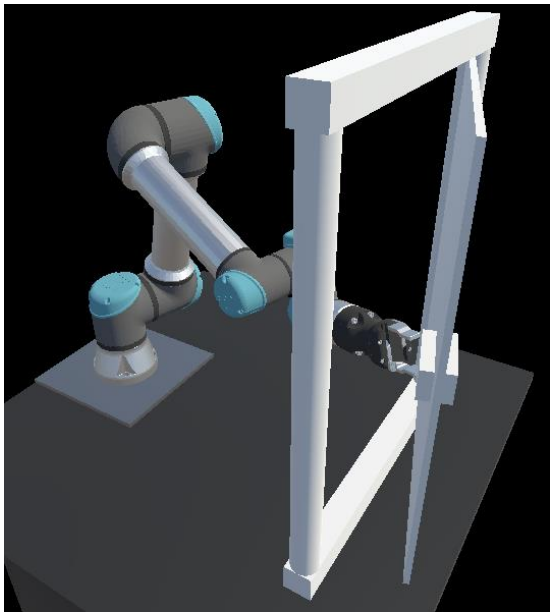
For å implementere denne funksjonen, har vi laget tomme objekter som heter “Preset 1” opp til “Preset 8”. Koden er skrevet på en måte som gjør det enklest mulig å utvide listen med flere forhåndsdefinerte miljø (presets) dersom det skulle være nødvendig. Når en bruker ønsker å lage et nytt miljø, oppretter brukeren de objektene som trengs inni et av Preset-objektene, og disse vil vises i applikasjonen når brukeren velger det aktuelle miljøet. Alt dette er lagt under ImageTarget sammen med resten av miljøet slik at den plasserer seg i forhold til AR markøren. Siden målgruppen er avanserte brukere, er dette en god løsning som gir brukeren bedre muligheter til å lage avanserte miljøer.

Som et eksempel, har vi bygget et preset med en dør som kan åpnes av robotarmen, ved hjelp av Hinge Joint komponenten. Når robotarmen dyttes inn i døren, vil brukeren kjenne motkraft, men kan fortsette å dytte for å åpne døren helt. Ved hjelp av andre Unity komponenter, kan mange andre scenarioer bygges inn, som ville vært veldig vanskelig eller umulig å bygge inni applikasjonen.

For å velge en preset, bruker Preset Opener komponenten, som passer på at bare ett preset er åpent om gangen.

```
public class PresetOpener : MonoBehaviour
{
    public GameObject[] presets;

    public void TogglePreset(int number)
    {
        for (int i = 0; i < presets.Length; i++)
        {
            presets[i].SetActive(false);
        }
        presets[number].SetActive(true);
    }
}
```

Figur 4.23: Koden som håndterer preset-åpning og lukking.



Figur 4.24: Roboten interakterer med dør-preset

5. RESULTATER

I starten av prosjektet forventet man minst et program system som kunne kommunisere mellom ROS og robot, Unity til ROS og Unity til HoloLens 2. Minstekravet for kollisjon med virtuelt objekt var at roboten stoppet når den detekterte kollisjon. Dette målet har blitt nådd og passert.

5.1. Prosjektresultat

5.1.1. Funksjonelle og ikke funksjonelle krav

Egenskap	**Beskrivelse**	**Delkrav**			
Virtuell hinder	Opprettelse og konfigurering av ett spillobjekt som roboten kan reagere på.	Nr.	Beskrivelse	Planlagt	Fullført
		1.1	Oppretting av spillobjekt.	Ja	Ja
		1.2	Robot reagerer på spillobjekt.	Ja	Ja
		1.3	Konfigurering av opprettede spillobjekter.	Ja	Ja
Legge til ny robot	Funksjonalitet for å legge til og konfigurere ny robot.	2.1	Funksjonalitet for å legge til og konfigurere nye roboter.	Ja	Nei
Valg av konfigurert robot	Funksjonalitet for å velge en av flere roboter som er tidligere konfigurert.	3.1	Funksjonalitet for å velge en av flere roboter som er tidligere konfigurert.	Ja	Nei
Reset	Funksjonalitet for å stille robot tilbake til standard posisjon og fjerning av plasserte objekter.	4.1	Funksjonalitet for å stille robot tilbake til standard posisjon.	Ja	Nei
		4.2	Funksjonalitet for å fjerne plasserte spillobjekter.	Ja	Ja

Tabell 2: Implementasjon av funksjonelle krav

Ikke-funksjonell egenskap	Beskrivelse	Delkrav			
ROS-Unity kommunikasjon	Kommunikasjon mellom ROS og Unity.	Nr.	Beskrivelse	Planlagt	Fullført
		1.1	Import av robot modell til Unity.	Ja	Ja
		1.2	TCP-Connector for å sende og motta robotposisjon og kommandoer for å styre roboten.	Ja	Ja
		1.3	ROS-TCP-Endpoint for å kommunisere meldinger fra TCP-Connector fra ROS til Unity.	Ja	Ja
Testing	Prosjektet blir testet og utviklet på en UR5E robot arm og Microsoft HoloLens 2.	2.1	Teste programvaren på HoloLens 2.	Ja	Ja
		2.2	Teste programvaren på en UR5E robot arm.	Ja	Ja
Unity-HoloLens 2 kommunikasjon	Kommunikasjon mellom Unity og HoloLens 2 eller tilsvarende AR-enhet.	3	Kommunikasjon mellom Unity og HoloLens 2.	Ja	Ja
Force feedback	Robot reagerer på virtuelle spillobjekt i banen med å bruke mer styrke på å bryte gjennom.	4.1	Robot kan rekylere ved kollisjon med spillobjekter.	Ja	Ja
		4.2	Robot kan bryte gjennom spillobjekter med nok kraft.	Ja	Ja

Tabell 3: Implementering av ikke funksjonelle krav

5.2. Evalueringsmetode

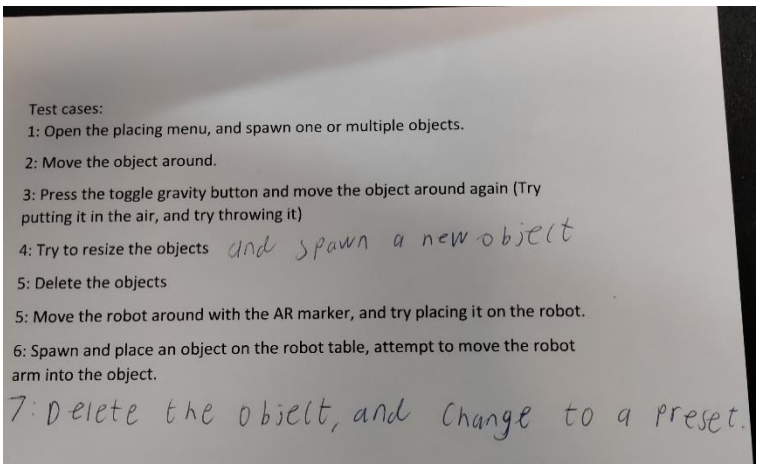
5.2.1. Funksjonell testing

Gjennom hele utviklings fasen har det blitt utført funksjonell testing. Den funksjonelle testingen ble utført stegvis under utvikling av hver komponent. Testingen ble utført med henhold til krav fra HVL Robotics. Kravene ble satt tidlig i utviklingen i form av brukstilfeller. Først ble det testet om komponenten kjørte og fungerte. I neste steg vart det sjekket om komponenten fungerte som ment og om den trengte justeringer eller større endringer. I disse to første stegene vart det testet lokalt på PC i Unity, videre i neste steg vart komponenten testet med HoloLens 2. Etter hvert brukstilfelle ble implementert ble disse testet sammen med HVL Robotics for å sikre at alt er i henhold til deres krav og spesifikasjoner.

5.2.2. Brukertest

Det ble utført en rekke brukertester med brukere fra ulike bakgrunner. Disse inkluderer HVL Robotics samt ansatte ved Verftet AS. Testingen ble utført ved at programsystemet ble satt opp og startet. Brukerene kunne da ta på Hololens 2 og ble bedt om å utføre en rekke testings instrukser.

Testingen vil vise en rekke interessante egenskaper for systemet. Noen av egenskapene er bruker vennlighet for en typisk endebruker, funksjonalitet for en typisk endebruker og den generelle erfaringen med system. Dette gir innsikt i hvordan det er å operere programvaren fra ett utestående perspektiv.



Figur 5.1: Brukertest instrukser

5.2.3. Integrasjonstest

Denne typen tester er kontinuerlig blitt gjennomført på modulnivå, gjennom hele utviklingsprosessen. En stor del av problemområdet prosjektet berører handler om å få de ulike komponentene til å kommunisere på riktig måte. Et enkelt eksempel på dette kan være å lese av input ROS får fra Unity, for å verifisere at den kommer i riktig format.

5.3. Evalueringsresultat

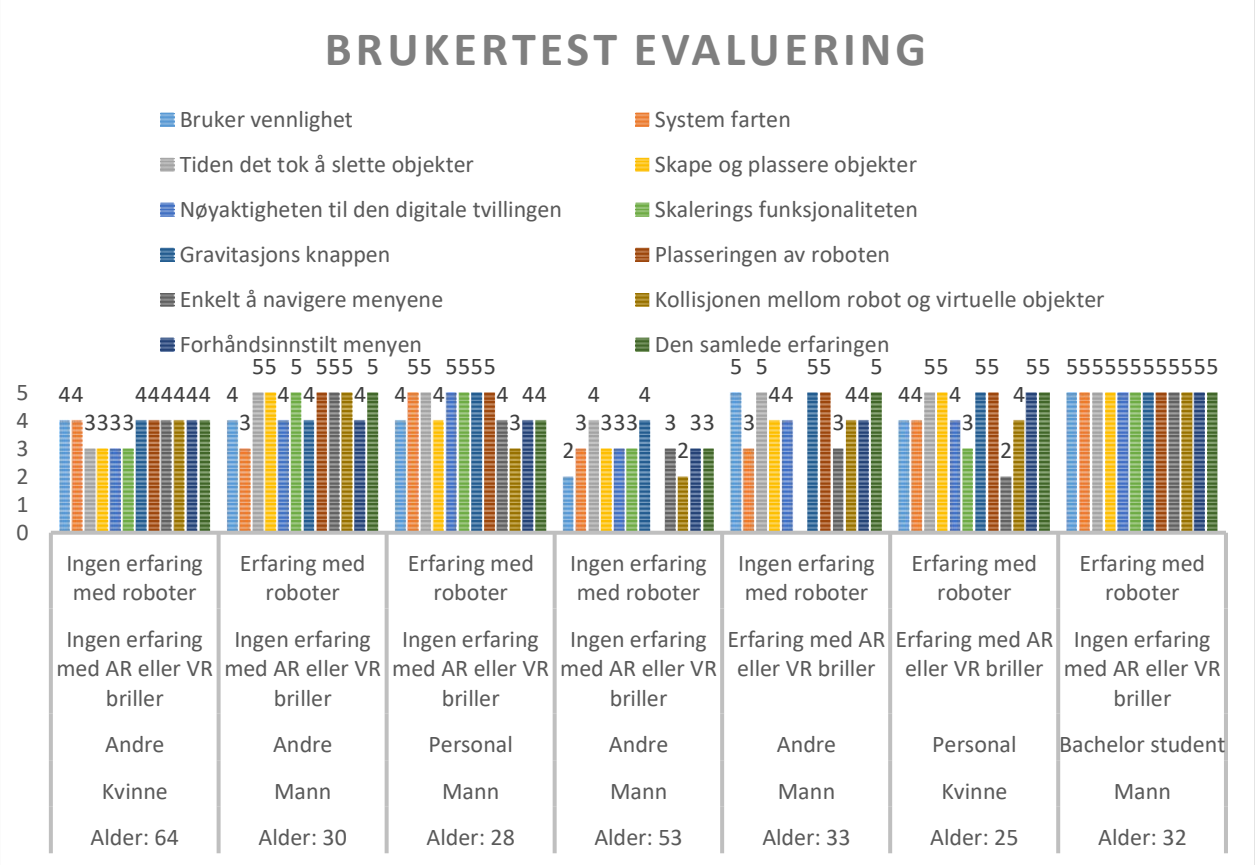
5.3.1. Funksjonell testing

Den funksjonelle testingen gjorde at komponenter var fullt funksjonelle før de ble sendt til GitHub. Dette skjermet programmet fra problemer med at GitHub koden ikke ville kjøre og dermed slapp problemer for andre som jobbet på samme kode. Den funksjonelle testingen førte også til endringer i tiltenkt løsning, da man fant at løsninger kunne simplifiseres når original ide viste seg lite bruker vennlig.

5.3.2. Brukertest

Figur 5.2 viser en del informasjon om systemet basert på tilbakemelding fra brukere etter en gjennomført brukertest. Figuren viser hver bruker sin evaluering av de forskjellige punktene med

en del kontekst om hvilken bakgrunn brukeren har. Basert på brukertest evalueringen kan det trekkes frem at bruker vennlighet kan forbedres da den på det laveste fikk en vurdering på 2 av 5, dette kan ha en sammenheng med meny navigering da denne også fikk en del under middels verdier. Testingen viste også at HoloLens 2s batteri tid ikke er veldig lang, da batteriet gikk tomt under brukertesting. Det førte til at resten av testene måtte gjennomføres med ladekabel koblet til. Testingen viste også at det kreves en tilvenningstid ved første bruk av HoloLens 2 som gjorde at testing for hver bruker tok lengre tid enn antatt.



Figur 5.2: Brukertest evaluering

5.3.3. Integrasjonstest

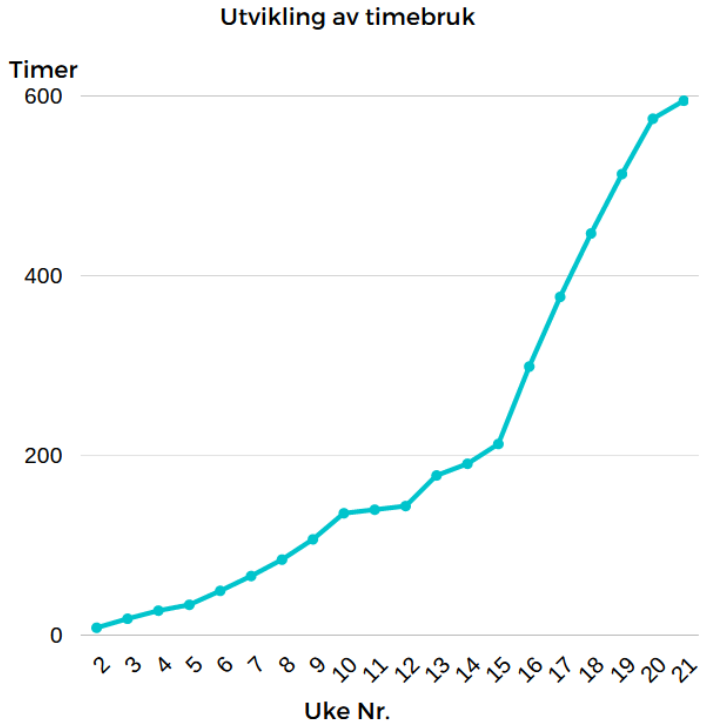
Disse testene ble kontinuerlig gjennomført gjennom utviklingsprosessen, hovedsakelig ved å overvåke ulike inputs, outputs og resultat. Negative testresultat har ved flere anledninger avdekket feil og mangler, men samtlige har blitt rettet og løst i ettertid.

5.4. Prosjektgjennomføring

Prosjektet har i god grad klart å følge fremdriftsplanen som ble framstilt i figur 3.3 og som er tilgjengelig i sin helhet i rapportens støttedokument. Her er det et spesielt unntak hvor utviklingen overskredet planlagt tidsbruk. Grunnen til dette var uforutsette problemer ved utviklingen som tok lengre tid enn forventet. Dette gjorde at gruppen var etter skjema i en mindre periode, noe som ble hentet inn raskt etter problemet ble løst og gruppen kom seg i rute igjen.

Generell tidsbruk lå i starten av prosjektet under forventet tidsbruk for hver periode. Dette ble identifisert og det ble satt i gang tiltak for å få fulgt framdriftsplanen. Dette gjorde at timebruken

økte og ble jevn mellom alle medlemmene gjennom utviklingsfasen. Tiltaket var at alle medlemmene møttes ved faste tider hver dag under utviklingen. Alle planlagte oppgaver og aktiviteter ble gjennomført i løpet av prosjektet og alle leveranser ble oppfylt.



Figur 5.3: Utvikling av total timebruk gjennom prosjektet

6. DISKUSJON

I dette kapittelet vil det diskuteres årsakene til avgjørelsene som ble gjort angående prosjektet, og utfordringer som oppstod med utviklingen. Hvorfor disse avgjørelsene ble gjort og om de hadde en positiv eller negativ effekt på prosjektet.

6.1. Læring ved demonstrasjon

Å designe programsystemet rundt fysisk manipulering av roboten ble vurdert som mest hensiktsmessig for prosjektet. Det var avgjørende å gjøre brukeropplevelsen tilgjengelig og intuitiv, og dette ble oppnådd ved å tillate direkte, fysisk interaksjon med roboten. Med den løsningen ble behovet for teknisk kompetanse hos brukeren redusert, systemet fikk en mer brukervennlig opplevelse- en nøkkel til å sikre en sømløs kollisjon mellom den fysiske roboten og de virtuelle objektene.

En alternativ løsningsmetode kunne vært å anvende læring ved demonstrasjon, som ville introdusert mer avanserte funksjoner og automatisering i systemet. I den opprinnelige oppgavebeskrivelsen var det faktisk et ønske fra oppdragsgivere om å inkludere funksjonalitet for dette. Imidlertid ville en slik implementasjon krevd betydelig mer tid og teknisk innsikt, noe som strider mot prosjektets ressurs- og tidsbegrensninger. Derfor så både oppdragsgiver og

gruppen at det var mer hensiktsmessig å konsentrere seg om direkte manipulasjon av roboten. Dette var klart i god tid før planleggingen av utviklingen.

Valget av denne metoden ga også fordeler når det kom til testing og iterasjon av systemet i det blandede virkelighetsmiljøet. Med direkte manipulasjon av roboten kunne systemet testes og justeres raskt, basert på umiddelbar feedback fra deltagere i brukertestene.

6.2. ROS miljø

Tidlig i prosjektet ble det bestemt at en virtuell maskin som ble brukt i sammenheng med et robotikk fag på HVL skulle brukes. Denne maskinen kjørte ROS versjon Melodic og operativ systemet Ubuntu versjon 18.04. I begynnelsen av utviklingsfasen oppdaget vi at denne versjonen av ROS Melodic ikke støttet versjon 3 av programmeringsspråket Python, som var nødvendig for UR RTDE pakken. Det ble derfor satt opp ny virtuell maskin med ROS Noetic som støttet Python 3 og Ubuntu versjon 20.04.

6.3. Unity

Valg av Unity versjon ble diskutert med oppdragsgiver og det ble anbefalt seneste versjonen av Unity da det ikke var noen kompatibilitets problemer som trengtes å bli tatt hensyn til. Det ble da valgt den siste LTS versjonen 2021.3.21f1 når utviklingen startet grunnet tilgang til de nyeste funksjonene og tjenestene.

6.3.1. MRTK

Når det kom til valg av MRTK versjon stod valget mellom versjon 3 og 1. MRTK 1 er den første versjonen som ble utviklet, og denne versjonen ble valgt bort da man ønsket nyest mulig versjon. MRTK2 er en nyere versjon som ble anbefalt av oppdragsgiver da den er veldokumentert og relativt ny. MRTK3 er den nyeste versjonen, men er mindre dokumentert og ble derfor valgt bort i favør for MRTK2.

6.3.2. Vuforia

Plassering av roboten var planlagt med en ide om at det ville ligne på metoden brukt i baneplanlegging prosjektet (Benjaminsen & Dankel, 2022). Denne metoden inkluderte en QR-kode som ble plassert ved basen til roboten. En metode for å gjøre dette blir beskrevet i en blogg post av Joost van Schaik (Schaik, 2021). Denne metoden beskriver måten å plassere objekter basert på en QR kode som Vuforia. Etter vurdering av denne metoden ble det vurdert at Vuforia med AR markører passet bra for prosjektet da det er gratis for ikke-kommersielle prosjekter.

6.4. HoloLens 2

Ved første forsøk på å rulle ut programvaren til HoloLens2 oppstod det problemer under bygging av prosjektet, som siden ble løst sammen med oppdragsgiver. Ellers ble det erfart at batteritiden er relativt kort på MR-brillene. Denne batteri tiden førte til forsinkelser ved både den funksjonelle testingen og brukertestene. Det ble bestemt at en eller flere strømbanker burde bli tatt med på demonstrasjoner og liknende.

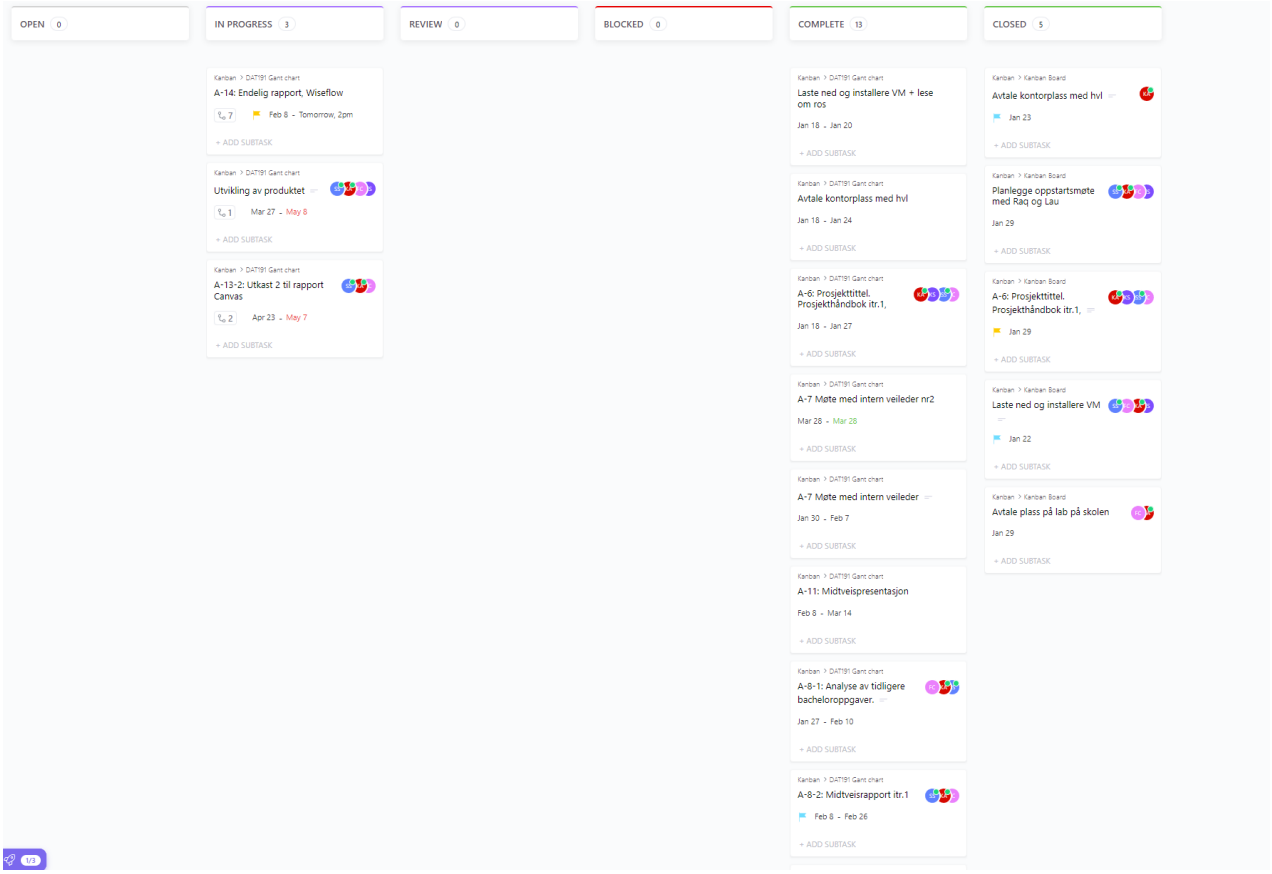
6.5. Brukertest

Resultatene fra denne forteller oss at enkelte testdeltagere hadde litt å utsette på aspekter som brukervennlighet og tidsbruk. For dette prosjektet tolkes dog testresultatene som positive, da alle deltagere oppnådde suksess med alle brukseksempeler. I en forlenging eller utvidelse av prosjektet ville det vært naturlig å jobbe videre med aspektene der enkelte testere ikke ga full score. Snitt på 4/5 i den samlede erfaringen er positivt, da det også må ta hensyn til at få i det hele tatt har anvendt MR-teknologi tidligere.

6.6. Prosjektgjennomføring

Gjennomføringen av prosjektet har fungert godt. Det ble jevnt holdt møter innad i gruppen hvor verktøy fra ClickUp ble brukt for å organisere arbeidsoppgaver for gruppemedlemmene som vist i figur 6.1. Det var også jevne møter med veileder hvor fremdrift, problemstillinger og krav ble drøftet.

Kanban brett ga gruppen en visuell representasjon av arbeidsflyten, og hjalp gruppen med å identifisere flaskehalser og hindringer i prosessen. Dette ga gruppen mulighet til å justere og tilpasse arbeidsflyten etter behov, og sørge for at gruppen holdt seg på rett spor under utvikling, planlegging av møter og rapportskriving.



Figur 6.1: Utdrag fra Clickup verktøy

Underveis i prosjektet har det vært en klar og nødvendig opptrapping av tidsbruk. I retrospekt er det tydelig at en mer konsentrert innsats i prosjektets første måneder ville gitt en større fleksibilitet i prosjektets siste måneder. Til tross for dette har alle målene satt av oppdragsgiver

blitt nådd innenfor tidsrammen. I løpet av prosjektperioden har også alle frister blitt overholdt, både interne, selvpålagte og eksterne fra skole eller oppdragsgivere.

7. KONKLUSJON OG VIDERE ARBEID

Dette kapitlet går over konklusjonen av prosjektet og anbefaling for videre utvikling av programsystemet.

7.1. Konklusjon av prosjektet

Målet med prosjektet var å lage en programvare hvor en bruker kan lage virtuelle miljø rundt roboter der roboten kan reagere på, og benytte seg av disse for å gjøre prosessen ved å lære roboter nye oppgaver enklere. Dette målet ble løst relativt tidlig i utviklingsprosessen hvor det ble utviklet en enkel versjon hvor roboten stanset ved detektering av et virtuelt objekt. Ved tilbakemelding fra oppdragsgiver ble dette videreutviklet til en bedre versjon hvor roboten reagerte retningsbestemt basert på hvor roboten treffer objektet.

Ved ferdig utvikling ble det levert et programsystem med et overlevert system for objekt detektering og tilbakemelding fra robot, samt et overlevert brukergrensesnitt med flere funksjoner enn det som var planlagt.

Tilbakemeldingen fra oppdragsgiver ved ferdig utvikling var svært positiv. Oppdragsgiver påpekte at det var utviklet gode løsninger for plasseringen av robothologrammet, at den digitale tvillingen fungerte bedre enn forventet og at løsningen rundt hvordan roboten reagerer på virtuelle objekt var godt løst.

7.2. Videre arbeid

7.2.1. Implementering av andre metoder for læring

Programsystemet som ble utviklet har ikke implementert nye metoder for å lære roboten nye oppgaver eller metoder å flytte roboten på. For å flytte roboten automatisk uten fysisk input brukes programmering med det innebygde systemet Polyscope fra Universal Robots. Dette vil være et område med stort potensial for videre utvikling.

Selv om det innebygde systemet fungerer bra, og gjør det enkelt å programmere bevegelsene til roboten, er det fortsatt veldig tidkrevende og vanskelig å bruke effektivt i MR. For å forbedre dette kan det utvikles andre metoder som vil fungere bedre.

I et tidligere prosjekt for HVL Robotics ble det, som nevnt i 2.1.1, utviklet et programsystem for å uføre plukk og plasser oppgaver med AR. Dette kan være et naturlig prosjekt å integrere med vårt system, og vil være en god løsning for å effektivt flytte på roboten i AR uten å fysisk flytte på roboten med hånden.

7.2.2. Anbefaling for videre arbeid i ROS

Programsystemet som har blitt utviklet under prosjektet har stort potensial for videre utvikling. På ROS siden av prosjektet vil det være naturlig å videreutvikle funksjonaliteten til griperen på roboten slik at den kan brukes til å plukke opp og flytte virtuelle objekter. Dette vil gi stor forbedring i potensialet til programsystemet, og det vil kunne bli brukt til å lære roboter mer

avanserte oppgaver som for eksempel flytting av objekter fra og til samlebånd uten å trenge riktig fysisk miljø rundt roboten.

En annen ting som kan nevnes her er at roboten for øyeblikket reagerer likt på alle virtuelle objekter, uavhengig av objektets masse eller størrelse. Dette er et potensielt område for videre forbedring, hvor realisme kan økes ved å innføre forskjellige reaksjoner basert på objektets egenskaper, som størrelse, form, og masse.

7.2.3. Anbefaling for videre arbeid i Unity

Da det tidlig i prosjektet ble vurdert en meny der man kan konfigurere og legge til roboter, vil dette virke som et naturlig videreutviklings mål. Den originale ideen der man velger hvilken type robot man ønsker å bruke kan forbli uendret. URDF-modellen til roboten må importeres i Unity og settes opp før den kan legges til av listen av roboter som applikasjonen støtter.

8. REFERANSER

- *Benjaminsen, S. H., & Dankel, H. (2022). Robotarm baneplanlegging med blandet virkelighet (Bacheloroppgave, HVL). Hentet fra https://hdl.handle.net/11250/3020798*
- *Tholey, Gregory BS; Desai, Jaydev P. PhD; Castellanos, Andres E. MD. (2005) 'Force Feedback plays a Significant Role in Minimally Invasive Surgery: Results and Analysis', Annals of Surgery vol. 241, p102-109. doi: 10.1097/01.sla.0000149301.60553.1e.*
- *Semple, K., Dodds, Z., Creasey, R., Redman, J., Cullen, N., Grobler, L., & Angell, C. (2022). MRTK2-Unity Developer Documentation - MRTK 2, MRTK2-Unity Developer Documentation - MRTK 2 | Microsoft Learn. Microsoft. Hentet fra https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05 (Hentet: 26. april, 2023).*
- *Dattalo, A. (2018). ROS Introduction. ROS. Hentet fra http://wiki.ros.org/ROS/Introduction (Hentet: 26. april 2023).*
- *Shahria, M., Sunny, M., Zarif, M., Khan, M., Modi, P., Ahamed, S., & Rahman, M. (2022) 'A Novel Framework for Mixed Reality–Based Control of Collaborative Robot: Development Study', JMIR Biomedical Engineering, 7(1), e36734. doi: 10.2196/36734. Tilgjengelig på: https://biomedeng.jmir.org/2022/1/e36734 (Hentet: 20. februar 2023)*
- *Billing, Erik A. and Hellström, Thomas. "A Formalism for Learning from Demonstration" Paladyn, Journal of Behavioral Robotics, vol. 1, no. 1, 2010, S. 1-13. https://doi.org/10.2478/s13230-010-0001-5*
- Unity Technologies. (2018). Prefabs. Unity User Manual. Tilgjengelig på https://docs.unity3d.com/Manual/Prefabs.html (Hentet 9. mai 2023)
- Sheldon, R. (2022). What is virtual reality (VR)? - Definisjon fra WhatIs.com. TechTarget. Tilgjengelig på: https://www.techtarget.com/whatis/definition/virtual-reality (Hentet 9. mai 2023).
- Microsoft. (2023). Microsoft Hololens 2. Tilgjengelig på: https://www.microsoft.com/en-us/hololens (Hentet 7. mars 2023)

- Universal Robots. (2023). UR5E. Tilgjengelig på: https://www.universal-robots.com/no/nedlastingssenter/#/e-series (Hentet 7. mars 2023)
- Arena, F., Collotta, M., Pau, G. and Termine, F. (2022). An Overview of Augmented Reality. Computers, 11(2), S.28. Tilgjengelig på: https://doi.org/10.3390/computers11020028 (Hentet 9. mai 2023).
- Adobe. (2023). Mixed Reality. Tilgjengelig på https://www.adobe.com/no/products/substance3d/discover/mixed-reality.html (Hentet 9. mai 2023).
- HVL Robotics Lab. (2023). HVL Robotics - Research Lab. Tilgjengelig på https://www.hvl.no/siteassets/filer-og-dokumenter/forskning/grupper/robotics/lab/hvl_robotics_lab.jpg (Hentet 10. mars 2023)
- Schaik, J. (2021). Positioning QR Codes in Space with HoloLens 2: Building a 'Poor Man's Vuforia'. Tilgjengelig på: https://localjoost.github.io/Positioning-QR-codes-in-space-with-HoloLens-2-building-a-'poor-man's-Vuforia'/ (Hentet: 16. mai 2023)
- Capgemini Research Institute. (2022). Digital Twins: Driving Innovation, Efficiency, and Sustainability. Tilgjengelig på: https://prod.ucwe.capgemini.com/wp-content/uploads/2022/05/Capgemini-Research-Institute_DigitalTwins_Web.pdf (Hentet 21. mars 2023)

9. VEDLEGG

- Prosjekthåndbok
- Kravdokumentasjon
- Visjonsdokument
- Kode
- Demo

10. Ordliste

Ord	Forklaring
Learning by Demonstration	Roboten lærer en oppførsel fra en eller flere demonstrasjoner, som vanligvis blir utført av et menneske eller en lærer. (Billing and Hellström, 2010)
Robot Operating System (ROS)	ROS er ett åpen kildekode robotikk rammeverk som baserer seg på noder og informasjonsflyten mellom disse. (Dattalo, 2018)
Unity	Unity er en spillmotor som kan brukes til å utvikle tre dimensjonale (3d), og to dimensjonale (2) virtuelle miljøer for en rekke plattformer.

Force feedback	Force feedback er en type simulering av kreftene man kan bli utsatt for i den virkelige verden.
Minimalt invasiv kirurgi	Minimalt invasiv kirurgi går ut på å gjøre operasjoner med mindre snitt ved hjelp av små kameraer og roboter.
IDE	En IDE er ett miljø som kan forstå kode og tilbyr en rekke verktøy for å hjelpe med utvikling av kode.
Skript	Ett skript er en fil som inneholder kode som skal bli utført når skriptet blir kallet.
Mixed Reality Toolkit (MRTK)	MRTK er en rekke verktøy utviklet av Microsoft for å akselerere utvikling av MR applikasjoner i Unity. (Semple et al., 2022)
Universal Robots RTDE (Real-Time Data Exchange)	UR_RTDE er en protokoll som lar utviklere utveksle data mellom UR-roboter og eksterne applikasjoner.
Virtuell maskin	En virtuell maskin er en virtuell emulering av en fysisk datamaskin som kjører på en annen datamaskin eller server.
Sammarbeidende robot	En samarbeidende robot, er en type robot som er designet og programmert til å samarbeide og jobbe trygt sammen med mennesker i et felles arbeidsområde.
Unity scene	En "Unity scene" er et digitalt miljø- eller spilleområde som er opprettet og utviklet ved hjelp av Unity.
Unity Component	En “component" i Unity er en individuell modul eller enhet i Unity som gir funksjonalitet og egenskaper til objekter i en scene.
Digital tvilling	En virtuell robot som speiler sin fysiske motpart.
Preset	Betyr forhåndsinnstilt - brukes i rapporten i kontekst med forhåndsinnstilte virtuelle miljøer
AR-markør	Bilde som AR eller MR-enhet kan gjenkjenne og plassere virtuelle objekter i en posisjon i forhold til bildet.